Programming

Manual

Distributed Control Systems

- DM50 Series Industrial RTU Router
- DM100 and RTU300 Series RTU (Remote Terminal Unit)
- DM500 Series Rack RTU

DOCUMENT NAME	DATE	VERSION
MIKRODEV_SM_RTU_PM_EN v1.2	12 / 2024	Telediagram 19.0.0 (Official Build)

CONTENTS

Preface

Mikrodev telemetry and automation systems are equipped with high electromagnetic compatibility, powerful PLC features and multiple communication channels and protocols.
Thanks to Telediagram (Mikrodev RTU Programming Editor), open and expandable automation and telemetry applications can be developed easily and quickly.
Mikrodev DM50, DM100, DM500 and RTU300 series DCS products have different types of I/O cards and communication port options, and I/O numbers can be expanded by adding expansion modules to all products.

ELECTRICAL POWER AUTOMATION

Mikrodev Distributed Control Systems devices, Smart Electrical Devices (protection relays, reclosers, energy and quality analyzers etc.) in the electrical sector can be read and controlled with the industry standard protocols IEC 60870, DNP3, Modbus TCP, MQTT and ICCP TASE.2. Thay can also communicate with SCADA or control center software via IEC 60870, DNP3 and MODBUS TCP protocols. Mikrodev RTU products are preferred in electrical energy applications with their easy, flexible and fast programming capabilities and I/O expansion possibilities.

WATER AND WASTE SECTOR

Mikrodev RTU products are used in the remote monitoring and control of pumping stations, storages, wells, pipelines, meters and flow meters, valves, dosing and water quality measuring stations. According to the Master/Slave operating principle, you can create simultaneous and multiple communication channels between stations such as wells, warehouses, lift-pump centers. At this point, the entire water network is managed in a distributed manner. Even when the central SCADA service is out of service, stations on the field continue to work by communicating with each other. Once SCADA is active again, past events and data can be transmitted to the system without loss of data with time-tagged messages of IEC104 and DNP3.

In the programming of DCS series devices, Function Block Diagram - FBD language which is defined in IEC 61131-3 standard is used. Thanks to the programming with the FBD language, the project can be developed easily and quickly with the drag and drop logic.
In this document, the function block library elements used in programming Mikrodev DM50 series industrial RTU router, DM500 series rack RTU, DM100 and RTU300 series RTUs with FBD are explained.

Please follow our website www.mikrodev.com for the up to date version of the document.

About Mikrodev

Since 2006, MIKRODEV has been developing and manufacturing industrial control and communication products. MIKRODEV serves the system integrators in the public and private sector, OEM and end users.

Our products are manufactured complying with the quality standards required by the industrial automation industry and the quality of our products are proved on the field for many years

MIKRODEV is one of the few companies in the world that has its own designed IEC 61131-3 compliant library for its programmable logic control devices. In addition, the open, flexible, programmable SCADA solution developed by MIKRODEV is also available to customers.

MIKRODEV products' performance and wide range of applications make them possible for customers to achieve faster, simplified and cost-effective results.

WARNING!

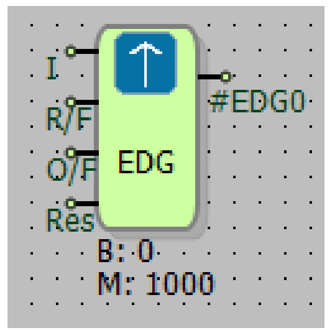
- ✓ Use the programming editor only for Mikrodev Certifed devices
- ✓ When you change your physical hardware configuration, update your development to the appropriate version.
- ✓ The developed program should be tested separately before taking to field service and should be shipped to the field after the tests are successfully completed.
- ✓ Take all accident prevention measures and safety measures identified by local law

Failure to comply with these rules may result in death, serious injury or property damage

1 LOGIC GATE BLOCKS

1.1 EDGE GATE

1.1.1 Connections

I: Signal input R/F: Rising or falling edge selection O/F: One/Full cycle selection Res: Reset pin		#EDG0: Output of the block

1.1.2 Connection Explanations

I: Signal input

It is the input that detects edge state.

R/F: Rising or/and falling edge selection

It is used for choosing rising or falling edge detection from outside of the block.

If 0, falling edges are detected,

If 1, rising edges are detected,

If 2, both falling and rising edges are detected.

O/F: One/full cycle selection

If it is 0, full cycle is selected. After an edge is detected, until the reset signal is detected output signal becomes and stays high(1).

If it is 1, one cycle is selected. After an edge is detected, output becomes high(1) for one clock cycle and then becomes low(0).

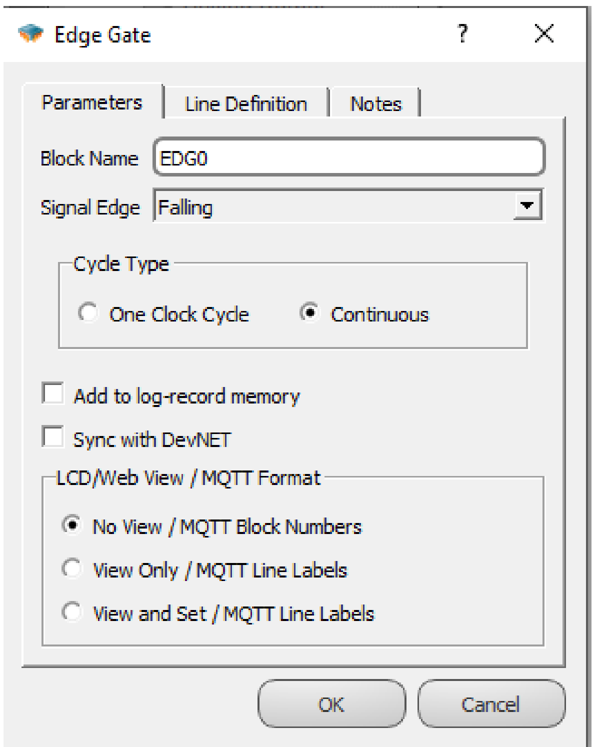
Res: Reset pin

It is used to reset the signal when full cycle is selected. Detects the high(1) signal.

<u>#EDG0: Output of the block</u>

It is a binary output

1.1.3 Block Settings

<table>
<tr>
<td rowspan="2">
</td>
<td>Signal Edge: R/F: It has the same purpose with rising or/and falling edge selection pin. Rising, Falling or Rising/Falling options are available.</td>
</tr>
<tr>
<td>Cycle Type: O/F: It has the same purpose with one/full cycle selection pin.

One cycle or full cycle options are available.</td>
</tr>
</table>

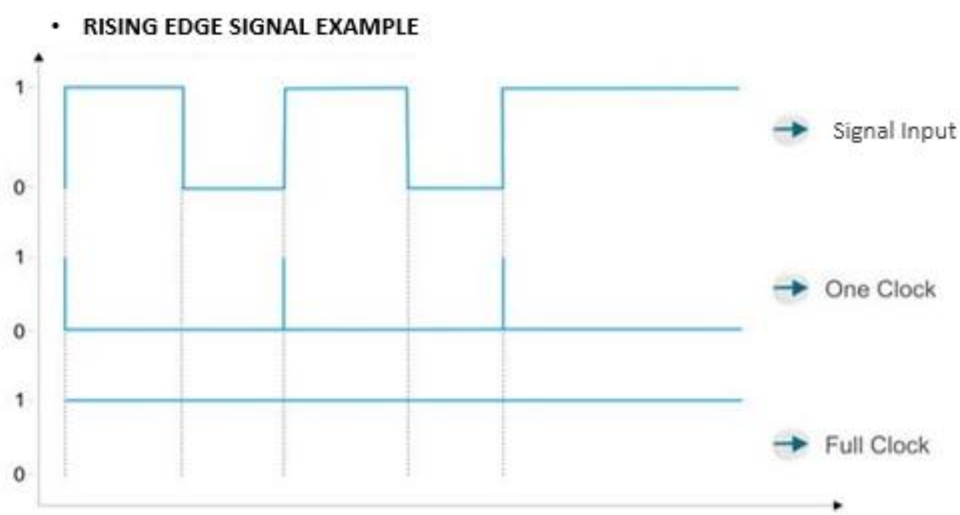
1.1.4 Block Explanation

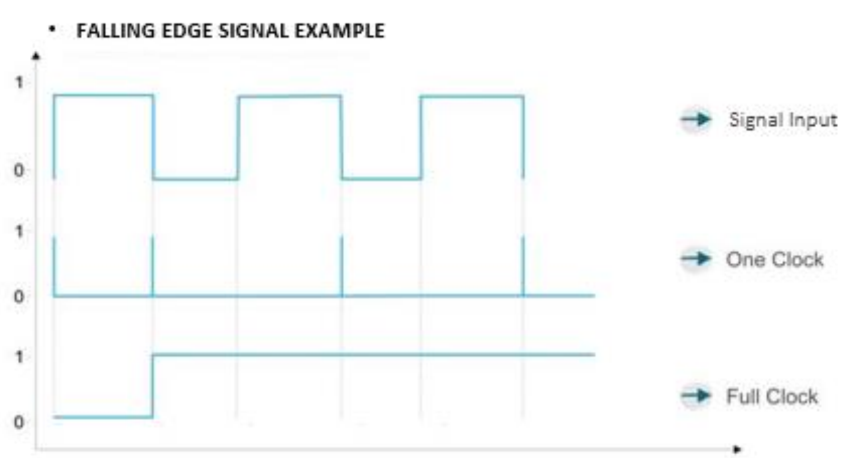
Edge Gate block is used for edge triggering purposes. It detects the rising or the falling edge of a signal and stays high for one clock cycle or full clock cycle. “R/F” input and “O/F” input specifies the edge to be detected and cycle type of the output signal. “R/F” input and “O/F” input can be adjusted in Block settings or can be adjusted by connecting a high or low signal to the block inputs.

1.1.4.1 Truth Table

Previous I	Current I	R/F	O/F	Res	Previous #EDG0	Current #EDG0
0	1	0	X	0	0	1
1	1	0	0	0	1	1
1	1	0	1	0	1	0
1	0	0	X	0	0	0
1	0	1	X	0	0	1
0	1	2	X	0	0	1
1	0	2	X	0	0	1
X	X	X	X	1	X	0

1.1.4.2 Signal Flow Diagram

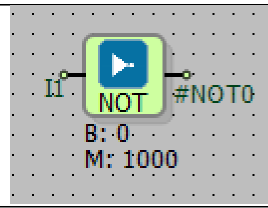




,

1.2 NOT GATE

1.2.1 Connections

I1: Signal input		#NOT0: Output of the block

1.2.2 Connection Explanations

<u>I1: Signal input</u>

It is the input of the NOT gate.

<u>#NOT0: Output of the block</u>

It is the output of the NOT gate.

1.2.3 Block Settings

There are no block settings.

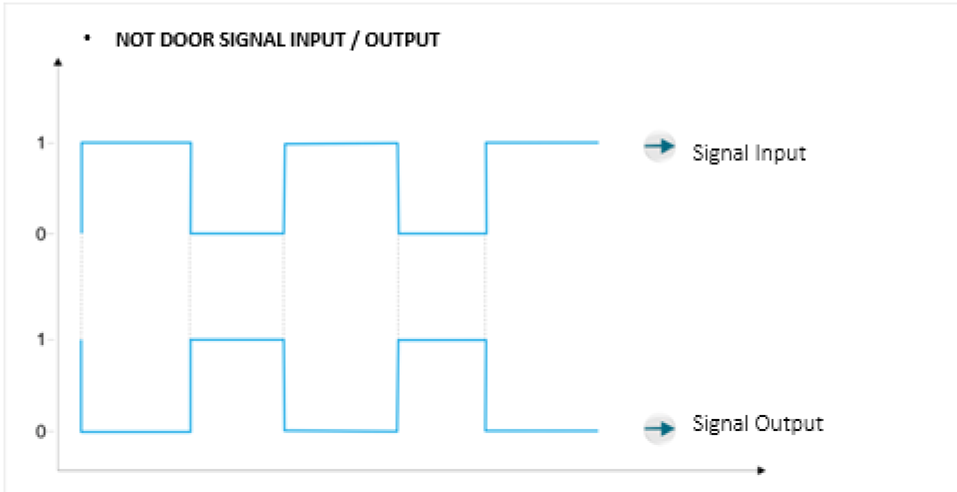
1.2.4 Block Explanation

Not Gate block is used for inverting the input signals. If the input signal is high(1) the output will be low(0) and if the input signal is “0” the output will be “1”.

1.2.4.1 Truth Table

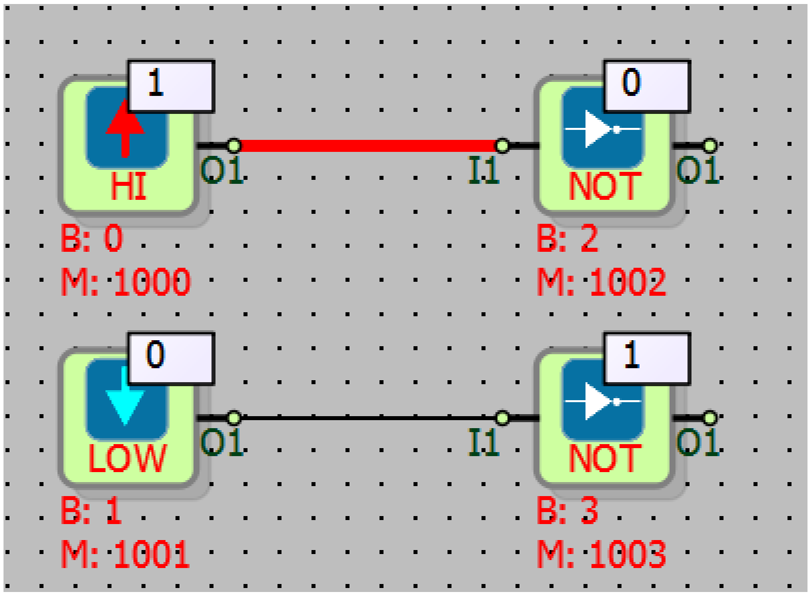
I1	#NOT0
1	0
0	1

1.2.4.2 Signal Flow Diagram



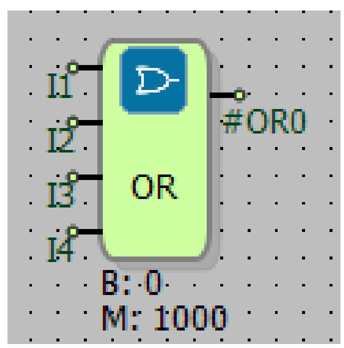
1.2.5 Sample Application

In the example, HIGH and LOW signals are inverted using NOT Gate.



1.3 OR GATE

1.3.1 Connections

I1: Signal input		#OR0: Output of the block
I2: Signal input		
I3: Signal input		
I4: Signal input		

1.3.2 Connection Explanations

<u>I1: Signal input</u>

It is the input of the OR gate.

<u>I2: Signal input</u>

It is the input of the OR gate.

I3: Signal input

It is the input of the OR gate.

I4: Signal input

It is the input of the OR gate.

#OR0: Output of the block

It is the output of the OR gate.

1.3.3 Block Settings

There are no block settings.

1.3.4 Block Explanation

Performs the logic OR operation to the input signals. Truth tables for this gate can be seen in tables below

1.3.4.1 Truth Table for Two Inputs

Input 1	Input 2	Output 1
0	0	0
0	1	1
1	0	1
1	1	1

1.3.4.2 Truth Table for Three Inputs

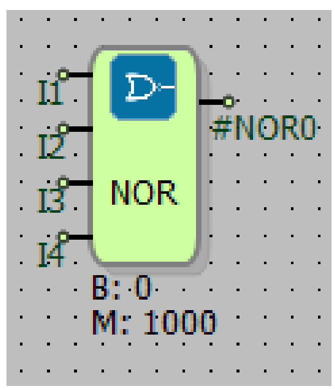
Input 1	Input 2	Input 3	Output 1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

1.3.4.3 Truth Table for Four Inputs

Input 1	Input 2	Input 3	Input 4	Output 1
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

1.4 NOR GATE

1.4.1 Connections

I1: Signal input		#NOR0: Output of the block
I2: Signal input		
I3: Signal input		
I4: Signal input		

1.4.2 Connection Explanations

<u>I1: Signal input</u>

It is the input of the NOR gate.

<u>I2: Signal input</u>

It is the input of the NOR gate.

I3: Signal input

It is the input of the NOR gate.

I4: Signal input

It is the input of the NOR gate.

#NOR0: Output of the Block

It is the output of the NOR gate.

1.4.3 Block Settings

There are no block settings.

1.4.4 Block Explanation

NOR Gate is a combination of an OR Gate and a NOT Gate. It gives output as if a NOT gate is connected to the output of an OR gate. To use this block, at least two inputs must be connected. When all the inputs are low(0), output will be high(1). Truth tables for this gate can be seen in diagram below.

1.4.4.1 Truth Table for Two Inputs

Input 1	Input 2	Output 1
0	0	1
0	1	0
1	0	0
1	1	0

1.4.4.2 Truth Table for Three Inputs

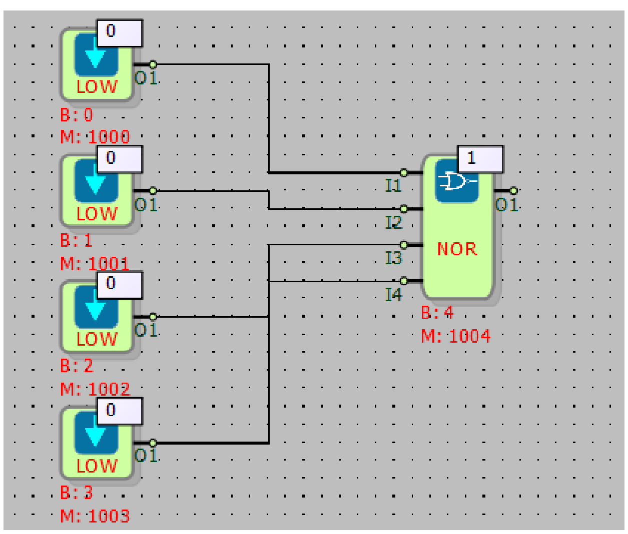
Input 1	Input 2	Input 3	Output 1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

1.4.4.3 Truth Table for Four Inputs

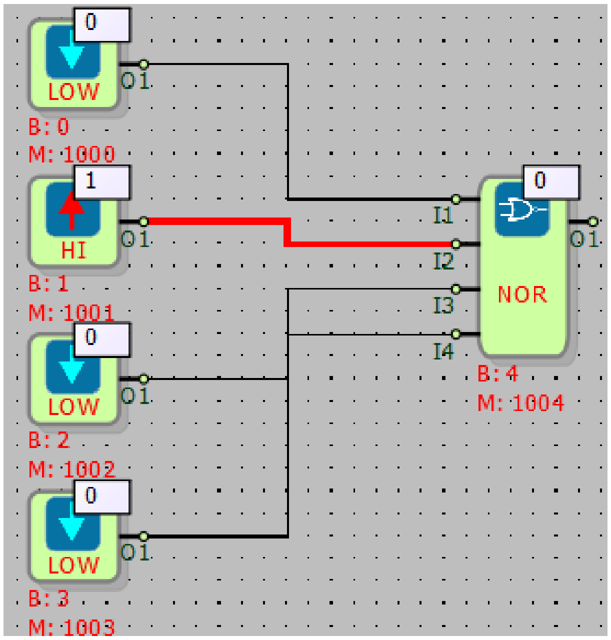
Input 1	Input 2	Input 3	Input 4	Output 1
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

1.4.5 Sample Application

1.4.5.1 High Output

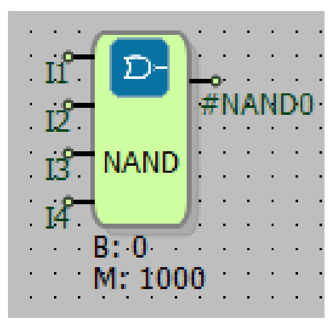


1.4.5.2 Low Output



1.5 NAND GATE

1.5.1 Connections

<table>
<tr><td>I1: Signal input</td><td rowspan="4">
</td><td rowspan="4">#NAND0: Output of the block</td></tr>
<tr><td>I2: Signal input</td></tr>
<tr><td>I3: Signal input</td></tr>
<tr><td>I4: Signal input</td></tr>
</table>

1.5.2 Connection Explanations

<u>I1: Signal input</u>

It is the input of the NAND gate.

<u>I2: Signal input</u>

It is the input of the NAND gate.

I3: Signal input

It is the input of the NAND gate.

I4: Signal input

It is the input of the NAND gate.

#NAND0: Output of the block

It is the output of the NAND gate.

1.5.3 Block Settings

There are no block settings.

1.5.4 Block Explanation

Performs the logic NAND operation to the input signals. It is a combination of an AND Gate and a NOT Gate. It gives output as if a NOT gate is connected to the output of an AND gate. Output becomes low(0) only when all the inputs are high(1) otherwise the output is always high(1). To use this block, at least two inputs must be connected. When two inputs are connected, other inputs can be left unconnected. Truth tables for this gate can be seen in diagram below.

1.5.4.1 Truth Table for Two Inputs

Input 1	Input 2	Output 1
0	0	1
0	1	1
1	0	1
1	1	0

1.5.4.2 Truth Table for Three Inputs

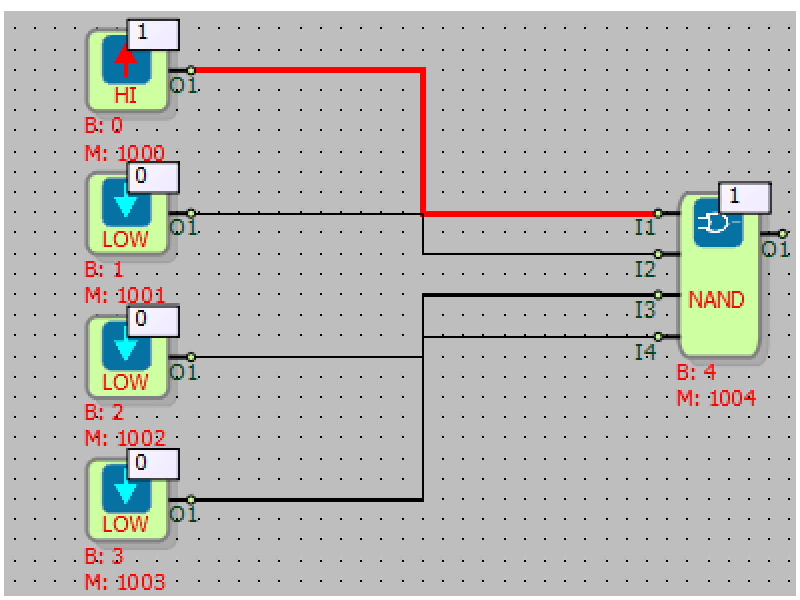
Input 1	Input 2	Input 3	Output 1
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

1.5.4.3 Truth Table for Four Inputs

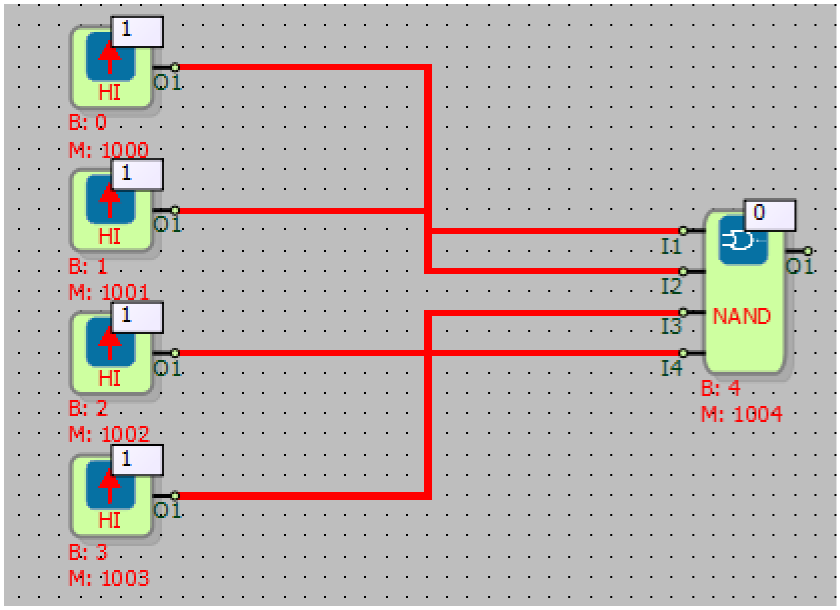
Input 1	Input 2	Input 3	Input 4	Output 1
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

1.5.5 Sample Application

1.5.5.1 HIGH Output

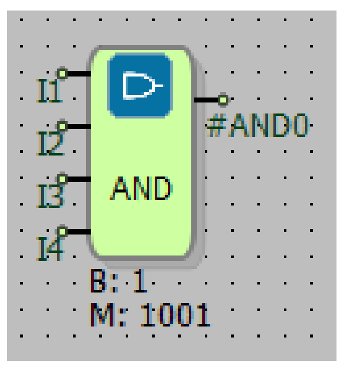


1.5.5.2 LOW Output



1.6 AND GATE

1.6.1 Connections

I1: Signal input		#AND0: Output of the block
I2: Signal input		
I3: Signal input		
I4: Signal input		

1.6.2 Connection Explanations

I1: Signal input

It is the input of the AND gate.

I2: Signal input

It is the input of the AND gate.

I3: Signal input

It is the input of the AND gate.

I4: Signal input

It is the input of the AND gate.

#AND0: Output of the block

It is the output of the AND gate.

1.6.3 Block Settings

There are no block settings.

1.6.4 Block Explanation

Performs the logic AND operation to the input signals. To use this block, at least two inputs must be connected. Truth tables for this gate can be seen in diagrams below.

1.6.4.1 Truth Table for Two Inputs

Input 1	Input 2	Output 1
0	0	0
0	1	0
1	0	0
1	1	1

1.6.4.2 Truth Table for Three Inputs

Input 1	Input 2	Input 3	Output 1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

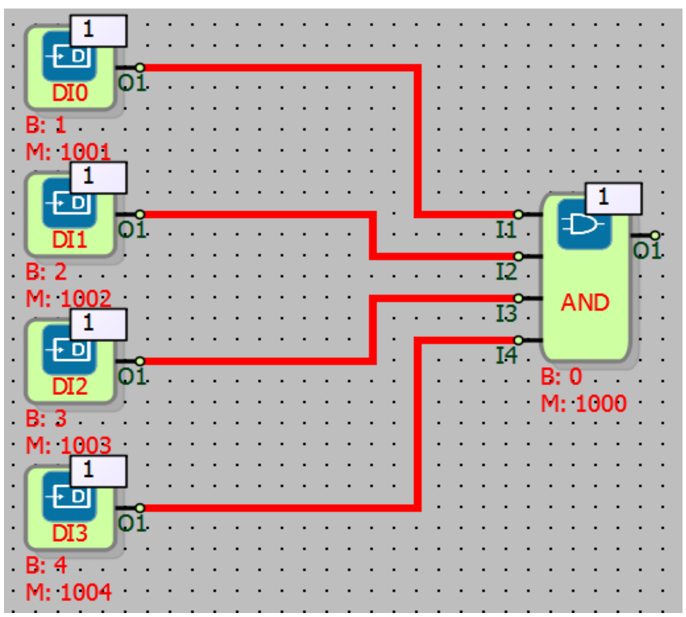
1.6.4.3 Truth Table for Four Inputs

Input 1	Input 2	Input 3	Input 4	Output 1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0

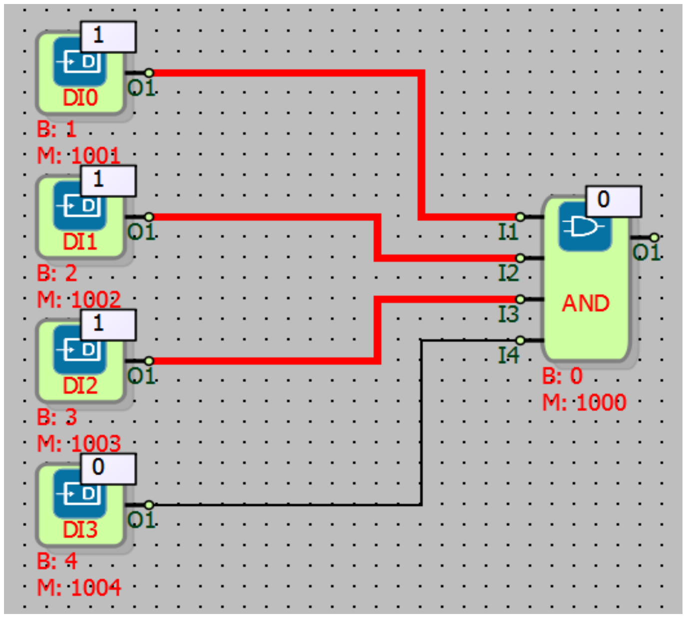
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

1.6.5 Sample Application

1.6.5.1 HIGH Output

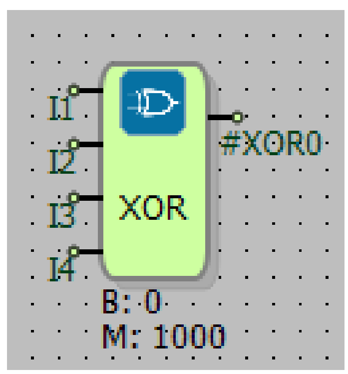


1.6.5.2 LOW Output



1.7 XOR GATE

1.7.1 Connections

I1: Signal input		#XOR0: Output of the block
I2: Signal input		
I3: Signal input		
I4: Signal input		

1.7.2 Connection Explanations

<u>I1: Signal input</u>

It is the input of the XOR gate.

<u>I2: Signal input</u>

It is the input of the XOR gate.

<u>I3: Signal input</u>

It is the input of the XOR gate.

I4: Signal input

It is the input of the XOR gate.

#XOR0: Output of the block

It is the output of the XOR gate.

1.7.3 Block Settings

There are no block settings.

1.7.4 Block Explanation

Performs the logic XOR operation to the input signals. Output becomes high(1) when odd numbers of high(1) signals present in the input signals. For example, if three inputs are connected and only one of the inputs are high(1), then the output becomes high(1). To use this block, at least two inputs must be connected. When two inputs are connected, other inputs can be left unconnected. Truth tables for this gate can be seen in diagram below.

1.7.4.1 Truth Table for Two Inputs

Input 1	Input 2	Output 1
0	0	0
0	1	1
1	0	1
1	1	0

1.7.4.2 Truth Table for Three Inputs

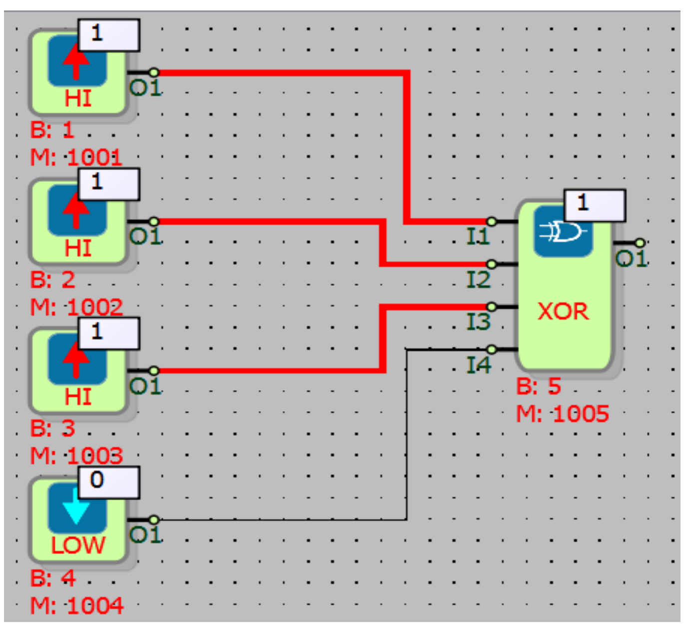
Input 1	Input 2	Input 3	Output 1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

1.7.4.3 Truth Table for Four Inputs

Input 1	Input 2	Input 3	Input 4	Output 1
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

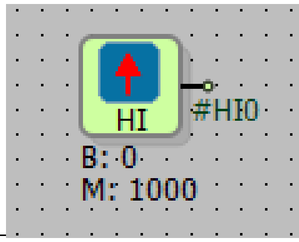
1.7.5 Sample Application

1.7.5.1 HIGH Output



1.8 HIGH GATE

1.8.1 Connections

	#HI0: Output of the block

1.8.2 Connection Explanations

<u>#HI0: Output of the block</u>

It is output of the High gate.

1.8.3 Block Settings

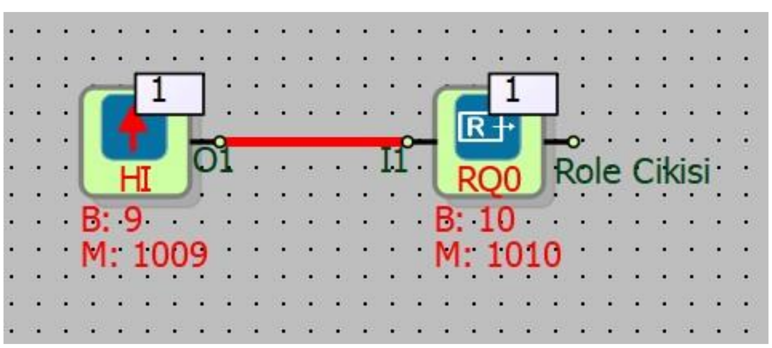
There are no block settings.

1.8.4 Block Explanation

The block output is always high(1).

1.8.5 Sample Application

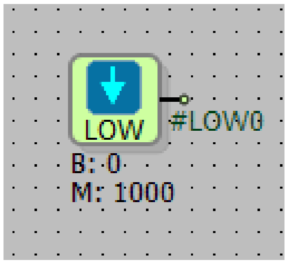
1.8.5.1 HIGH Output



The output of the High Gate block is connected to the input of the Relay Output block. If Mikrodev PLC system is ON, the output value of the Relay Output block will be high(1), otherwise, the output of the Relay Output block will be low(0).

1.9 LOW GATE

1.9.1 Connections

	#LOW0: Output of the block

1.9.2 Connection Explanations

#LOW0: Output of the block

It is output of the High gate.

1.9.3 Block Settings

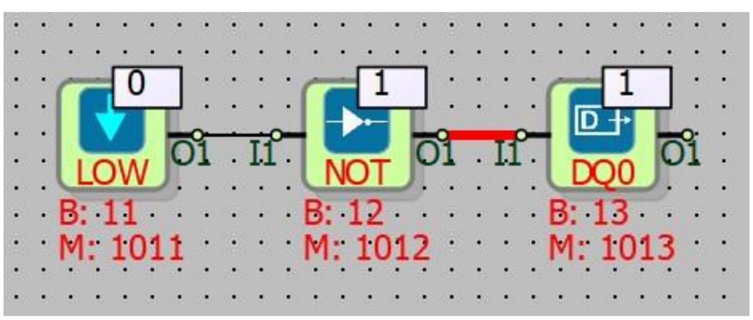
There are no block settings.

1.9.4 Block Explanation

The block output is always low(0).

1.9.5 Sample Application

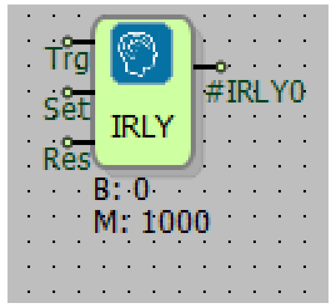
1.9.5.1 LOW Output



The output of the Gate Low block is connected to the input of the Not Gate block. The output of the Not Gate block is also connected to the input of the Digital Output block. If Mikrodev PLC system is on, the Digital Output block output will be high (1), otherwise the Digital Output block output will be low (0).

1.10 IMPULSE RELAY

1.10.1 Connections

<table>
<tr><td>Trg: Trigger input</td><td rowspan="3">
</td><td rowspan="3">#IRLY0: Output of the block</td></tr>
<tr><td>Set: Block set input</td></tr>
<tr><td>Res: Block reset input</td></tr>
</table>

1.10.2 Connection Explanations

<u>Trg: Trigger input</u>

Retrieves not the current state of the block output when a rising edge trigger is sent to the "Trg" input.

<u>Set: Block set input</u>

It is the block input that always makes the block output high (1) in rising edge triggering

<u>Res: Block reset input</u>

It is the block input that always makes the block output low (0) in rising edge triggering.

<u>#IRLY0: Output of the block</u>

It is the block output that produces a low (0) or high (1) output depending on the status of the block inputs.

1.10.3 Block Settings

There are no block settings

1.10.4 Block Explanation

Impulse Relay block is used for operations such as on-off, set and reset. It is a gate that gives logic output.

Block output changes position in rising edge trigger coming to "Trg" block input. When the block output is low (0), when a rising edge trigger (logic 1) signal is applied to the "Trg" input of the block, the block output “#IRLY0” goes high (1). While the block output is high (1), the block output “#IRLY0” goes low (0) when a rising edge trigger (1) signal is applied to the “Trg” block input.

When the “Set” block input is high (1), the block output “#IRLY0” always goes high (1) if the “Sif” input of the block is not high (1). When the “Set” block input is in the high (1) position, the block output “#IRLY0” is high (1) regardless of the position of the “Trg” block input.

Block output “#IRLY0” always goes to low (0) state in case of rising edge trigger coming to the “Res” input of the block. When the “Res” block input is high (1), the block output “#IRLY0” is always low (0) even if the other inputs are high (1).

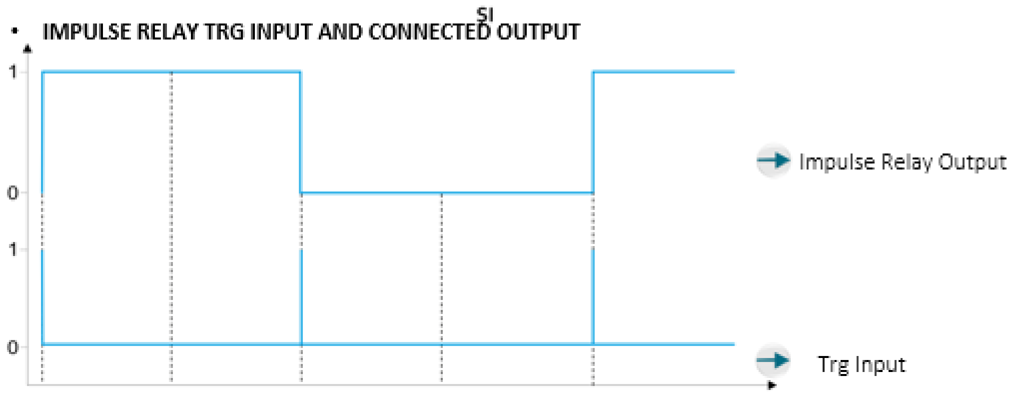
1.10.4.1 Truth Table

The operations in the truth table are done in order from top to bottom in the table.

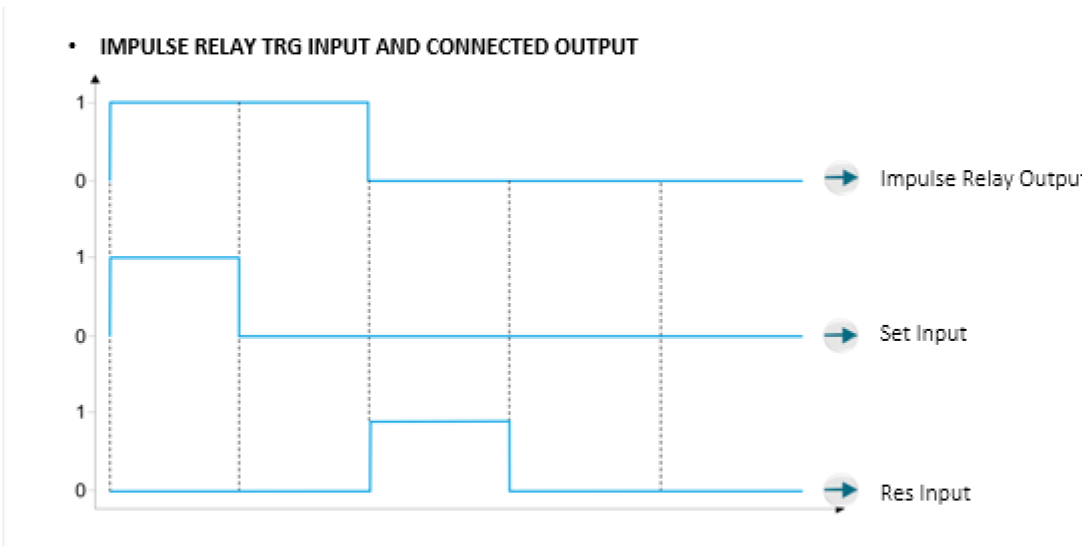
Trg	Set	Sıf	#IRLY0
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	0
0	0	0	0
1	0	0	1
0	0	1	0
0	1	0	1
1	1	0	1
1	1	1	0

1.10.4.2 Signal Flow Diagram:

Block Output with Trg Input (#IRLY0)

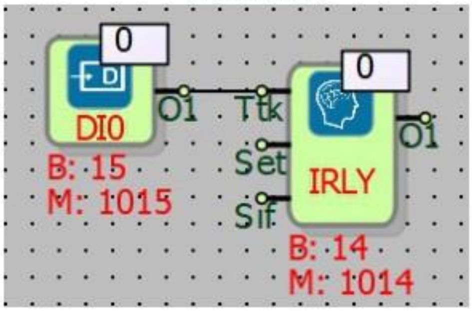


Block Output with Set and Res Input (#IRLY0)

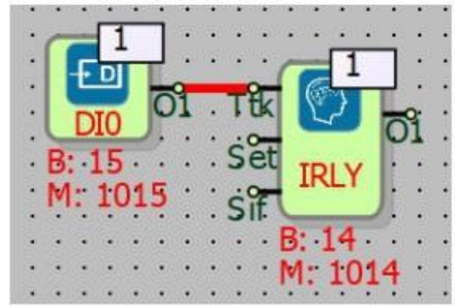


1.10.5 Sample Application

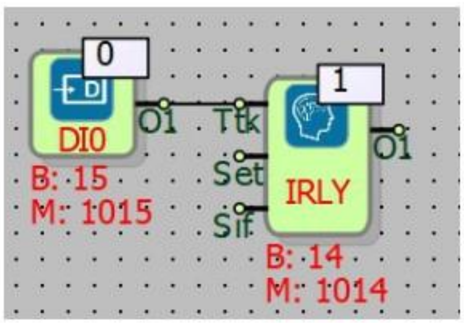
1.10.5.1 Trg Input



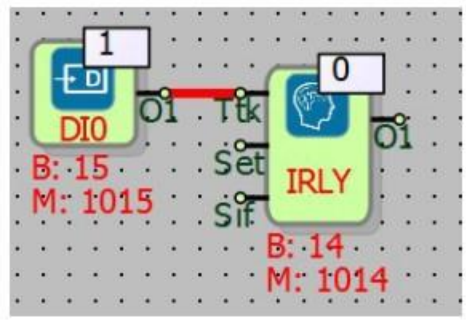
(1)



(2)



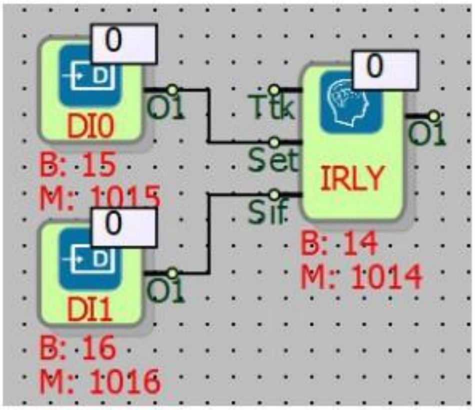
(3)



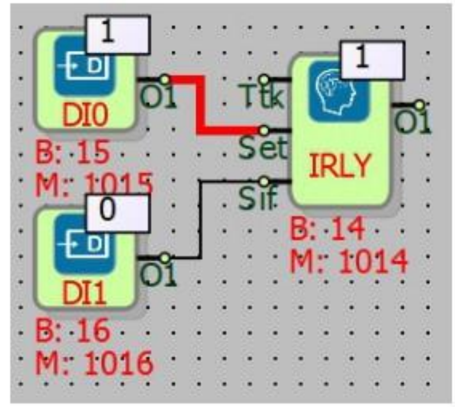
(4)

The “#IRLY0” position of the block output is observed in the example, depending on the rising edge trigger coming to the “Trg” block input. Initially, the “Trg” block input and the block output “#IRLY0” are low (0), while the “Trg” input of the block is high (1), the block output “#IRLY0” is also high (1). When the “Trg” block input goes low (0), the block output “#IRLY” stays high (1). When the “Trg” block input goes to high (1) again, the block output “Q1” goes to the low (0) position. When the “Trg” block input goes low (0) again and then goes high (1) again, the block output “#IRLY0” will go high (1) again.

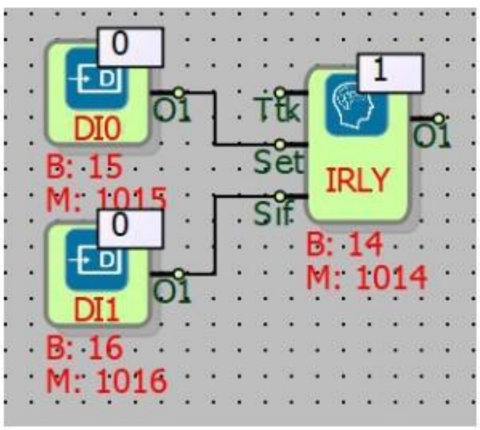
1.10.5.2 Set Input



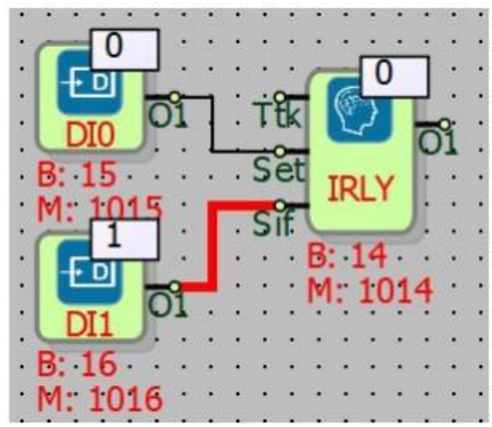
(1)



(2)



(3)

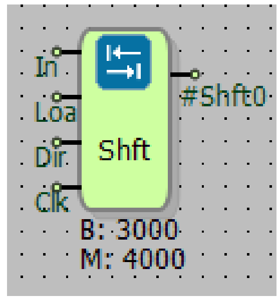


(4)

In the example, with the rising edge trigger coming to the “Set” input of the block, the block output “#IRLY0” has moved to the high (1) position. Although the “Set” block input went low (0), the block output “#IRLY0” kept its high (1) position. When a high (1) signal is applied to the “Res” block input, the block output “O1” is set to low (0).

1.11 SHIFT BLOCK

1.11.2 Connections

In: Value input to shift Loa: Value loading input Dir: Direction input Clk: Start shifting input		#Shft0: Output of the block

1.11.3 Connection Explanations

<u>In: Value input to shift</u>

The “In” block input is the value input to be shifted.

<u>Loa: Value loading input</u>

In order for the value of the "In" input of the block to be shifted to be loaded into the block, a rising edge trigger must be given to this input.

<u>Dir: Direction input</u>

Bloğun “Dir” girişi, “In” blok girişindeki değerin kaydırılacağı yönü belirlemek için kullanılır.

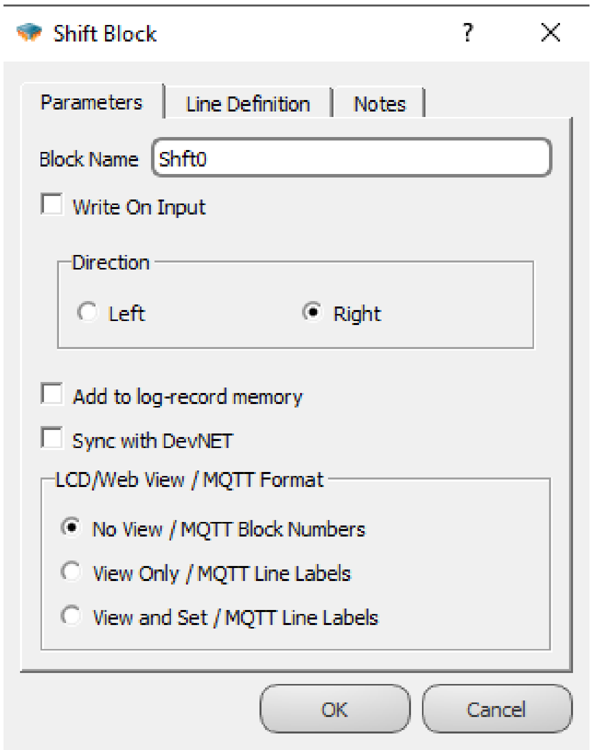
<u>Clk: Start shifting input</u>

The “Clk” block input starts the shift of the value in the “In” block input, which is enclosed in each rising edge trigger..

<u>#Shft0: Output of thr block</u>

The output of the block “#Shft0” is the output of the block to which the shifted value is transferred.

1.11.4 Block Settings

	Write On Input: If selected, the shifted value overwrites the value in the "In" input of the block.
	Direction: Right: If when selected, shifting is done to the right. (divide by two.) Left: If when selected, shifting is done to the left. (divide by two)

1.11.5 Block Explanation

The Shift block is used when a value is shifted to the right or left. Shift means shifting one bit right or left, i.e., multiplying by 2 or dividing by 2.

In input: It is the input of the value to be shifted. Since the block output is a 16-bit word, the value to be shifted should be defined accordingly.

Loa input: It is used to include the value of the "In" input of the load, that is, the block to be shifted, into the block.

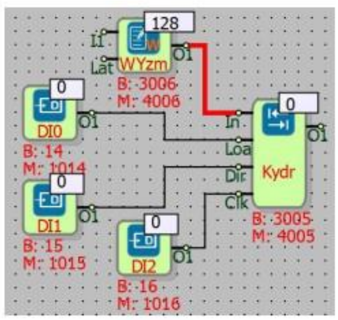
Clk input: Performs scrolling on each rising edge trigger.

The shifted value is transferred to the "#Shft0" output.

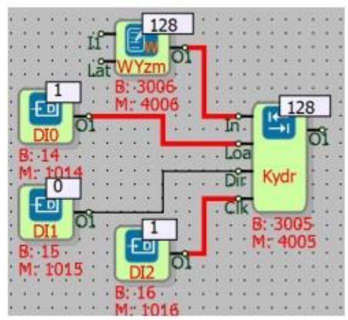
The working logic of the shift block, the register data at the input of the "In" block, when a high level signal is applied to the "Loa" input of the block, the data to be shifted is taken into the block. When the rising edge trigger is applied to the "Clk" block input, the "Dir" block input value is shifted according to the direction status. If a high level signal comes to the “Loa” input of the block while the scrolling process is in progress, the value at the “In” block input of the shift is reloaded into the block. Scrolling only once as long as information comes to the "Loa" block input.

1.11.6 Sample Application

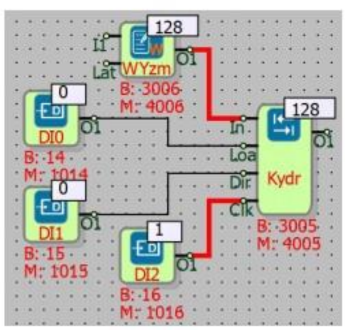
1.11.6.1 Shift Right



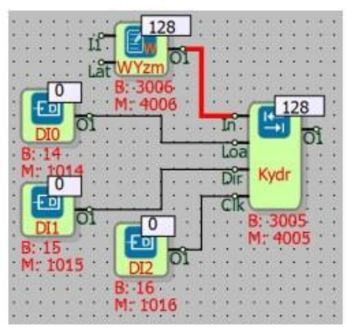
(1)



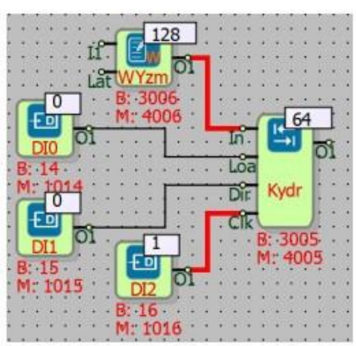
(2)



(2)



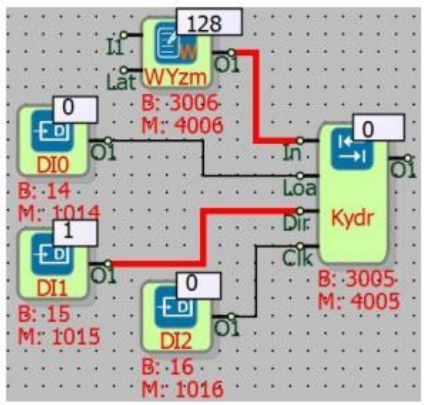
(4)



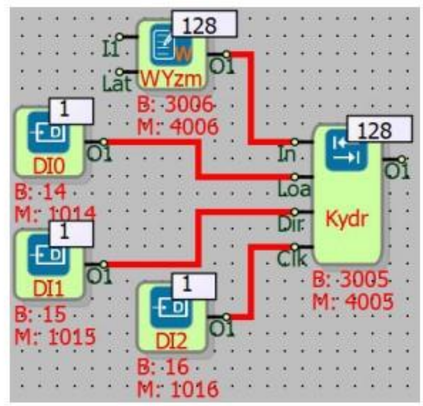
(5)

In the example, the right shift is done. The value in the "In" block input is included in the Shift block and divided by 2. After the value to be shifted is written to the “In” input of the block, the “Loa” block input is made high (1) and the value at the “In” block input is included in the Shift block. Since the value in the "In" block input is included in the shift block, the "Loa" block input is set to low (0) in picture (3). Then, in each rising edge trigger that comes to the "Clk" block input, the value in the block is shifted to the right by 1 bit (divided by 2) and the shifting process will continue until the value in the block is reset. Low (0) is selected because the "Dir" block input will be shifted to the right.

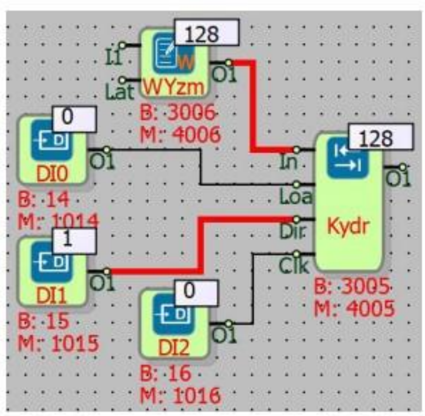
1.11.6.2 Shift Left



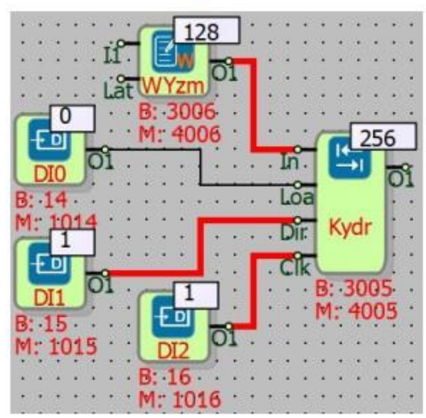
(1)



(2)



(3)

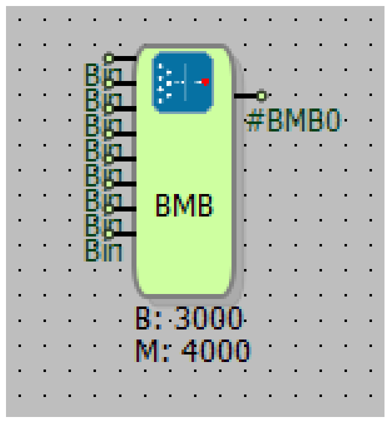


(4)

In the left shift example, firstly, the "Dir" block input is set to high (1) so that the left shift can be performed. Then, the value to start the shifting operation is written to the Word Register block connected to the “In” block input. In Picture (2), the "Loa" and "Clk" block inputs are made high (1) and the value in the "In" block input is written to the block output. In picture (3), the "Loa" block input is reset and in picture (4), the value in the block is shifted to the left in each rising edge trigger that comes to the "Clk" block input. (Multiplied by 2.)

1.12 BİT MERGE BLOCK

1.12.7 Connections

Bin: 0. Bit input		#BMB0: Output of the block
İki: 1. Bit input		
İki: 2. Bit input		
İki: 3. Bit input		
İki: 4. Bit input		
İki: 5. Bit input		
İki: 6. Bit input		
İki: 7. Bit input		

1.12.8 Connection Explanations

<u>İki: 0. Bit input</u>

0. Bit identification input

<u>İki: 1. Bit input</u>

1. Bit identification input

<u>İki: 2. Bit input</u>

2. Bit identification input

<u>İki: 3. Bit input</u>

3. Bit identification input

<u>İki: 4. Bit input</u>

4. Bit t identification input

<u>İki: 5. Bit input</u>

5. Bit identification input

İki: 6. Bit input

6. Bit identification input

İki: 7. Bit input

7. Bit identification block

#BMB0: Output of the block

Output where bits are combined and written in decimal

1.12.9 Block Settings

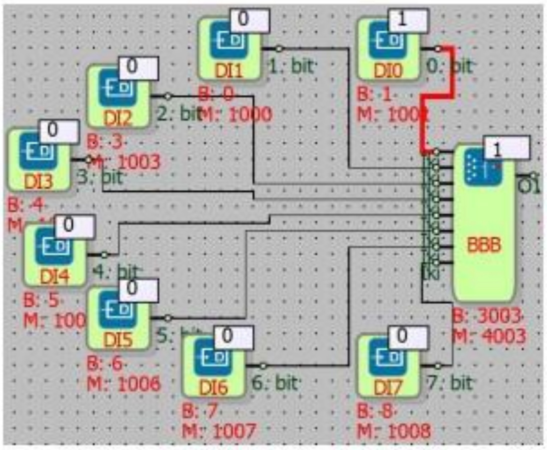
There are no block settings.

1.12.10 Block Explanation

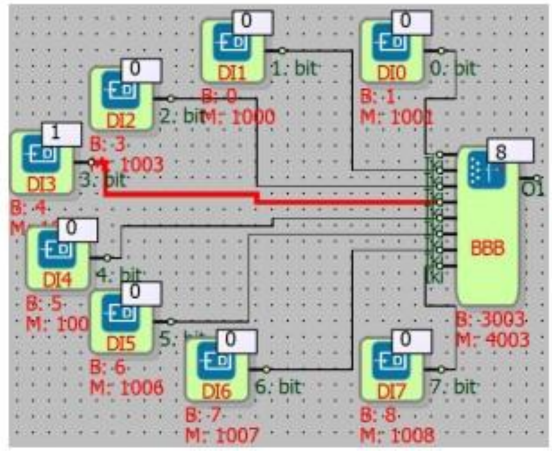
It is used to combine a maximum of 8 bits in binary and transfer them to the block output as 1 byte. If all the bits connected to the block input are high (1), the block output takes the maximum value (255). Of the block inputs, the 0th bit input is for the least significant bit (LSB), and the 7th Bit input is for the most significant bit (MSB).

1.12.11 Sample Application

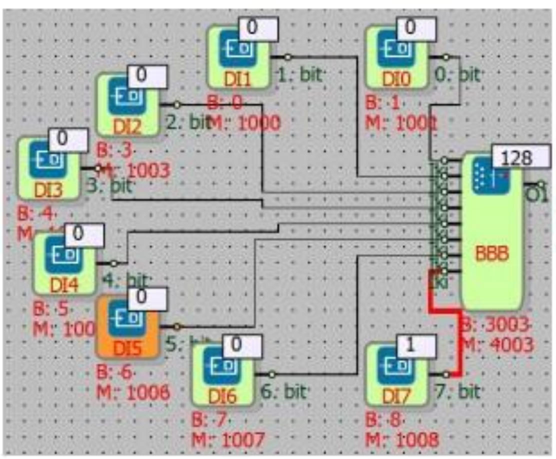
1.12.11.1 8 Bit Merge



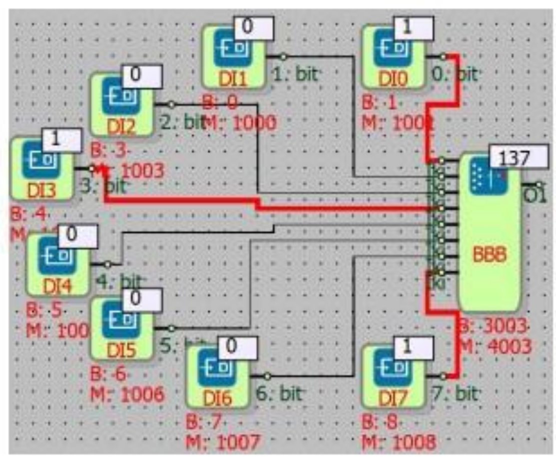
(1)



(2)



(2)



(4)

In the example, if the 0, 3 and 7 bits of the bit Bit Merge block are high (1) and the other bits are low (0), the decimal values of the bits are seen at the output of the block.

In the 1st picture; the 0th bit input is in the high (1) position; The decimal equivalent of the 0th bit is written to the 2^0=1 block output..

In the 1st picture; The 3rd bit input is in the high (1) position; The decimal equivalent of the 3rd bit is written to the 2^3=8 block output.

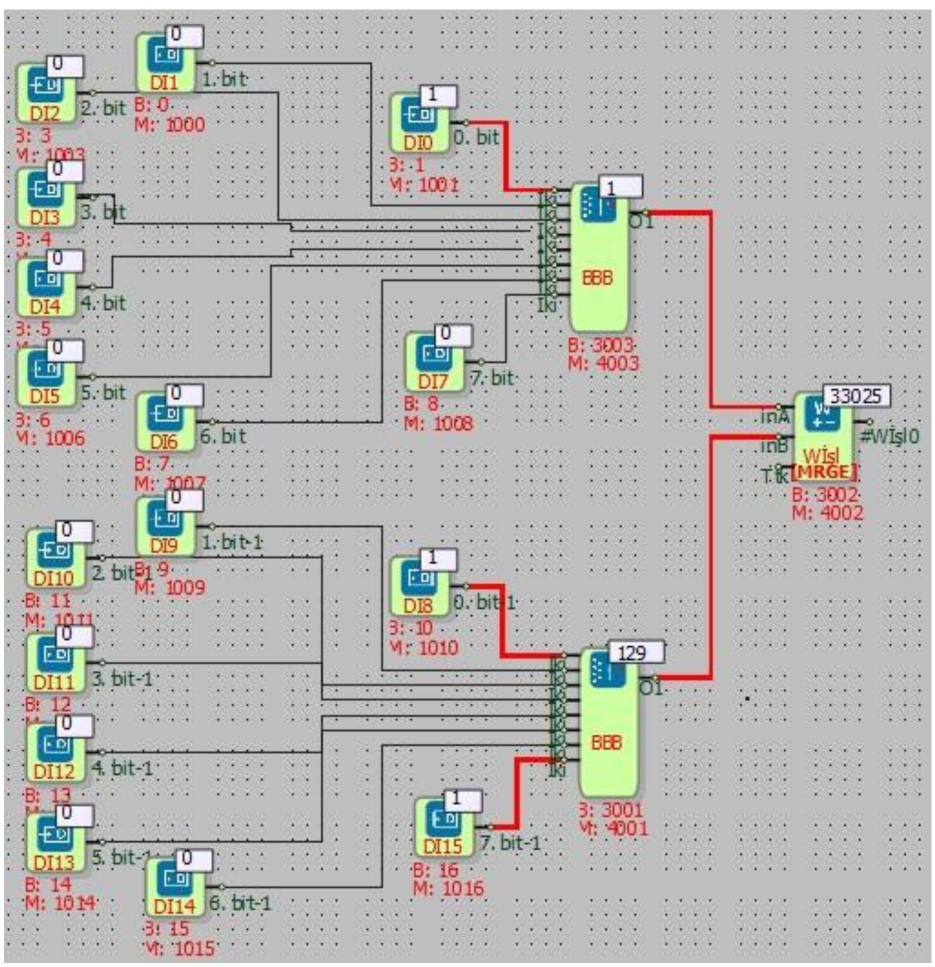
In the 2nd picture; The 7th bit input is in the high (1) position; The decimal equivalent of the 7th bit is written to the 2^7=128 block output.

In the 3rd picture; Since the 0th, 3rd, and 7th bit inputs are in the high (1) position; The decimal equivalent of the 0th, 3rd, and 7th bits is written to the block output as (1+8+128) =137.

1.12.11.2 16 Bit Merge

16 bits can be combined using 2 Bit Merge blocks. For this, the output of one of the Bit Merge block must be connected to the “InA” input of the Word Math block “Q1” and the output of the other Bit Combining block “Q1” to the “InB” input of the Word Math block. The Word Process block settings (double click on the block) and select the math type Merge A-B.

In this case, the decimal number value at the output of the Bit Merge block connected to the “InA” input of the Word Math block is transferred directly to the output of the Word Math block. The decimal number value at the output of the Bit Merge block connected to the “InB” input of the Word Math block is transferred to the output of the Word Math block by shifting 8 bits (by multiplying the decimal value of each bit by 256).



In the example, bit 0 of the Bit Merge block connected to the “InA” input of the Word Math block is high (1) and the decimal number value is 2^0=1.

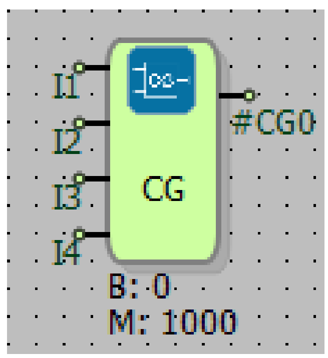
Bit 0 of the Bit Merge block connected to the “InB” input of the Word Math block is high (1) and 2^0*256=256 since this value will be transferred to the Word Math block output by multiplying by 256.

The 7th bit of the Bit Merge block connected to the “InB” input of the Word Math block is high (1) and it is 2^7*256=32768 since this value will be transferred to the Word Math block output by multiplying by 256.

The decimal value of the 3 high (1) bits; Read in Word Math block output as 1+256+32768=33025.

1.13 CUSTOM GATE

1.13.1 Connections

I1: Signal input		#CG0: Output of the block
I2: Signal input		
I3: Signal input		
I4: Signal input		

1.13.2 Connection Explanations

<u>I1: Signal input</u>

It is the 1st input of the Custom Gate block.

<u>I2: Signal input</u>

It is the 2nd input of the Custom Gate block.

<u>I3: Signal input</u>

It is the 3rd input of the Custom Gate block.

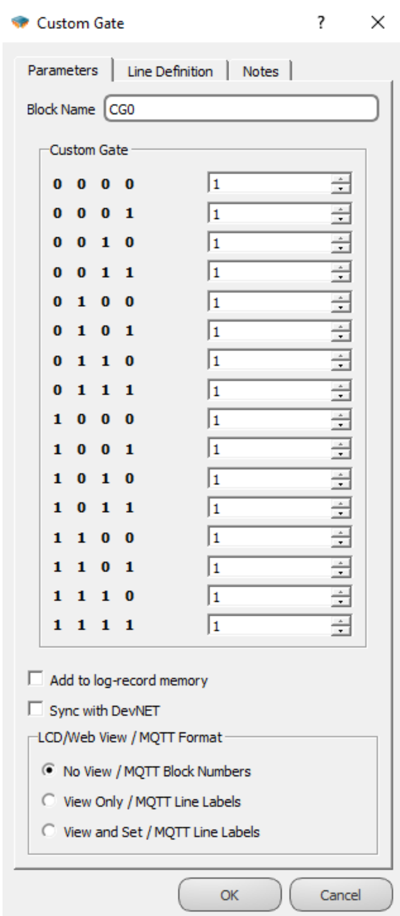
<u>I4: Signal input</u>

It is the 4th input of the Custom Gate block.

<u>#CG0: Output of the block</u>

It is the output of the Custom Gate block. It produces a binary (1-0) value.

1.13.3 Block Settings

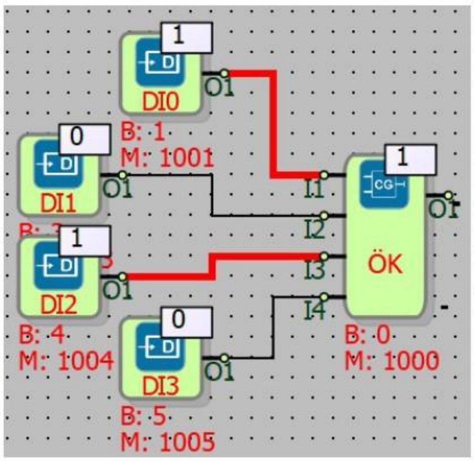
<table>
<tr>
<td>


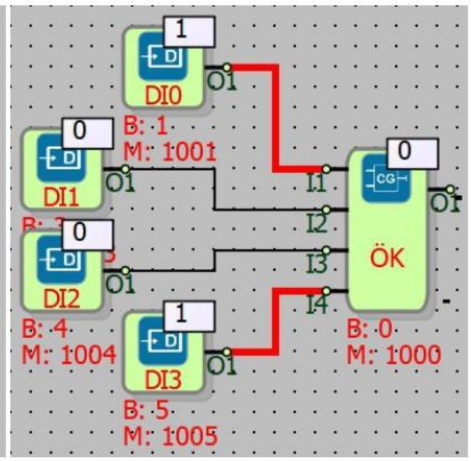
</td>
<td>Custom Gate: It is a special output definition table for 4 block inputs.</td>
</tr>
</table>

1.13.4 Block Explanation

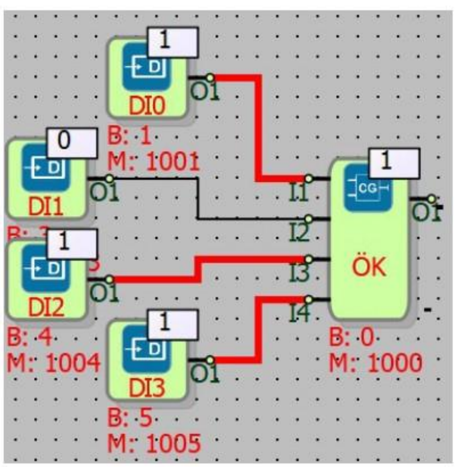
It allows the user to design a desired type of logic gate with the selections made from the block settings. According to the position of the inputs from the block settings, the user can choose which value to be displayed in the output.

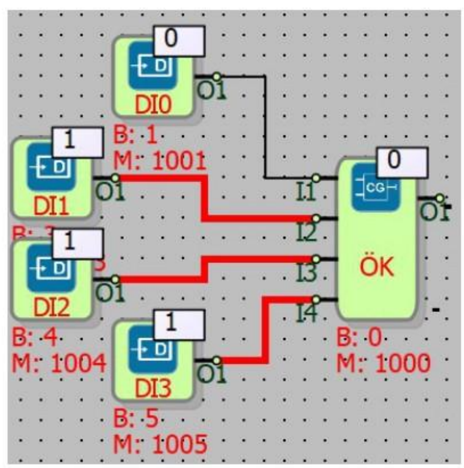
1.13.5 Sample Application





(1) (2)





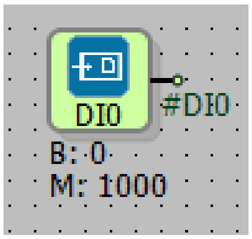
(3) (4)

In the example, in the selections made from the block properties, if the “I1” and “I3” block inputs are high (1) at the same time, the block output will be high (1), in all other cases the block output will be low. In the design, whether the “I2” and “I4” block inputs are high (1) or low (0) has no effect on the state of the output signal.

2 INPUT-OUTPUT BLOCKS

2.1 DIGITAL INPUT BLOCK

2.1.1 Connections

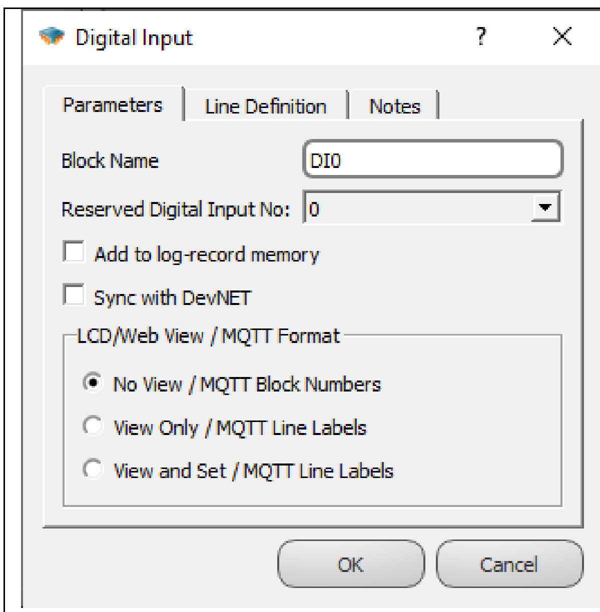
	#DI0: Output of the block

2.1.2 Connection Explanations

<u>#DI0: Output of the block</u>

Output of the block which represents the digital input

2.1.3 Block Settings

	Reserved Digital Input No: Digital input index no can be assigned in Block Settings.

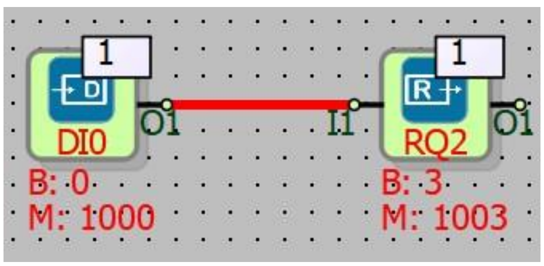
2.1.4 Block Explanation

It is used to read the physical digital input on the device. Used for the inputs on the main unit and expansion units.

Digital Input block is an input which takes binary (0,1) values. Some examples are optical sensors and switches.

Available inputs are listed while selecting the digital input in the block settings menu. Inputs which are used before will not be listed on the list, so there is only one block to add to the project for a physical input on the device. If the digital input will be used in multiple blocks' inputs, related digital input block's output can be labeled and can be used in related blocks

2.1.5 Sample Applications

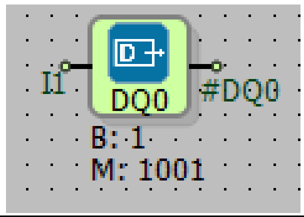


When the Digital Input block (DI0) is set 1, the block output is set to 1 too. The Relay Output block (RQ1) connected to the block output is also set to 1.

(In the example, the digital input is a button, and the relay output may also be used to operate a motor.)

2.2 DIGITAL OUTPUT BLOCK

2.2.1 Connections

I1: I nput of the block		#DQ0: Output of the block

2.2.2 Connection Explanations

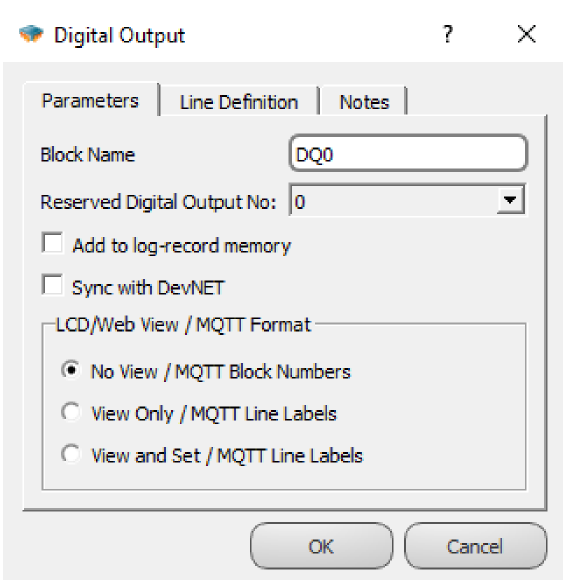
<u>I1: Input of the block</u>

Input of the block for the value which will be written to the digital output

<u>#DQ0: Output of the block</u>

Output of the block which represents the value of the digital output.

2.2.3 Block Settings:

	Reserved Digital Output Number: Digital output number can be assigned in Block Settings.

2.2.4 Block Explanation

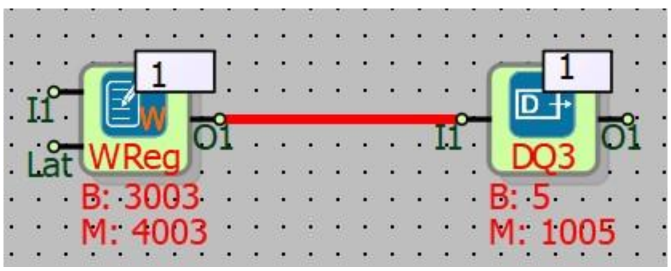
It is used to write values to the physical digital outputs on the device. Used for the outputs on the main unit and the expansion units.

Digital Output block is an output which takes binary (0,1) values.

Available outputs are listed while selecting the digital output in block settings menu. Outputs which are used before will not be listed, so there is only one output block to add to the project for a physical output on the device.

Some digital outputs may be used with PWM /PTO blocks. After the PWM/PTO blocks are activated, related physical digital outputs will be controlled by PWM/PTO blocks. When the PWM/PTO blocks are deactivated, physical digital outputs will be controlled by the Digital Output block on the project.

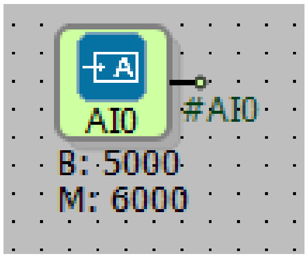
2.2.5 Sample Applications



The Digital Output block's I1 input is connected to the Word Register block's output. When in Word Register block is set to a value other than 0, Digital Output block is set 1 and active. When the Digital Output block receives a value of 1, a device connected to this output can be set on. (engine, lamp, pump etc.)

2.3 ANALOG INPUT BLOCK

2.3.1 Connections

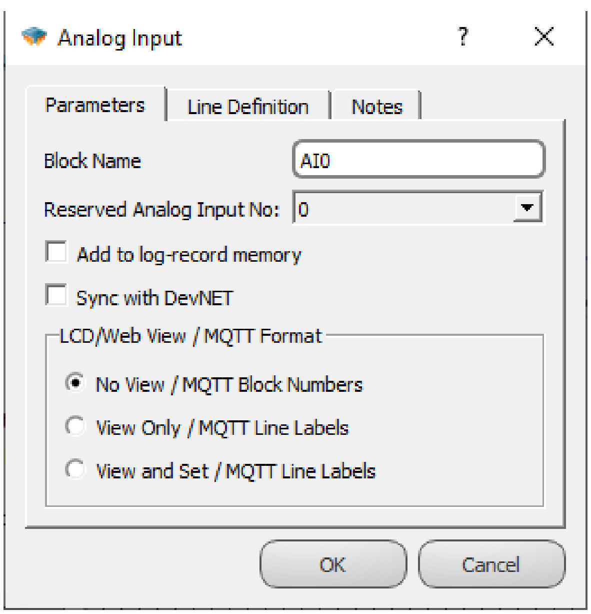
	#AI0: Output of the block

2.3.2 Connection Explanations

<u>#AI0: Output of the block</u>

Output of the block which represents the analog input.

2.3.3 Block Settings

	Reserved Analog Input No: Analog input index can be selected within the block.

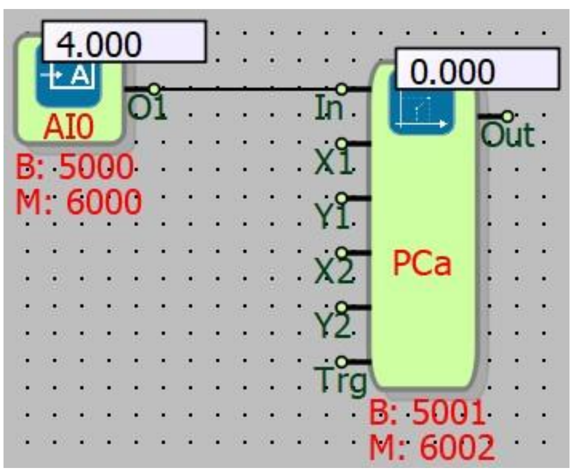
2.3.4 Block Explanation

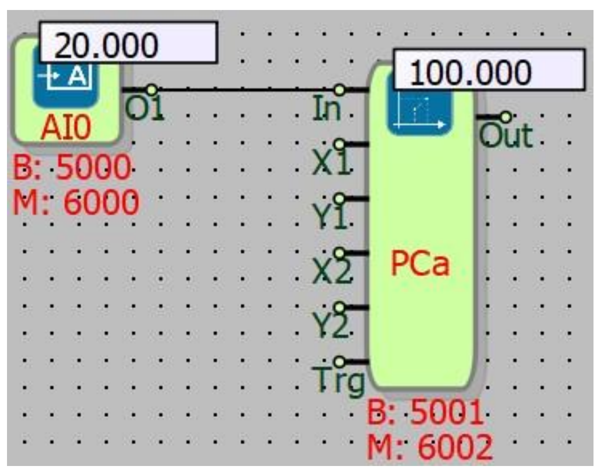
It is used to read physical analog input on the device. Used for the inputs on the main unit and expansion units.

It is determined by the hardware that the block's input is a voltage output or current output. Range of the input is also determined by the hardware.(i.e. 0-10mV, 0, 20mA…) It is assumed that the developer has that information. The value read on the output of the block is floating point value. For example, if 12.48 mA current is applied to the analog input, the output of the block will have the value 12.48.

Available analog inputs are listed while selecting the analog input in block settings menu. Analog inputs which are used before will not be listed, so there is only one Analog Input block to add to the project for a physical analog input on the device. If the analog input will be used in multiple blocks' inputs, related analog input block's output can be labeled and can be used in related blocks.

2.3.5 Sample Application

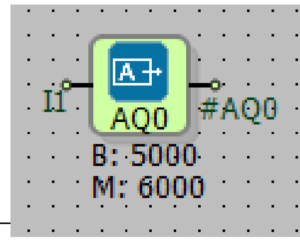




Analog Input0 (AI0) is selected in the example. A device model with physical analog input values between 4-20mA is selected. An analogue sensor is usually connected to the analog input. (level, flow meter, temperature, humidity etc.) The analog input block is connected to the point calibrator block. It is calibrated to zero for 4 mA and 100 for 20 mA.

2.4 ANALOG OUTPUT BLOCK

2.4.1 Connections

I1: Input of the block		#AQ0: Output of the block

2.4.2 Connection Explanations

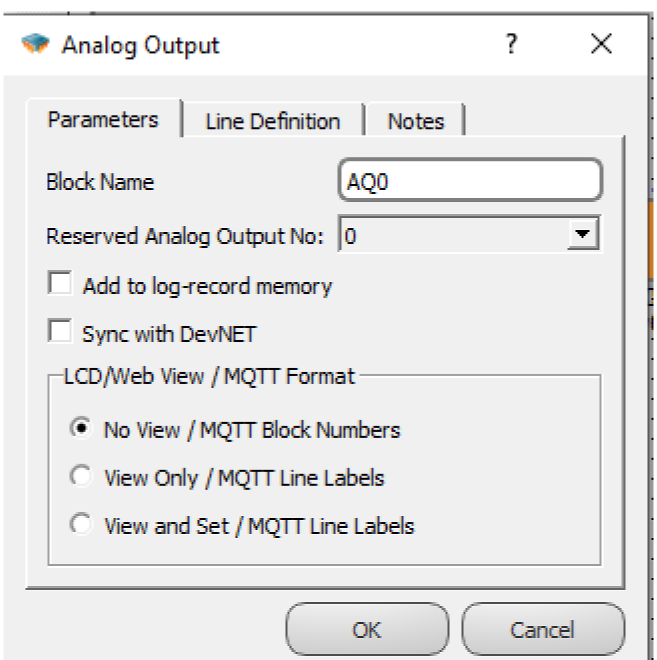
<u>I1: Input of the block</u>

Input of the block for the value that will be written to the analog output.

<u>#AQ0: Output of the block</u>

Output of the block which represents the value of the analog output.

2.4.3 Block Settings

	Reserved Analog Output Number: Analog output number can be assigned in Block Settings.

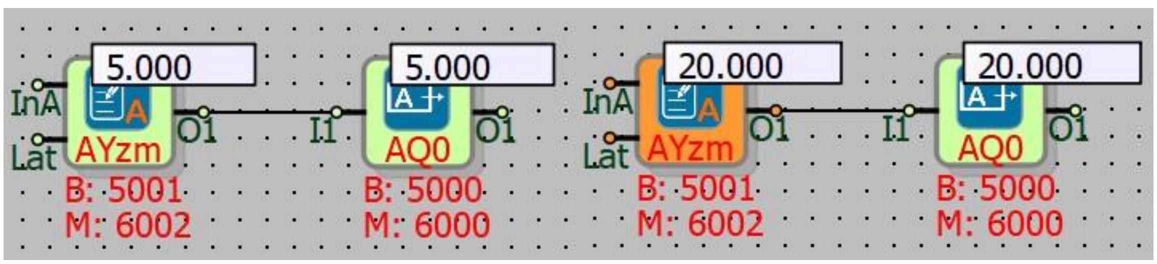
2.4.4 Block Explanation

It is used to write values to the physical analog outputs on the device. Used for the outputs on the main unit and the expansion units.

It is decided by the hardware that the output is a voltage output or current output. Range of the output is also decided by the hardware.(i.e. 0-10mV, 0, 20mA…) It is assumed that the developer has that information. The value read on the output of the block is floating point value. For example, if the desired voltage on the output is 7.56 V, 7.56 should be written on the input of the block.

Available outputs are listed while selecting the analog output in block settings menu. Outputs which are used before will not be listed, so there is only one output block to add to the project for a physical analog output on the device.

2.4.5 Sample Application

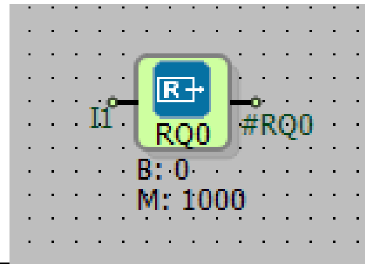


In the example, 5 and 20 values are written to the analog outputs. If the device is 0-20mA compatible output model, the values read at the analog output will be 5mA and 20mA.

Note: When the analog output is less than 0 or greater than 20, it is filtered and a maximum of 20mA at the block output is read at a minimum of 0mA.

2.5 RELAY OUTPUT BLOCK

2.5.1 Connections

I1: Input of the block		#RQ0: Output of the block

2.5.2 Connection Explanations

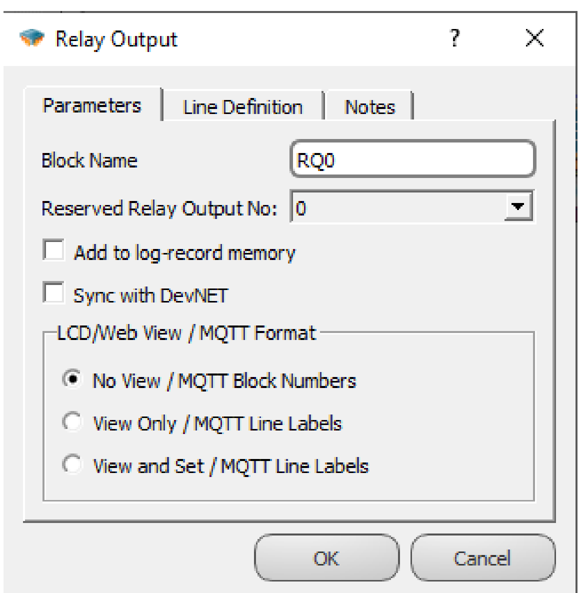
I1: Input of the block

Input of the block for the value which will be written to the relay output.

#RQ0: Output of the block

Output of the block which represents the value of the relay output.

2.5.3 Block Settings

	Reserved Relay Output Number: Relay output index number can be assigned in Block Settings.

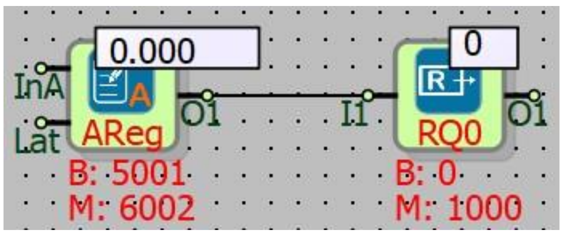
2.5.4 Block Explanation

It is used to write values to the physical relay outputs on the device. Used for the outputs on the main unit and expansion units.

Relay Output block is an output which takes binary values.(0,1).

Available relay outputs are listed while selecting the relay output in block settings menu. Relay outputs, which are used before, will not be listed. So there is only one Relay Output block to add to the project for a physical relay output on the device.

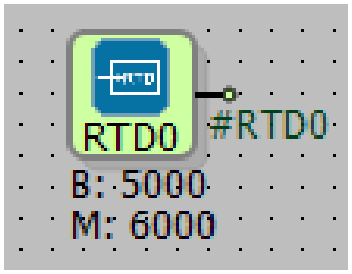
2.5.5 Sample Application



In the example, Relay Output 0 (RQ0) is selected. When a signal other than 0 is received of the I1 input of the Relay Output block, the relay output is set to 1. (In all values different than 0; -1, 0.001, 10, etc.) The relay coil is energized and the relay open contact is closed.

2.6 RTD INPUT BLOCK

2.6.1 Connections

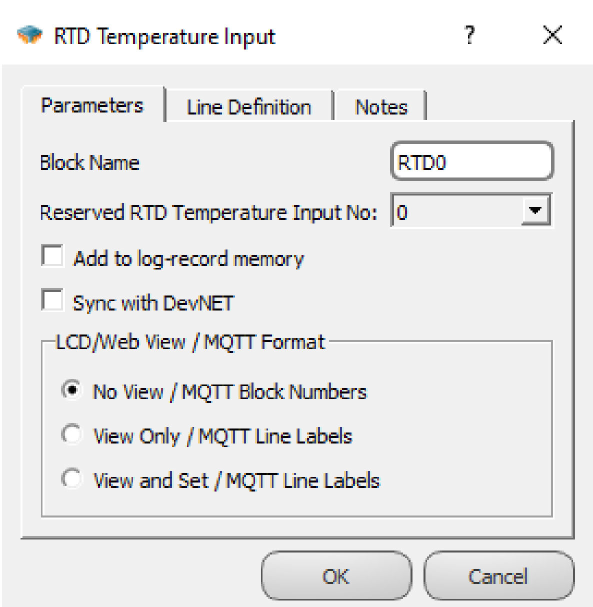
	#RTD0: Output of the block

2.6.2 Connection Explanations

#RTD0: Output of the block

Output of the block, which represents the RTD input.

2.6.3 Block Settings

	Reserved RTD Input Number: RTD input number can be assigned in Block Settings.

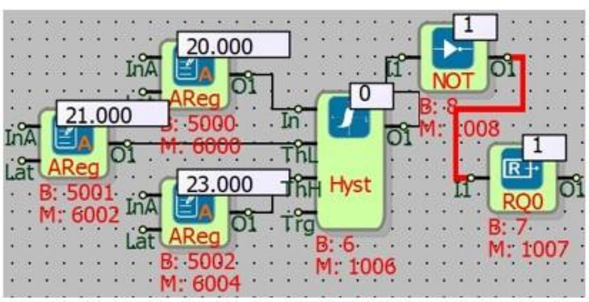
2.6.4 Block Explanation

It is used to read physical RTD inputs on the device. It is used for the inputs on the main unit and expansion units.

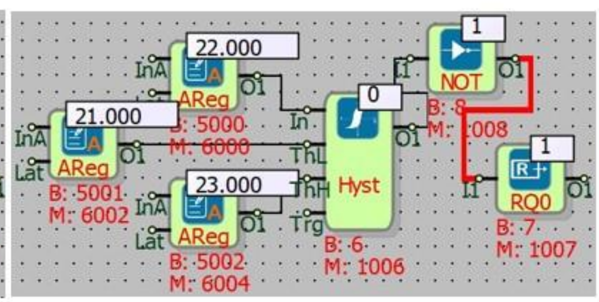
RTD Input block corresponds to one of the resistance thermometers, PT100, PT1000 or NTC. Type of the thermometer is determined by the hardware and it is assumed that the developer has the required information. The value read at the output of the block is a floating-point value. Block gives the corresponding temperature value of the resistance value read from the RTD Input block in Celsius. Integrated conversion tables for PT100 and PT1000 are provided.

Available RTD inputs are listed while selecting the RTD input in block settings menu. RTD inputs, which are used before, will not be listed. So there is only one RTD Input block to add to the project for a physical RTD input on the device. If the RTD input will be used in multiple blocks' inputs, related RTD Input block's output can be labeled and can be used in related blocks.

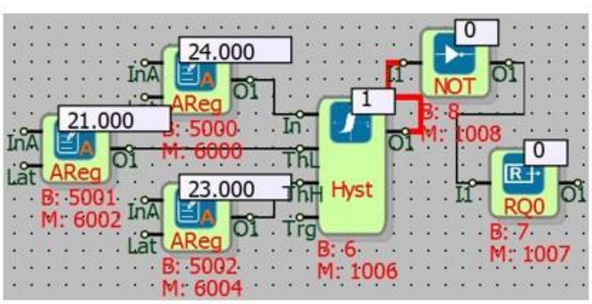
2.6.5 Sample Applications



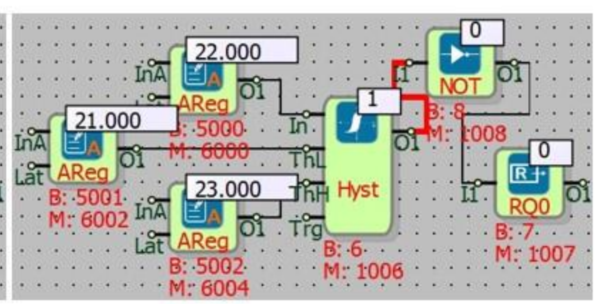
(1)



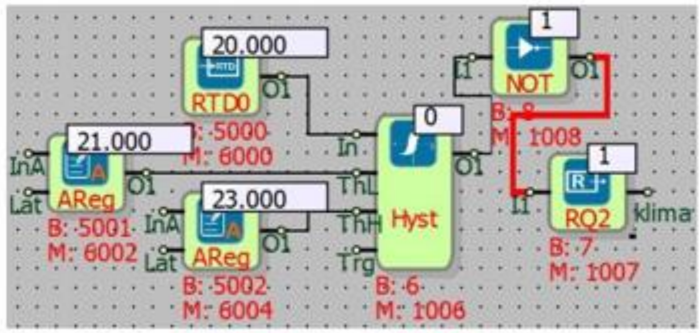
(2)



(3)



(4)



(5)

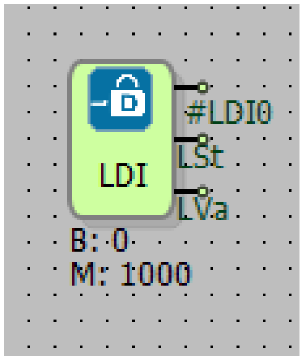
The temperature sensor was connected to the RTD input and the air conditioning heating temperature control was performed with the hysteresis block. With the hysteresis block, the low limit and high limit are selected with as minimum temperature of 21 degrees and as maximum of 23 degrees.

If the RTD Input block’s temperature is lower than 23 degrees at the beginning, the air conditioner that is connected output of relay will be operated. As soon as the RTD Input block’s temperature exceeds 23 degrees, the air conditioner will be switched off. As soon as the RTD Input block’s temperature is lower than 21 degrees, the air conditioner will be restarted. When the RTD Input blok’s temperature exceeds 23 degrees, will be closed again.

Thus, the RTD Input block’s temperature value will be kept constant between 21 and 23 degrees.

2.7 LOCKED DIGITAL INPUT BLOCK

2.7.1 Connections

	#LDI0: Real binary input LSt: Lock state LVa: Lock value

2.7.2 Connection Explanations

#LDI0: Real binary input

Locked digital input block value.

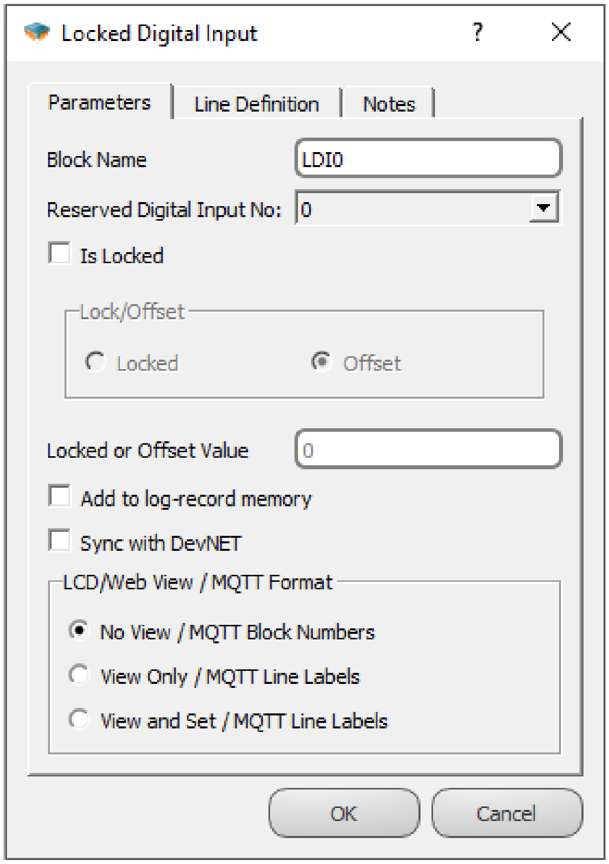
LSt: Lock state

Indicates whether the block is locked or not.

LVa: Lock value

Indicates the value which will be written to the output when locked.

2.7.3 Block Settings

<table>
<tr><td rowspan="3">
</td><td>Reserved Digital Input Number: Digital input index number can be assigned in Block Settings.</td></tr>
<tr><td>Locked: Locking settings of the block. Activates or deactivates locking.</td></tr>
<tr><td>Lock / Offset:
Lock: Writes the value at the lock or shift box to the output.

Offset: It is disabled on digital inputs with locks.</td></tr>
</table>

2.7.4 Block Explanation

Locked Input/Output blocks are used to assign values which are different from the real physical values to the physical input/output blocks. In some situations, expected logic value cannot be retrieved from the field, due to some reasons like an error with a sensor.

In order for the logic project to run properly, until the error is fixed, erroneous value should be ignored and some proper value must be forced onto input. Locked blocks are used to treat situations like this.

#LDI0 (the first output of the block): If the block is locked, the value at the first output of the block is equal to the locked value in the block settings. If the block is not locked, it is equal to the related physical input's value.

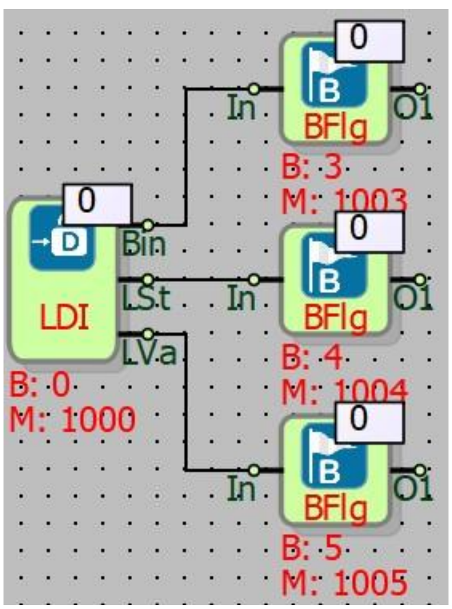
LSt (the second output of the block): Indicates the state of the block. If locking is active, it is high(1), otherwise it is low(0). This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of this value is “block output’s modbus address plus 1”.

LVa (the third output of the block): Indicates the value which will be written to the output when the block is locked. This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of this value is “block output’s Modbus address plus 2”.

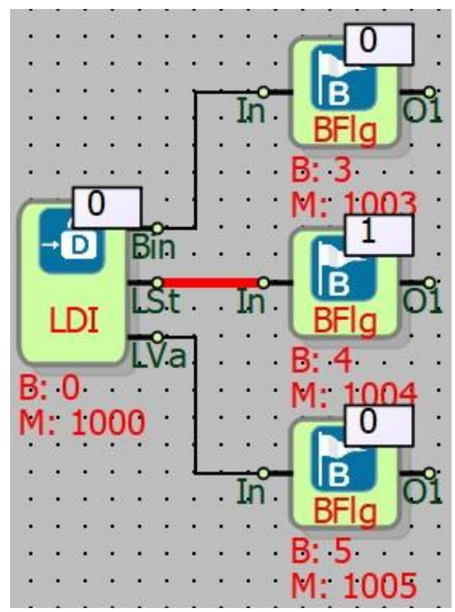
As an example; if block modbus address of the block is 1003, mapping will be the in following way: first output 1003, second output 1004, third output 1005.

2.7.5 Sample Applications

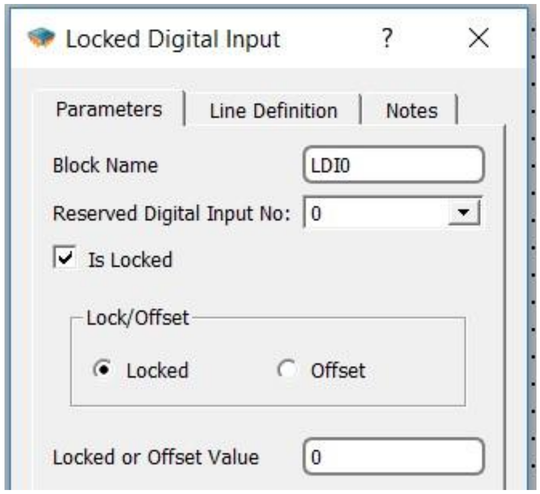
2.7.5.1 Locking to Zero (0)



(1)



(2)



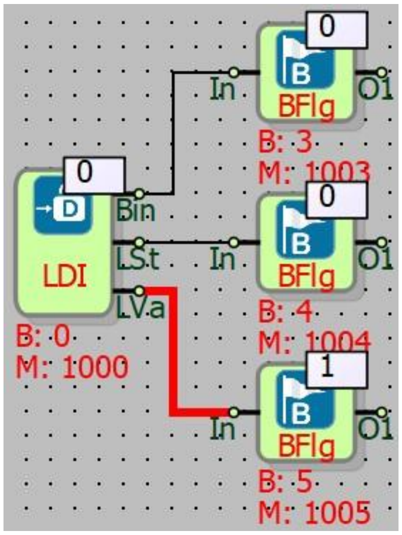
The example is designed to accept a value of 0 in the digital input when the actual signal value to the digital input is not desired. (sensor failure, etc.).

The block "Bin" output has the actual value read from the digital input according to the picture(1). If 0 is present at the “LSt” output, locking is not active.

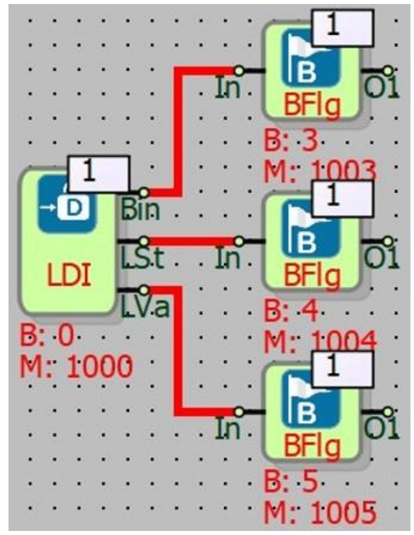
In the picture(2), the value of 1 at the "LSt" output means that the locking is active and that the value at the LVa output is printed on the "Bin" output. If the value at the output of "LVa" had been 1, also the value at the "Bin" output would had been 1.

To make the LSt output 1 or 0, you can write to the 1001st Modbus address. (for this example) In this case, it is possible to write to 1008th Modbus address to make LVa output 1 or 0. (The "Bin" output of the block is the Modbus Address 1000, the LSt output 1001 and the LVa output 1002. Each output of the block corresponds to a Modbus address.)

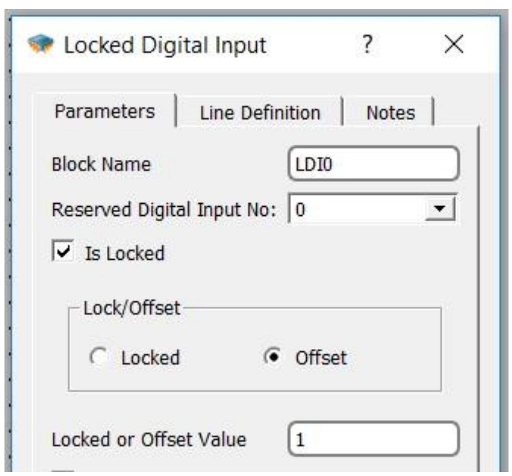
2.7.5.2 Locking to One (1)



(1)



(2)



The example is designed to accept a value of 1 in the digital input when the actual signal value to the digital input is not desired. (sensor failure, etc.).

The block "Bin" output has the actual value read from the digital input according to the picture(1). If 0 is present at the “LSt” output, locking is not active.

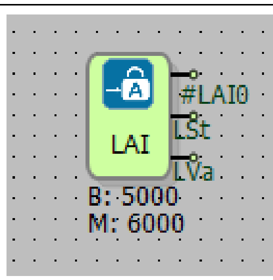
In the picture(2), the value of 1 at the "LSt" output means that the locking is active and that the value at the LVa output is printed on the "Bin" output. If the value at the output of "LVa" had been 1, also the value at the "Bin" output would had been 1.

The "Bin" output is 1 when the output "LSt" is 1. Because the locking value at the output of "LVa" is 1. Therefore, the value at output "LVa" is written on output "Bin" when output "LSt" is 1.

To make the LSt output 1 or 0, you can write to the 1001st Modbus address. (for this example) In this case, it is possible to write to 1008th Modbus address to make LVa output 1 or 0. (The "Bin" output of the block is the Modbus Address 1000, the LSt output 1001 and the LVa output 1002. Each output of the block corresponds to a Modbus address.)

2.8 LOCKED ANALOG INPUT BLOCK

2.8.1 Connections

	#LAI0: Analog input lock value
	LSt: Lock state
	LVa: Lock value

2.8.2 Connection Explanations

#LAI0: Analog input lock value

Locked analog input block value.

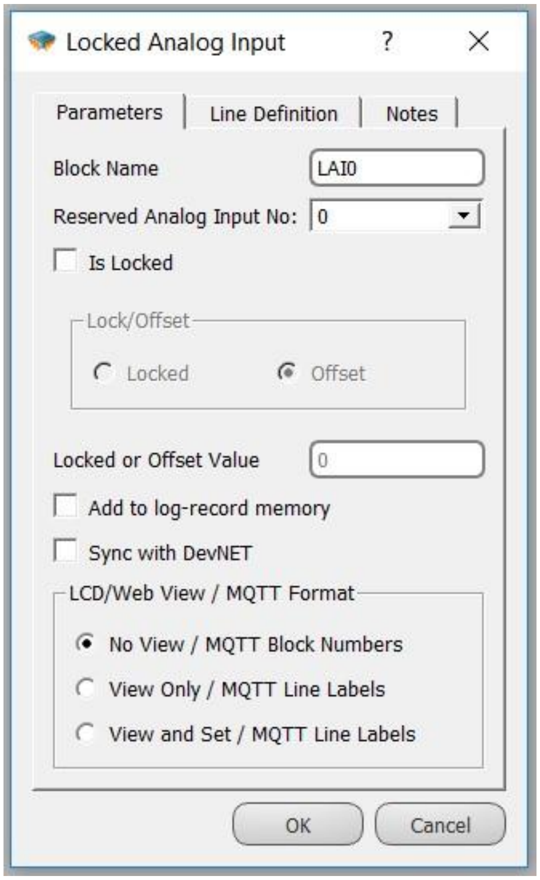
LSt: Lock state

Indicates whether the block is locked or not.

LVa: Lock value

Indicates the value which will be written to the output when locked.

2.8.3 Block Settings



Reserved Analog Input Number: Analog input index number can be assigned in Block Settings.
Locked: Locking settings of the block. Activates or deactivates locking
Lock / Offset: Lock: Write the value at the lock or shift box to the output. Offset: It is not activated on digital inputs with locks.

2.8.4 Block Explanation

Locked Input/Output blocks are used to assign values which are different from the real physical values to the physical input/output blocks. In some situations, expected logic value cannot be retrieved from the field, due to some reasons like an error with a sensor.

In order for the logic project to run properly, until the error is fixed, erroneous value should be ignored and some proper value must be forced onto input. Locked blocks are used to treat situations like this.

#LAI0 (the first input of the block): If the block is locked, the value at the first output of the block is equal to the locked value in the block settings. If the block is not locked, it is equal to the related physical input's value.

LSt (the second input of the block): Indicates the state of the block.

If the block is locked(active) and the specified value will be written to the output it is 1.0;

if the block is locked(active) and an offset value will be added to the real value it is 2.0;

if the block is not locked(passive) it is 0.0.

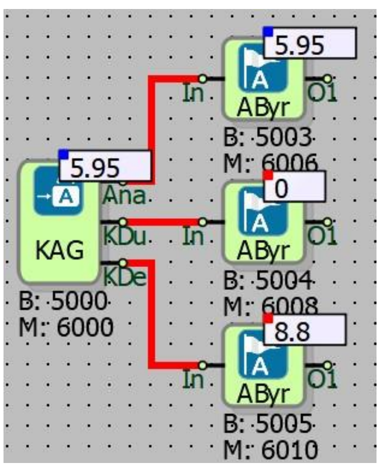
This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of the output is “block output’s modbus address plus 2”

LVa (third output of the block): Indicate the value which will be written or added to the output when the first output is 1 or 2 respectively. This output is mapped into Modbus addresses and can be read or written remotely. The Modbus address of the output is “the block output’s modbus address plus 4”.

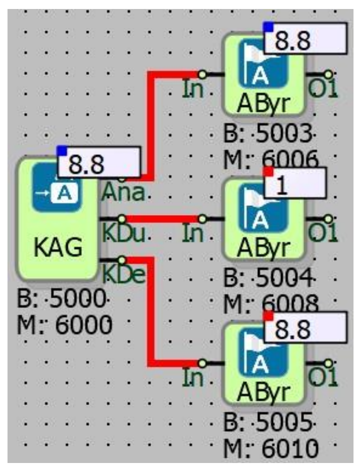
As an example; if block modbus address of the block is 6006, mapping will be the in following way: first output 6006, second output 6008, third output 6010.

2.8.5 Sample Applications

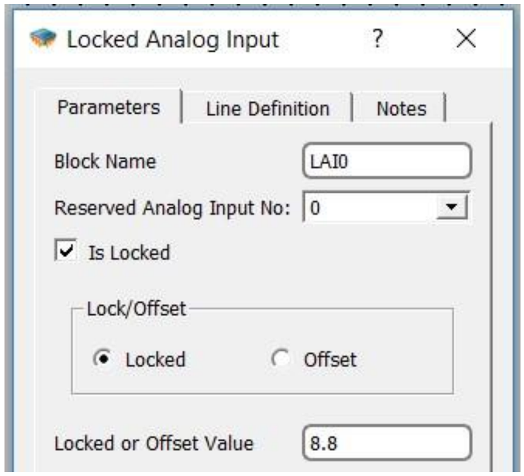
2.8.5.1 Locking a Value



(1)



(2)



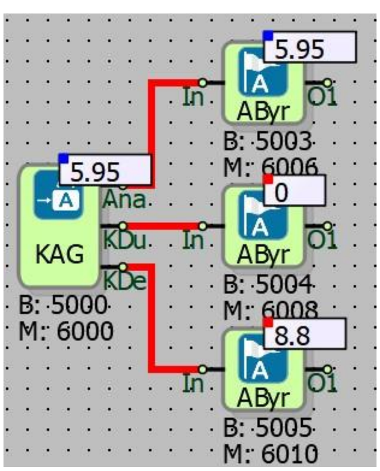
In the example, the actual signal value for the device analog input is 5.95. When the block "LSt" output is 0, the actual value of the analog input is written to the "Ana" output. (Picture1)

In the case where the analog input actual value is not desired to be used, LSt output is set to 1 and the locking value at the LVa output is written to the block "Ana" output. (Picture2)

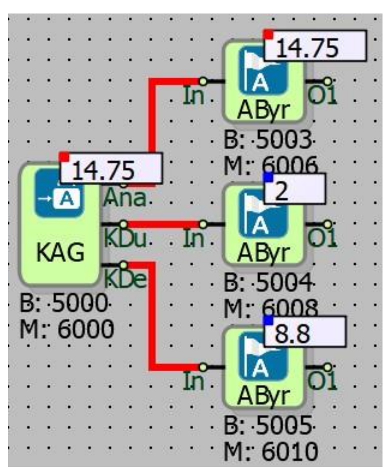
Block "Ana" output Modbus address is 6000. In this case the LSt output is 6002, the LVa output is 6004. When the 6002nd Modbus address is written 1, the value at 6004th Modbus address is written to the "Ana" output. The actual analog input signal value or lockout value of the block output is determined by the LSt output.

If the LSt output is 0, the actual signal value is written to the "Ana" block output, and if LSt output is 1, the Lock value is written to the "Ana" block output.

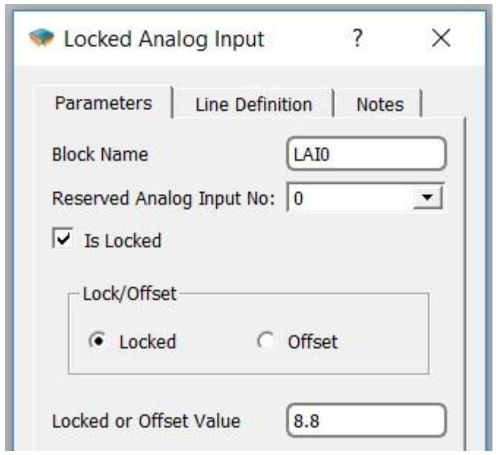
2.8.5.2 Adding Offset Value to Actual Value



(1)



(2)



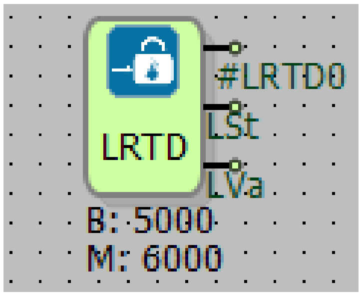
In the example, when the output LSt is 0, the analog input actual signal value is written to the block output. On Figure 2, 2 is written on LSt Output.

To add the value at the LVa output to the actual signal value, a value of 2 is written to the LSt output.

When the LSt output is 2, the value at the LVa output is added to the actual signal value (5.95; for this example), and the total value is written to the block Ana output. (5.95 + 8.8= 14.75)

2.9 LOCKED RTD INPUT BLOCK

2.9.1 Connections

	#LRTD0: RTD input lock value
	LSt: Lock state
	LVa: Lock value

2.9.2 Connection Explanations

#LRTD0: RTD input lock value

Locked RTD block value.

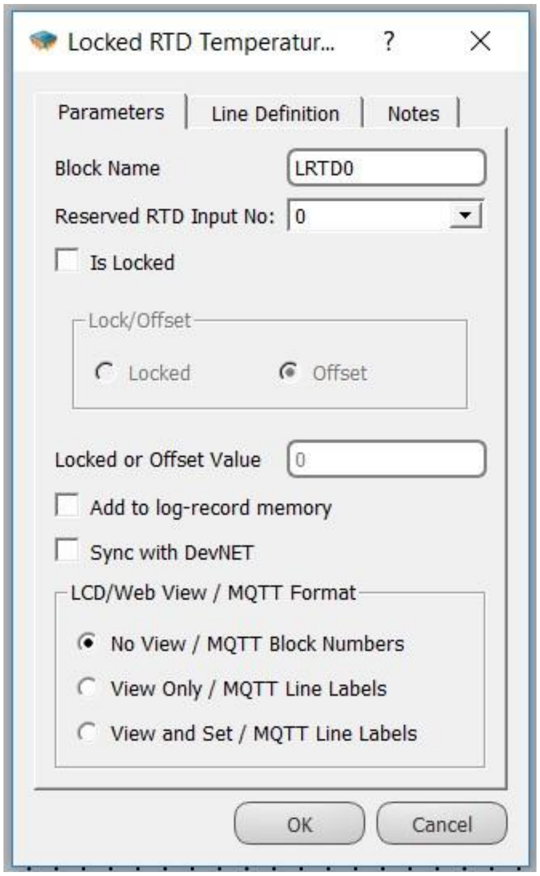
LSt: Lock state

Indicates whether the block is locked or not.

LVa: Lock value

Indicates the value which will be written/added to the output when locked.

2.9.3 Block Settings

	Reserved RTD Input Number: RTD input number can be assigned in Block Settings.
	Locked: Locking settings of the block. Activates or deactivates locking
	Lock / Offset: Lock: Write the value at the lock or shift box to the output. Offset: It is not activated on digital inputs with locks.

2.9.4 Block Explanation

Locked Input/Output blocks are used to assign values which are different from the real physical values to the physical input/output blocks. In some situations, expected logic value cannot be retrieved from the field, due to some reasons like an error with a sensor.

In order for the logic project to run properly, until the error is fixed, erroneous value should be ignored and some proper value must be forced onto input. Locked blocks are used to treat situations like this.

Offset property of the Locked RTD Input blocks, being different from the other locked blocks, is used to correct the cable resistance error between the RTD and the device. For example, a

PT1000 sensor which is 300 meters away from the unit has a cable resistance around 35 Ohms. Offset value should be set to -35.0 to compensate the extra resistance caused by the cable.

#LRTD0 (the first input of the block): If the block is locked, the value at the first output of the block is equal to the locked value in the block settings or it is equal to the sum of the real value and the offset value. If the block is not locked, it is equal to the related physical input's value.

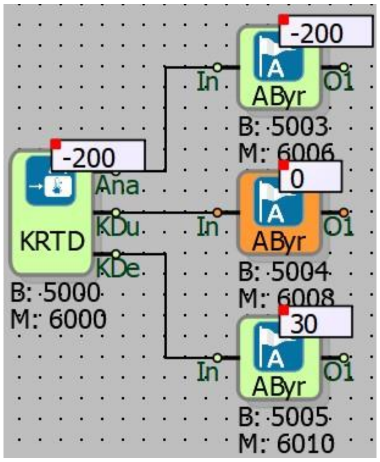
LSt (the second input of the block): Indicates the state of the block.
If the block is locked(active) and the specified value will be written to the output it is 1;
if the block is locked(active) and an offset value will be added to the real value it is 2;
if the block is not locked(passive) it is 0.
This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of the output is "block output's modbus address plus 2".

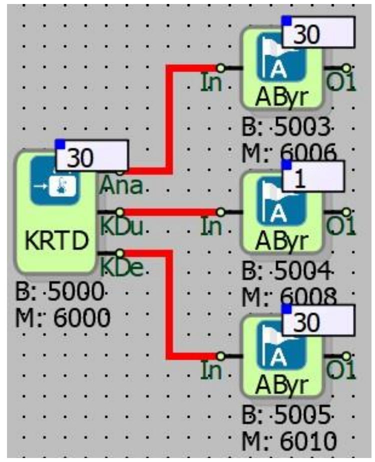
LVa (third output of the block): Indicates the value which will be written or added to the output when when the first output is 1 or 2 respectively. This output is mapped into Modbus addresses and can be read or written remotely. The Modbus address of the output is "block output's Modbus address plus 4".

As an example; if block modbus address of the block is 6012 : first output 6012, second output 6014, third output 6016.

2.9.5 Sample Applications

2.9.5.1 Locking a Value





In the example, the device shows the actual signal value -200 from the RTD temperature input. This value means that the sensor is not connected to the RTD input or there is a problem with the connected sensor or cable line. Picture(1)

In Picture 2, there is a problem with the sensor at RTD input. In this case, a value has been locked.

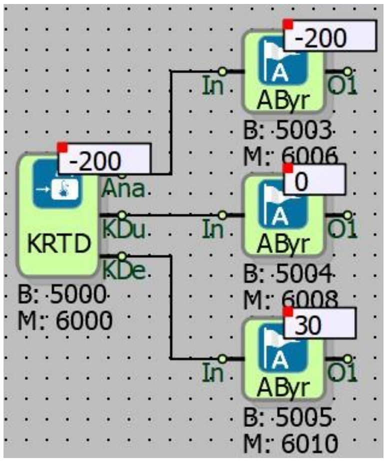
In the locked RTD temperature input block, the actual value at the block RTD input is set to the "Ana" output block while the LSt output is 0.

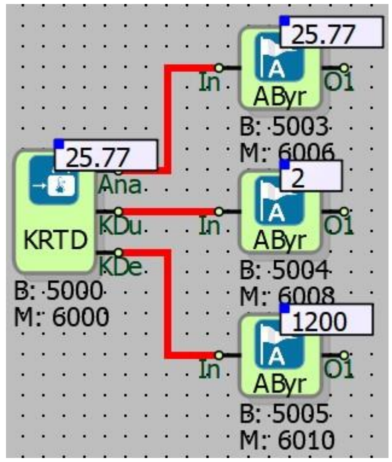
When there is a problem with the RTD input, 1 is written to the LSt output and the locking value at the LVa output is written in degrees Celsius to the "Ana" output.

In this example, the Modbus Address of the Locked RTD temperature input block is 6000 and can only be read.

In this case, the corresponding values can be written to the LSt output from the 6002nd Modbus address and the LVa output from the 6004th Modbus address.

2.9.5.2 Adding Offset Value to Actual Value





(1) (2)

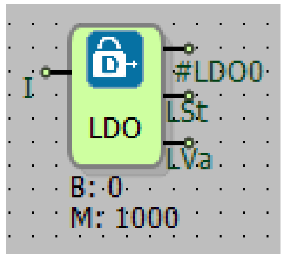
In this example, there is a problem with the sensor or sensor connection in Figure 1.

In Figure 2, it is written to LSt output 2 and the offset is added. The process of adding offset is in Ohms. The added 1200 Ohm value corresponds to 25.77 °C as the temperature.

Note: If LSt output is 1, the value at output LVa is in °Cs. If LSt output is 2, the value at LVa output is in Ohms.

2.10 LOCKED DIGITAL OUTPUT BLOCK

2.10.1 Connections

<table>
<tr><td rowspan="3">I1: Block input</td><td rowspan="3">
</td><td>#LDO0: Real binary output</td></tr>
<tr><td>LSt: Lock state</td></tr>
<tr><td>LVa: Lock value</td></tr>
</table>

2.10.2 Connection Explanations

<u>I1: Block input</u>

Indicates the value which will be written to the output when unlocked.

<u>#LDO0: Real binary output</u>

Locked digital output block value.

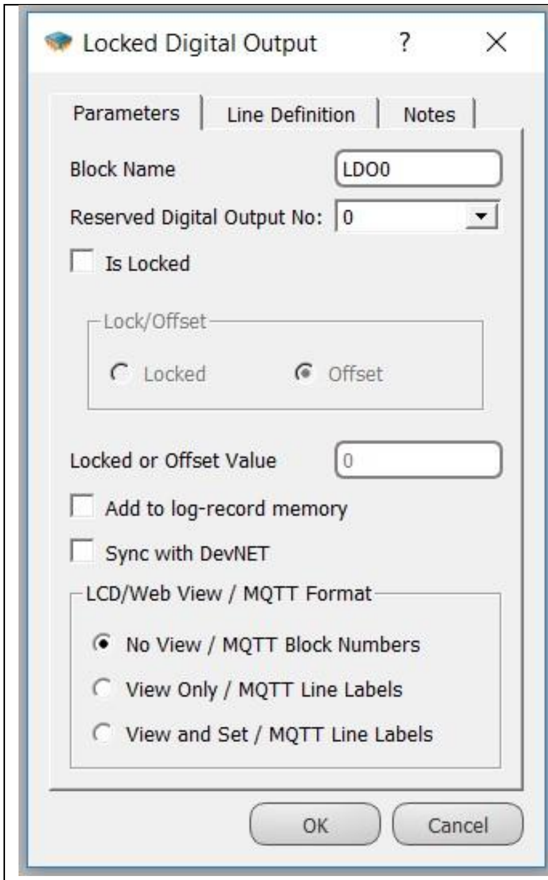
<u>LSt: Lock state</u>

Indicates whether the block is locked or not.

<u>LVa: Lock value</u>

Indicates the value which will be written to the output when locked.

2.10.3 Block Settings

	Reserved Digital Output Number: Digital output number can be assigned in Block Settings.
	Locked: Locking settings of the block. Activates or deactivates locking
	Lock / Offset: Lock: Write the value at the lock or shift box to the output. Offset: It is not activated on digital inputs with locks.

2.10.4 Block Explanation

Locked Input/Output blocks are used to assign values which are different from the real physical values to the physical input/output blocks. In some situations, expected logic value cannot be retrieved from the field, due to some reasons like an error with a sensor.

In order for the logic project to run properly, until the error is fixed, erroneous value should be ignored and some proper value must be forced onto input. Locked blocks are used to treat situations like this.

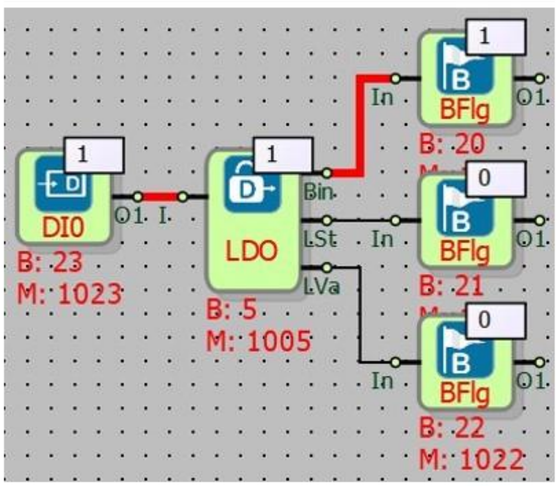
#LDO0 (the first output of the block): If the block is locked, the value at the first output of the block is equal to the locked value in the block settings. If the block is not locked, it is equal to the related physical input's value.

LSt (the second output of the block): Indicates the state of the block. It is 1 if the block is locked(active), 0 otherwise(passive). This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of the output is “the block output’s modbus address plus 1”.

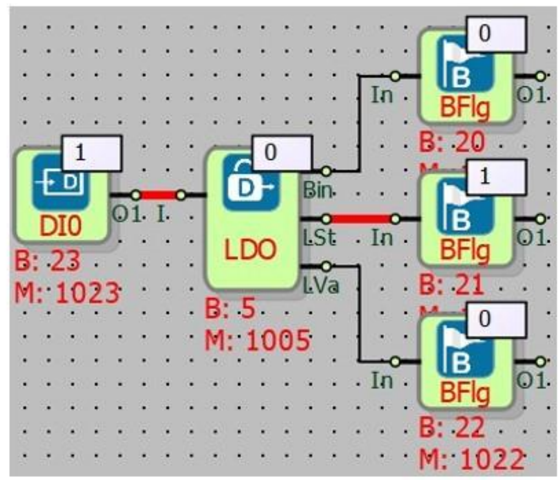
LVa (the third output of the block): Indicates the value, which will be written to the output when the block is locked. This output is mapped into Modbus addresses and can be read and written remotely. The Modbus of the value is “the block output’s modbus address plus 2”.

As an example, if block modbus address of the block is 1006: first output 1006, second output 1007, third output 1008.

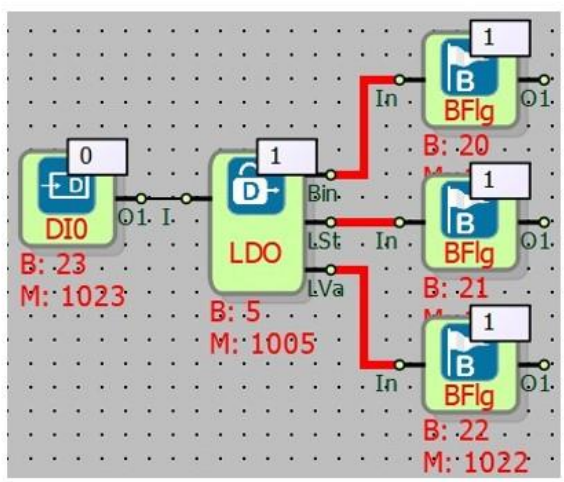
2.10.5 Sample Applications



(1)



(2)



(3)

In this project, the output of the Digital Input (DI0) block is connected to the Locked Digital Output block input.

Locking is not active in LDO block; The LDO block 'Bin' output will be 1 when the DI0 block is 1. The LDO block 'Bin' output will be 0 when the DI0 block is 0.

Locking not active: Locking is not active because LSt output is 0 in figure1. Therefore the value of the Digital Input (DI0) is transferred to the 'Bin' output of the LDO block.

Locking active: In figure(2), Locking is active because LSt output is shown as 1.

Therefore the value 0 at the LVa output is written to the block 'Bin' output.

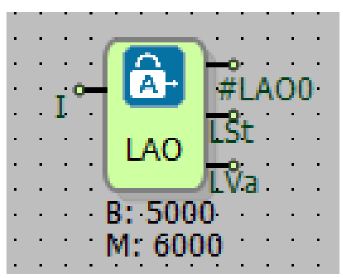
Locking active: In figure(3), Locking is active because LSt output is shown as 1.

Therefore the value 1 at the LVa output is written to the LDO block 'Bin' output.

Modbus addresses can be used to change the values of the LSt and LVa outputs of the LDO block. The Modbus address of the LDO block's Bin output is 1005. (for this example) In this case, the Modbus address of the LSt output is 1006, the Modbus address of the LVa output is 1007.

2.11 LOCKED ANALOG OUTPUT BLOCK

2.11.1 Connections

I1: Block input		#LAO0: Analog lock output LSt: Lock state LVa: Lock value

2.11.2 Connection Explanations

I: Block input

Indicates the value which will be written to the output when unlocked.

#LAO0: Analog lock output

Locked analog output block value.

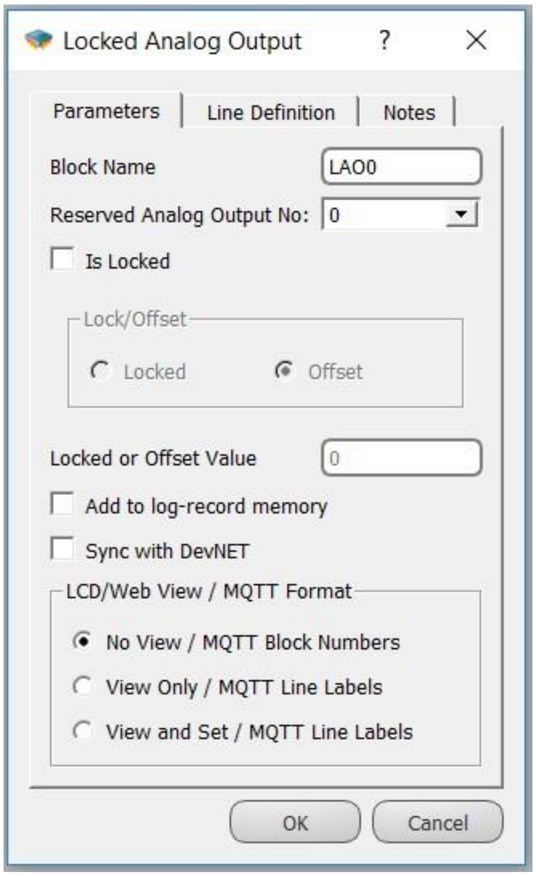
<u>LSt: Lock state</u>

Indicates whether the block is locked or not.

<u>LVa: Lock value</u>

Indicates the value which will be written to the output when locked.

2.11.3 Block Settings



Reserved Analog Output Number: Analog output number can be assigned in Block Settings.
Locked: Locking settings of the block. Activates or deactivates locking
Lock / Offset: Lock: Write the value at the Lock or Shift box to the output. Offset: It is not activated on digital inputs with locks.

2.11.4 Block Explanation

Locked Input/Output blocks are used to assign values, which are different from the real physical values to the physical input/output blocks. In some situations, expected logic value cannot be retrieved from the field, due to some reasons like an error with a sensor.

In order for the logic project to run properly, until the error is fixed, erroneous value should be ignored and some proper value must be forced onto input. Locked blocks are used to treat situations like this.

#LAO0 (the first output of the block): If the block is locked, the value at the first output of the block is equal to the locked value in the block settings. If the block is not locked, it is equal to the related physical input's value.

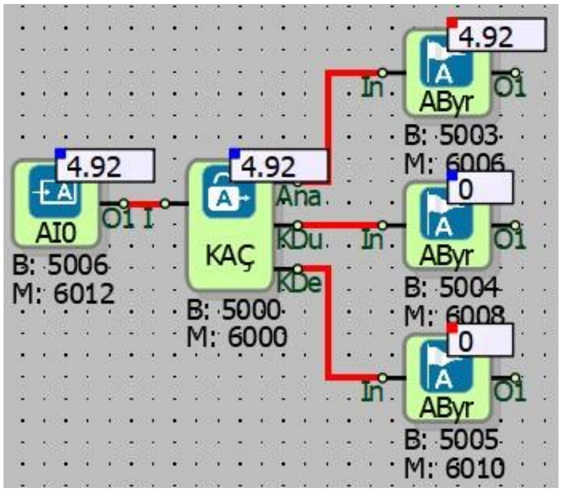
LSt (the second output of the block): Indicates the state of the block. It is 1 if the block is locked(active), 0 otherwise(passive). This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of the output is of the "block output's Modbus address plus 2".

LVa (the third output of the block): Indicates the value, which will be written to the output when the block is locked. This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of the output is "the block output's Modbus address plus 4".

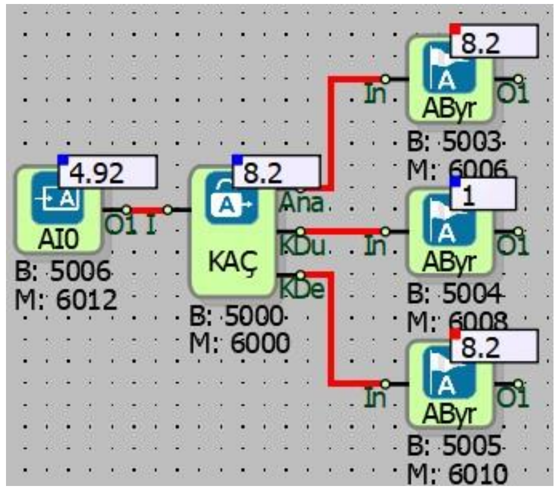
As an example; if modbus address of the block is 6018: first output 6018, second output 6020, third output 6022.

2.11.5 Sample Applications

2.11.5.1 Locking a Value



(1)



(2)

In the sample project, the output of the Analog Input block is connected to the Locked Analog Output (LOA) block input.

When locking is not active in LOA block; The LOA block “Ana” output will be 4.92 when the AI0 block is 4.92.

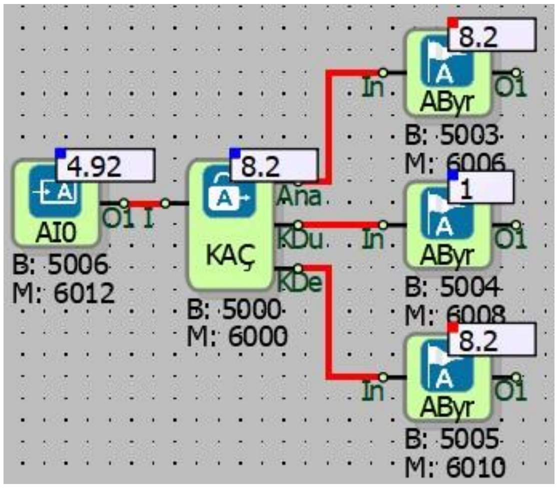
The value of the signal on the AI0 block and “Ana” output of the LOA block will always be the same when locking and offset scrolling is inactive.

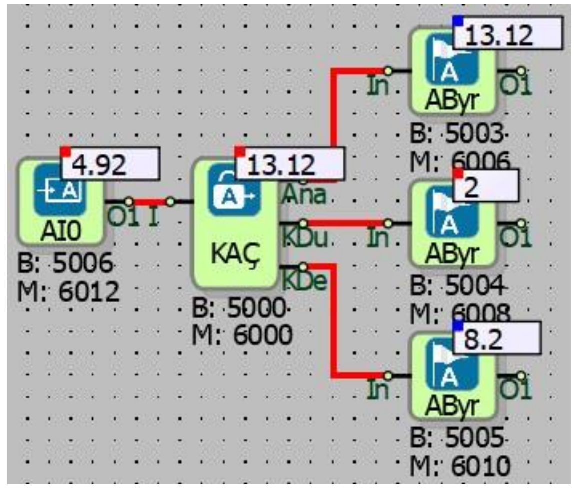
Locking not active: Locking is not active because LSt output is 0 in figure1. The value of the analog input (AI0) is written to the “Ana” output of the LOA block.

Locking active: Locking is active because LSt output is shown as 1 in figure2. '8.2' value at the LVa output is written to the LOA block “Ana” output.

Modbus addresses can be used to change the values of the LSt and LVa outputs of the LOA block. The Modbus address of the “Ana” output of the LOA block is 6000. (for this example) In this case the Modbus address of the LSt output is 6002, the Modbus address of the LVa output is 6004.

2.11.5.2 Adding Offset Value to Actual Value





(1) (2)

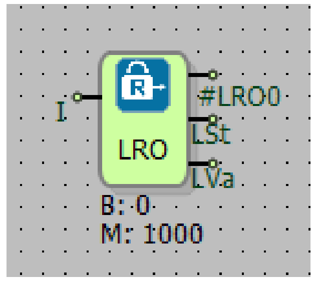
Locking active: Locking is active because LSt output is 1 in Figure1 and 8.2 value on LVa output is written to the "Ana" output of the block.

Shift active: The shift is active in Figure 2 because LSt output is 2, the value of 8.2 in the LVa output and the value of AI0 signal in the LOA block input have been collected and written to the block "Ana" output. (8.2 + 4.92 = 13.12)

Modbus addresses can be used to change the values of the LSt and LVa outputs of the LOA block. The Modbus address of the “Ana” output of the LOA block is 6000. (for this example) In this case the Modbus address of the LSt output is 6002, the Modbus address of the LVa output is 6004.

2.12 LOCKED RELAY OUTPUT BLOCK

2.12.1 Connections

I1: Block input		#LRO0: Relay lock output LSt: Lock state LVa: Lock value

2.12.2 Connection Explanations

I1: Block input

Indicates the value which will be written to the output when unlocked.

#LRO0: Relay lock output

Locked relay output block value.

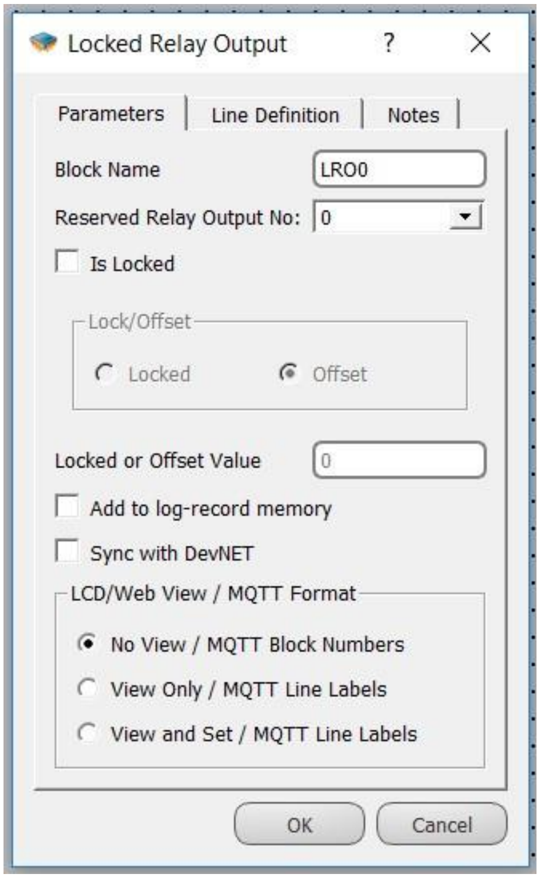
LSt: Lock state

Indicates whether the block is locked or not.

LVa: Lock value

Indicates the value which will be written to the output when locked.

2.12.3 Block Settings

	Reserved Relay Output Number: Relay output index number can be selected here
	Locked: Locking settings of the block. Activates or deactivates locking
	Lock / Offset: Lock: Write the value at the lock or shift box to the output. Offset: It is not activated on digital inputs with locks.

2.12.4 Block Explanation

Locked Input/Output blocks are used to assign values which are different from the real physical values to the physical input/output blocks. In some situations, expected logic value cannot be retrieved from the field, due to some reasons like an error with a sensor.

In order for the logic project to run properly, until the error is fixed, erroneous value should be ignored and some proper value must be forced onto output. Locked blocks are used to treat situations like this.

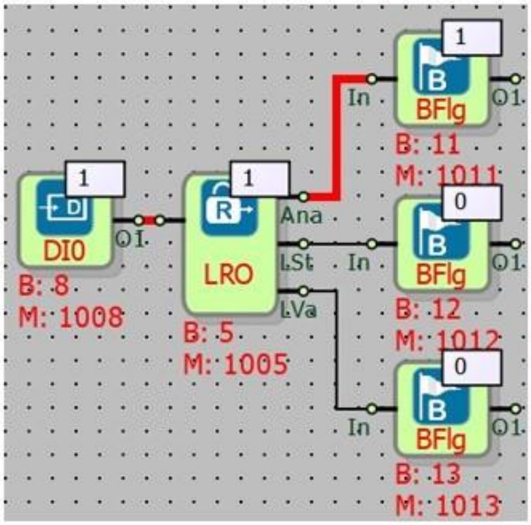
#LRO0 (the first output of the block): If the block is locked, the value at the first output of the block is equal to the locked value in the block settings. If the block is not locked, it is equal to the related physical input's value.

LSt (the second output of the block): Indicates the state of the block. It is 1 if the block is locked(active), 0 otherwise(passive). This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of the output is “the block output’s Modbus address plus 1”.

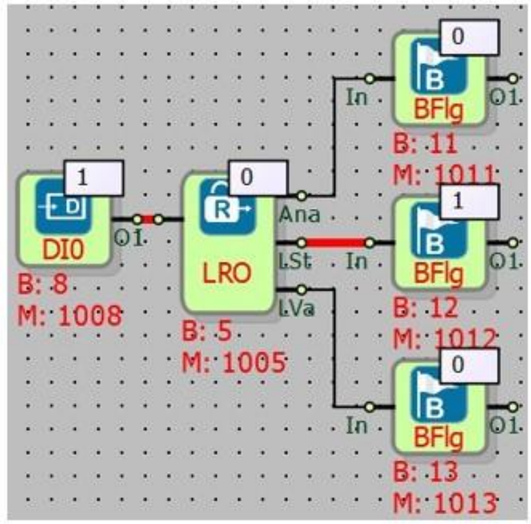
LVa (the third output of the block): Indicates the value which will be written to the output when the block is locked. This output is mapped into Modbus addresses and can be read and written remotely. The Modbus address of the output is “the block output’s Modbus address plus 2”.

As an example; if modbus address of the block is 1006: first output 1006, second output 1007, third output 1008.

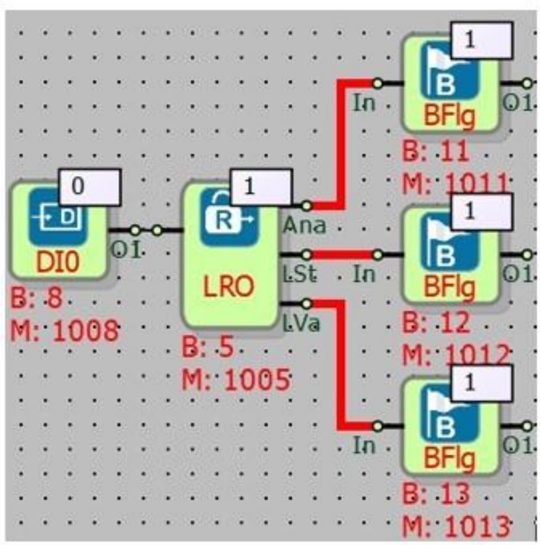
2.12.5 Sample Applications



(1)



(2)



(3)

In the Mikrodiagram example, the output of the Digital Input block is connected to the Locked Relay input (LRO).

When Locking is not active in the LRO block; The LRO block "Ana" output will also be 1 when the DI0 block is 1. When the block DI0 is 0, the LRO block "Ana" output will be 0.

Locking not active: Since LSt output is 0 in Figure1, the value of the Digital Input is written to the "Ana" output of the Locked Relay Output block.

Locking active: Locking is active because LSt output 1 is shown in figure2. 0 at the LVa output is written to the block output.

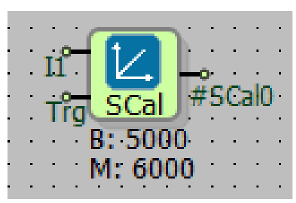
Locking active: Locking is active because LSt output 1 is shown in figure3. 1 at the LVa output is written to the block output.

Modbus addresses can be used to change the values of the LSt and LVa outputs of the LRO block. The Modbus address of the “Ana” output of the LRO block is 1005. (for this example) In this case the Modbus address of the LSt output is 1006, the Modbus address of the LVa output is 1007.

3 CALIBRATION BLOCKS

3.1 SLOPE CALIBRATOR

3.1.1 Connections

I1: Signal input		#SCal0: Block output
Trg: Trigger input		

3.1.2 Connections Applications

I1: Signal input
The input of the slope value to be used.

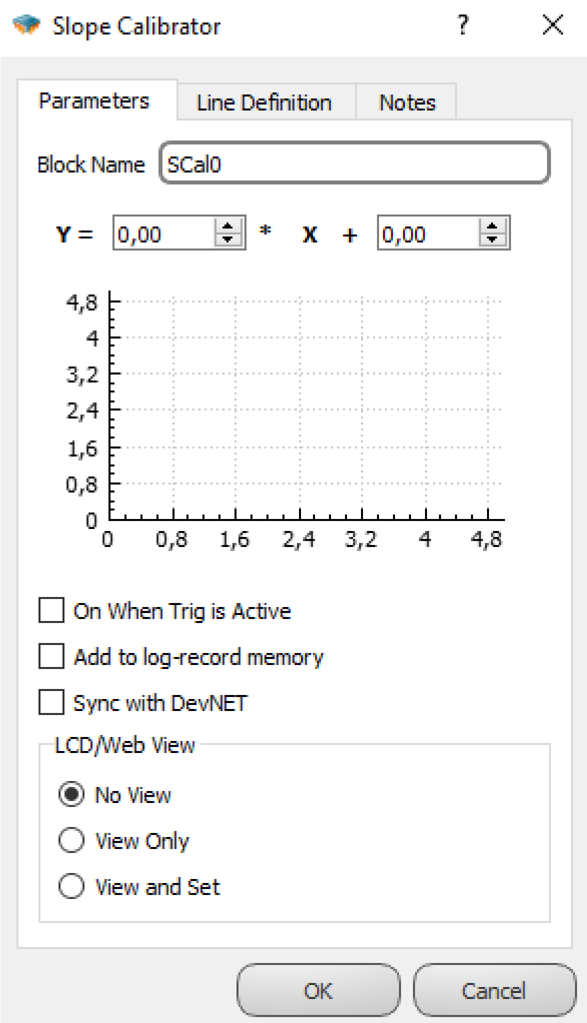
Trg: Trigger input

Trigger input can be left blank.

#SCal0: Block output

It is the output of the calibrated slope input.

3.1.3 Block Settings



Y: Q1 is the calibrated block output value.

X: I1 is the uncalibrated block input value.

m: The value of m in the equation Y = mX + c is the non-calibrated I1 input multiplier coefficient.

c: The value "c" in the equation Y = mX + c is the uncorrelated total coefficient for I1 input.

Trg: Trig Active Work

Not selected; It calibrates the input value and transfers it to the output in each PLC program cycle.

When selected; whenever the rising edge comes to the input of "Trg", it calibrates the input value and transfers it to the output.

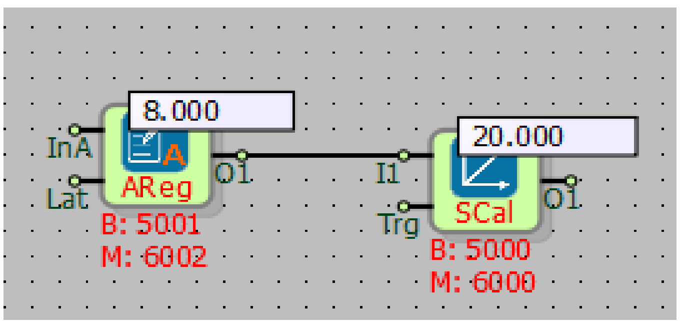
3.1.4 Block Explanation

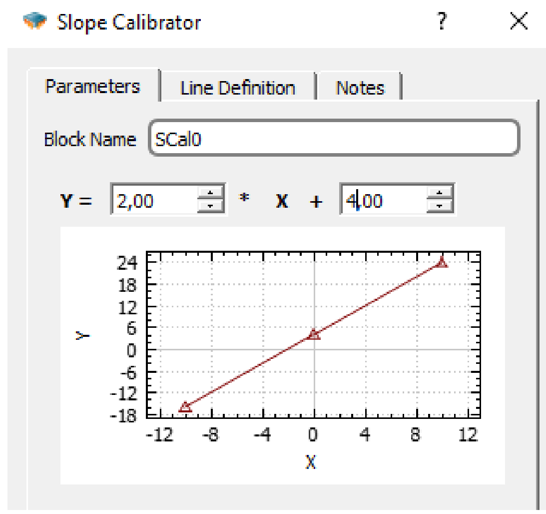
The slope calibrator block means that an analogue value is processed as "Y = mX + c".

The m and c values are coefficient values set from the block options.

The “X“ value is the input (I1) of the block and the value of Y is the output (Q1) of the operation.

3.1.5 Sample Applications



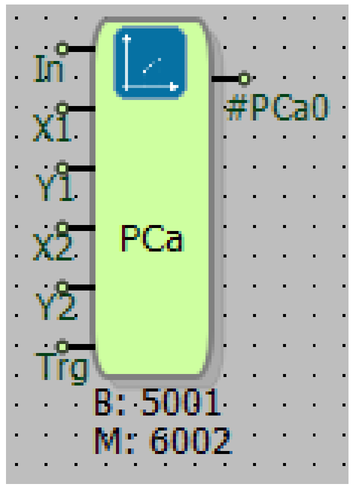


This is an example of connection of slope calibrator. In the example, m is set to 2, c is set to 4. When the coefficients are replaced in the Y=mX+c line equation, the equation is formed as Y=2X+4.

The Y value (block output (Q1)) is obtained according to the X value (input I1) defined at the Slope Calibrator block input. (Y=8*2+4=20)

3.2 POINT CALIBRATOR

3.2.1 Connections

In: Signal input X1: Signal input low limit Y1: Signal output low limit X2: Signal input upper limit Y2: Signal output upper limit Trg: Trigger input		#PCa0: Block output

3.2.2 Connection Explanations

<u>In: Signal input</u>
The signal input to be calibrated.

<u>X1: Signal input low limit</u>
The X value of the first point of calibration.

<u>Y1: Signal output low limit</u>
The Y value of the first point of calibration.

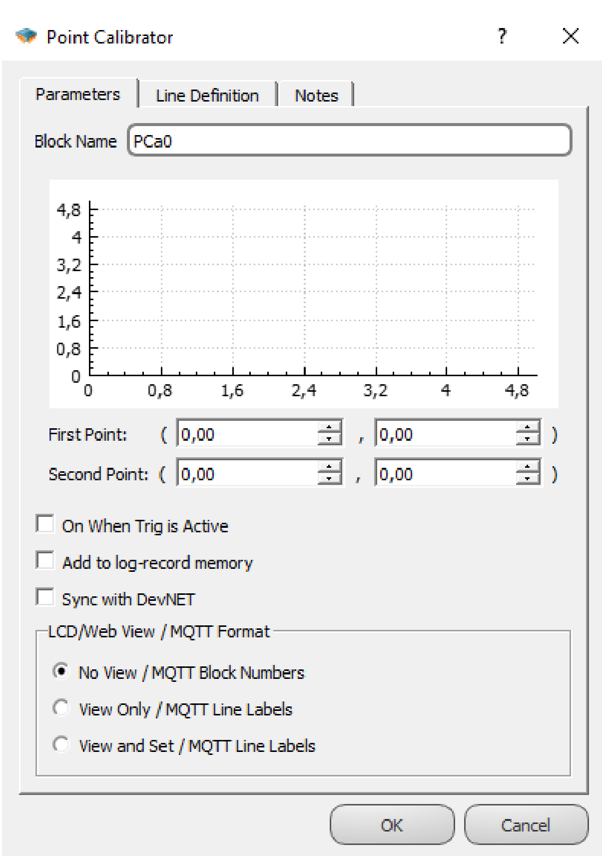
<u>X2: Signal output upper limit</u>
The X value of the second point of calibration.

<u>Y2: Signal output upper limit</u>
The Y value of the second point of calibration.

<u>Trg: Triggering input</u>
It is the block triggering input.

<u>#PCa0: Block output</u>
It is the calibrated block output.

3.2.3 Block Settings

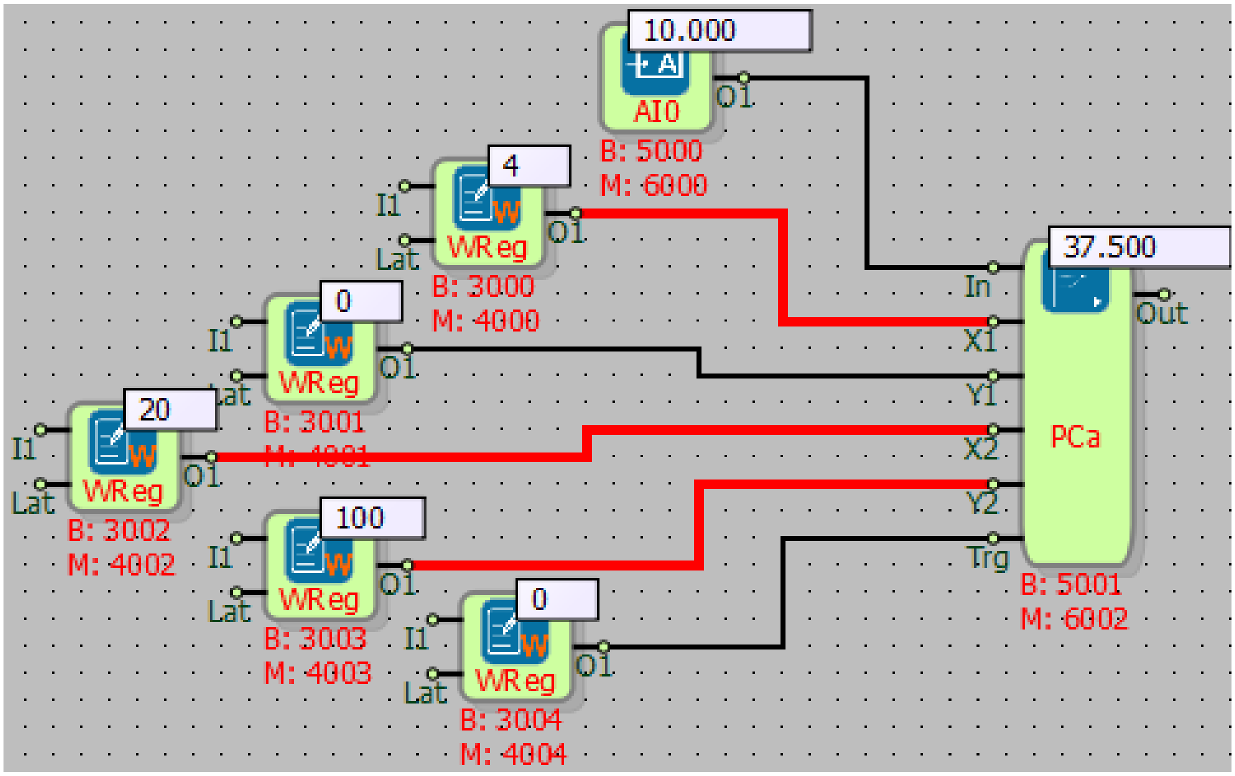
	First point (X): The value of the signal at the input of In. First point (Y): The value of the signal at the output of Out. Second point (X): The value of the signal at the input of In. Second point (Y): The value of the signal at the output of Out.
	On When Trig is Active: Block “Trg” input operation mode is selected. If selected, block input value is processed according to “Trg” input and transferred to the output.

3.2.4 Block Explanations

Especially for analog measurement sensors, there is a linear relationship between the read analog voltage / current value and the actual physical magnitude. This relationship or transformation can be defined by at least two points on the line.

In the point calibrator, instead of defining the slope and offset of the correct equation, the transformation is defined over two sample points.

3.2.5 Sample Application



The minimum value that can be input to the In input is "X1 = 4" and the maximum value is entered as "X2 = 20".

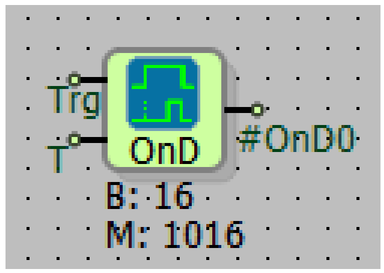
The minimum value that can be read from the Out's output is entered as "Y1 = 0", "Y2 = 100".

Out is "0" when In input is "4", Out is "100" when In input is "20"

4 DELAY/PULSE TIMERS

4.1 ON DELAY

4.1.1 Connections

Trg: The input of block trigger		#OnD0: Block output
T: The time of on delay		

4.1.2 Connection Explanations

<u>Trg: The input of block trigger</u>

It is the block signal input.

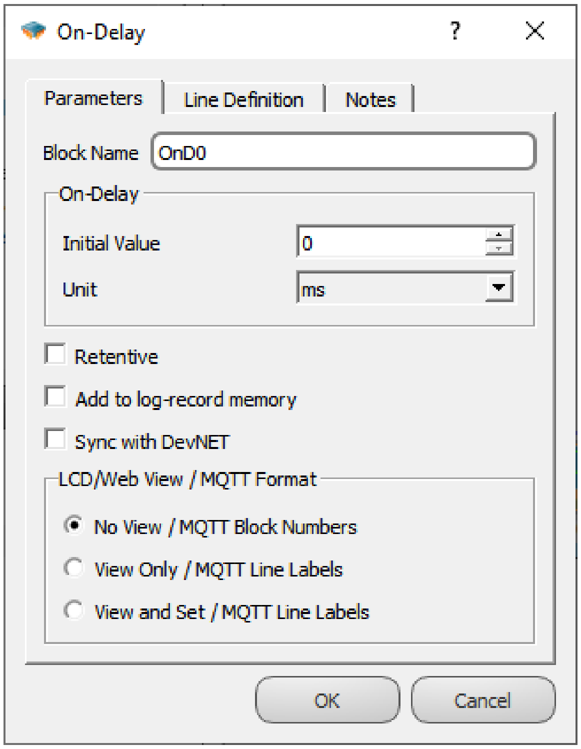
<u>T: The time of on delay</u>

This is the input is used to set the delay time if you require to change delay time using the block input connection

<u>#OnD0: Block output</u>

Block output signal.

4.1.3 Block Settings

	Initial Value (T): The on delay can be set in the block.
	Unit: Unit of time is selected. This selection has following options: milliseconds, seconds, minutes, hours.

4.1.4 Block Explanation

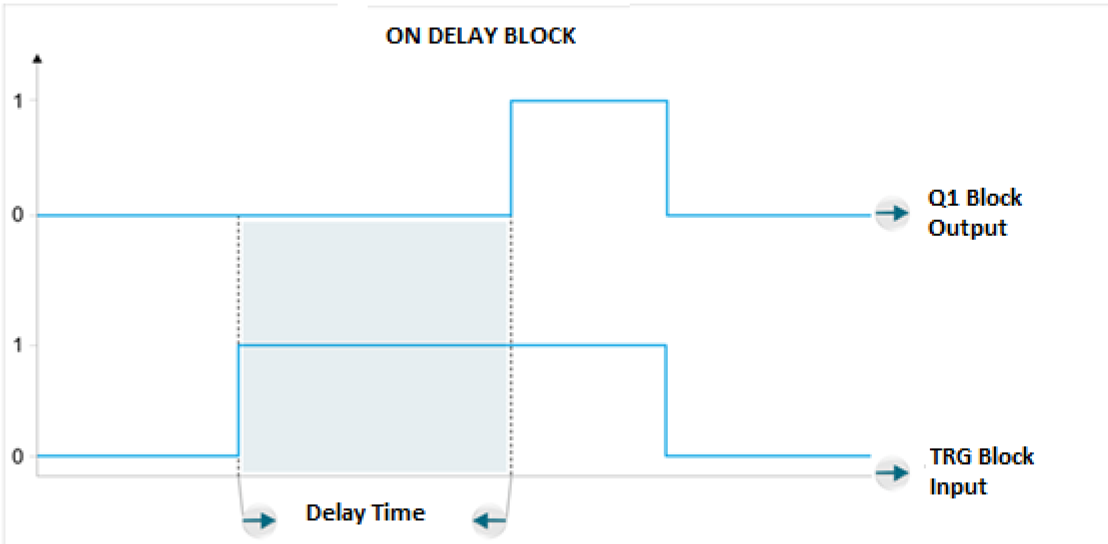
If TRG input change its state to Logic(1) and stay in this state during the determined delay time interval, Q1 output signal change its state from logic(0) to logic(1) after end of the delay time period.

As soon as received Logic(0) signal at Trg input, Q1 output state is changed to Logic(0)

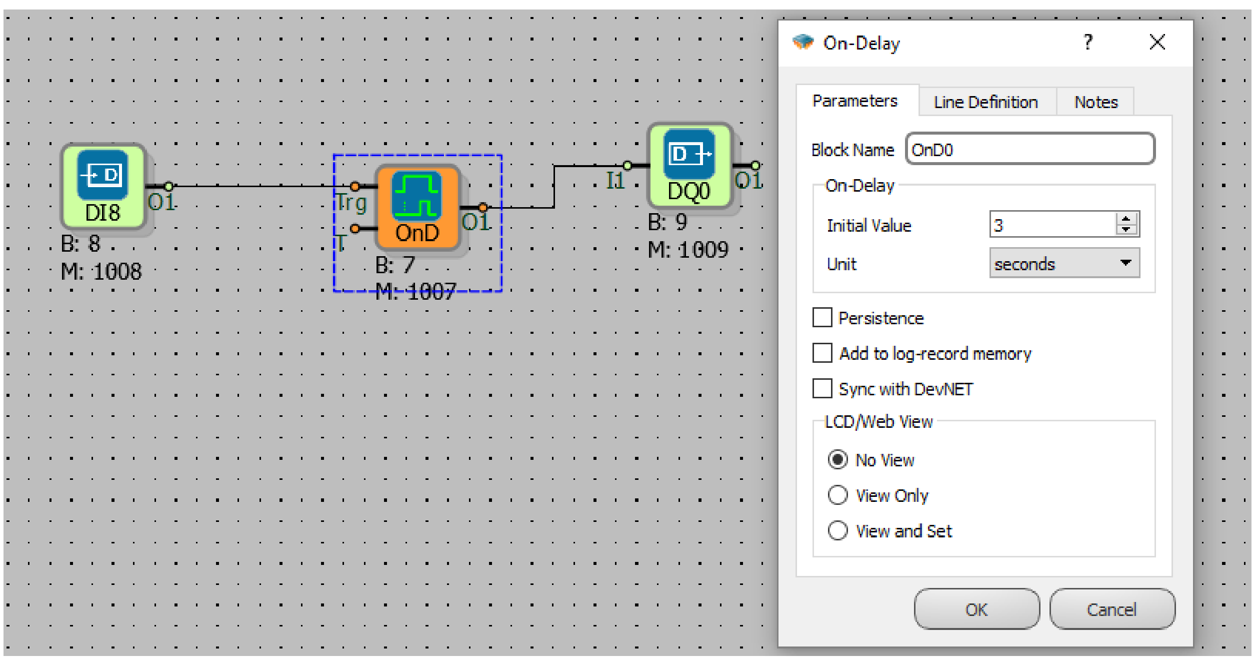
T value can be written in block block settings.

Any type of block signal “word”,”analog” or ”long” can be connected to the T input. T is number which is between the 0-65535 and be careful about variable type range.

4.1.4.1 Signal Flow Diagram



4.1.5 Sample Application

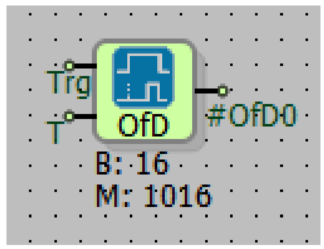


When DI0 goes to logic(1) ,after 3 seconds the DQ0 goes to logic(1).When DI0 goes to logic(0), DQ0 goes to logic(0), immediately.

.

4.2 OFF DELAY

4.2.1 Connections

Trg: The input of block trigger		#OfD0: Block output
T: The time of off delay		

4.2.2 Connection Explanations

Trg: The input of block trigger

It is the block signal input.

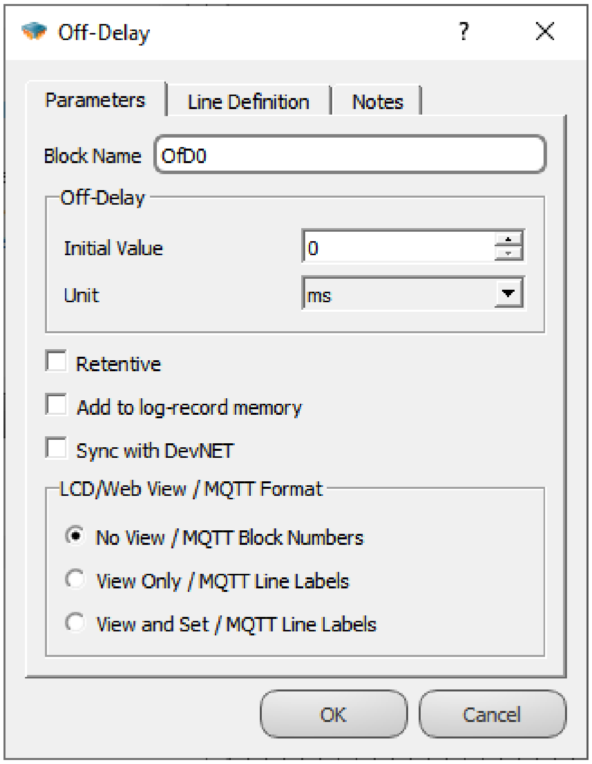
T: The time of off delay

This is the input is used to set the delay time if you require to change delay time using the block input connection

#OfD0: Block output

Block output signal.

4.2.3 Block Settings

<table>
<tr>
<td rowspan="2">
</td>
<td>Initial Value (T): The off delay can be set in the block</td>
</tr>
<tr>
<td>Unit: Unit of time is selected. This selection has following options: milliseconds, seconds, minutes, hours.</td>
</tr>
</table>

4.2.4 Block Explanation

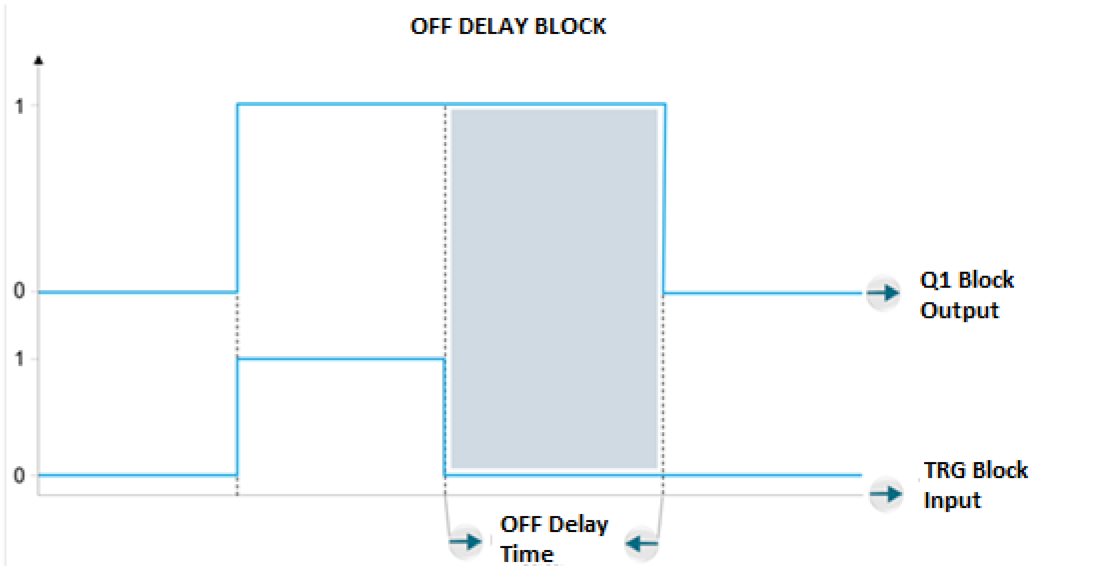
If TRG input change its state to Logic(0) and stay in this state during the determined off delay time interval, Q1 output signal change its state from logic(1) to logic(0) after end of the delay time period.

As soon as received Logic(1) signal at Trg input, Q1 output state is changed to Logic(1) immediately.

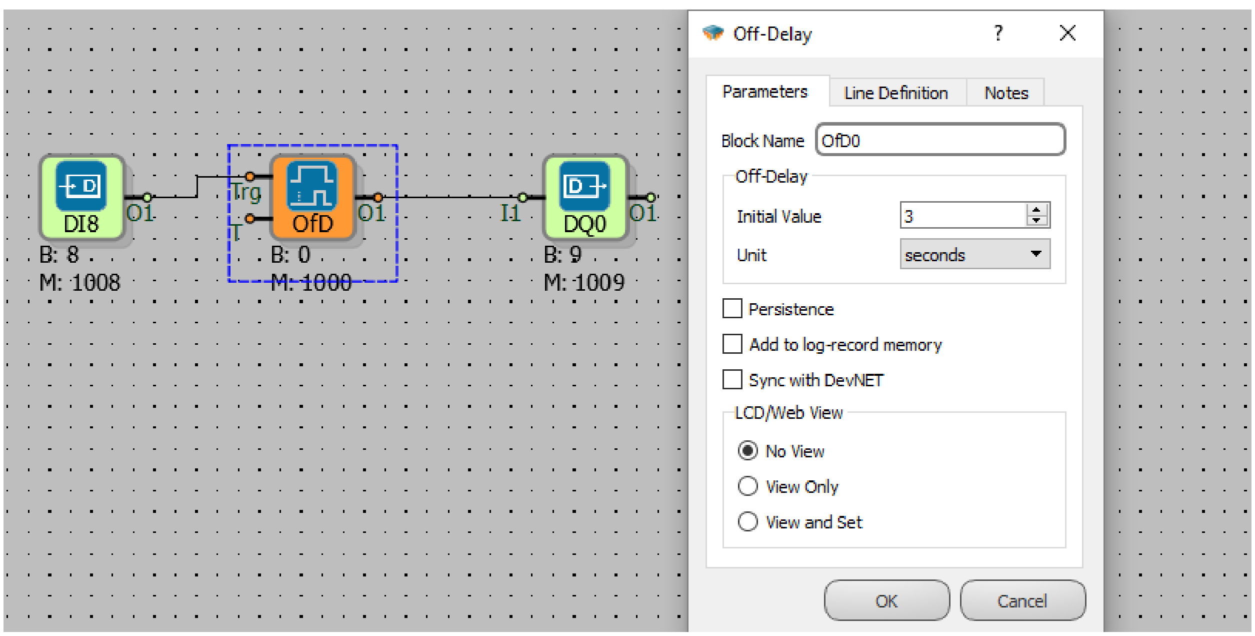
T value can be written in block block settings.

Any type of block signal “word”,”analog” or ”long” can be connected to the T input. T is number which is between the 0-65535 and be careful about variable type range.

4.2.4.1 Signal Flow Diagram



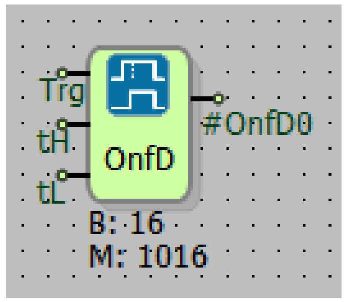
4.2.5 Sample Application



When DI0 goes to logic(0) ,after 3 seconds the DQ0 goes to logic(0).When DI0 goes to logic(1), DQ0 goes to logic(1), immediately.

4.3 ON/OFF DELAY

4.3.1 Connections

Trg: The input of block trigger		#OnfD0: Block output
tH: The input of time of on delay		
tL: The input of time of off delay		

4.3.2 Connection Explanations

Trg: The input of block trigger

It is the block signal input.

tH: The input of time of on delay

This is the input is used to set the ON delay time if you require to change ON delay time using the block input connection

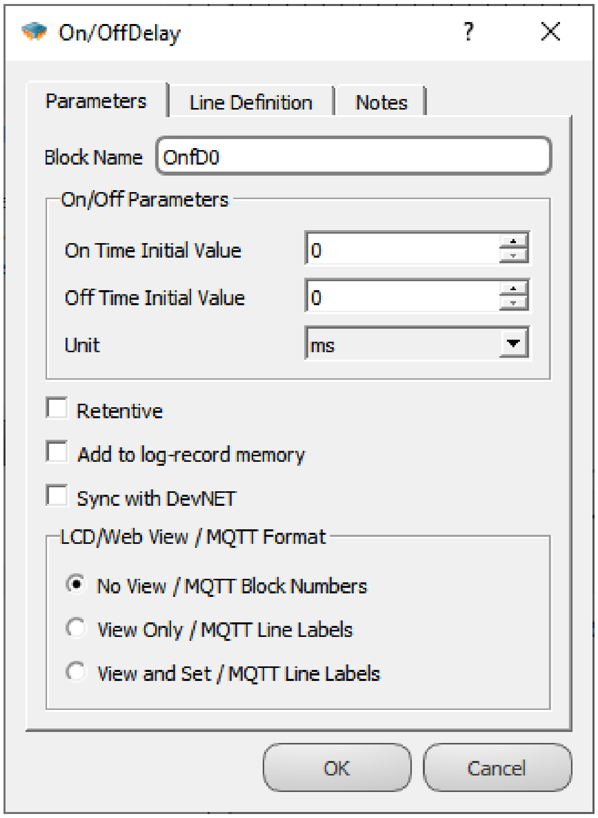
tL: The input of time of off delay

This is the input is used to set the OFF delay time if you require to change OFF delay time using the block input connection

#OnfD0: Block output

Block output signal.

4.3.3 Block Settings

<table>
<tr><td rowspan="3">
</td><td>On Time Initial Value (tH): The on delay can be set in the block</td></tr>
<tr><td>Off Time Initial Value (tL): The of off delay can be set in the block</td></tr>
<tr><td>Unit of time is selected. This selection has following options: milliseconds, seconds, minutes, hours.
Unit of ON Delay Time and OFF Delay Time has only this single selection. Both of them must have same unit.</td></tr>
</table>

4.3.4 Block Explanation

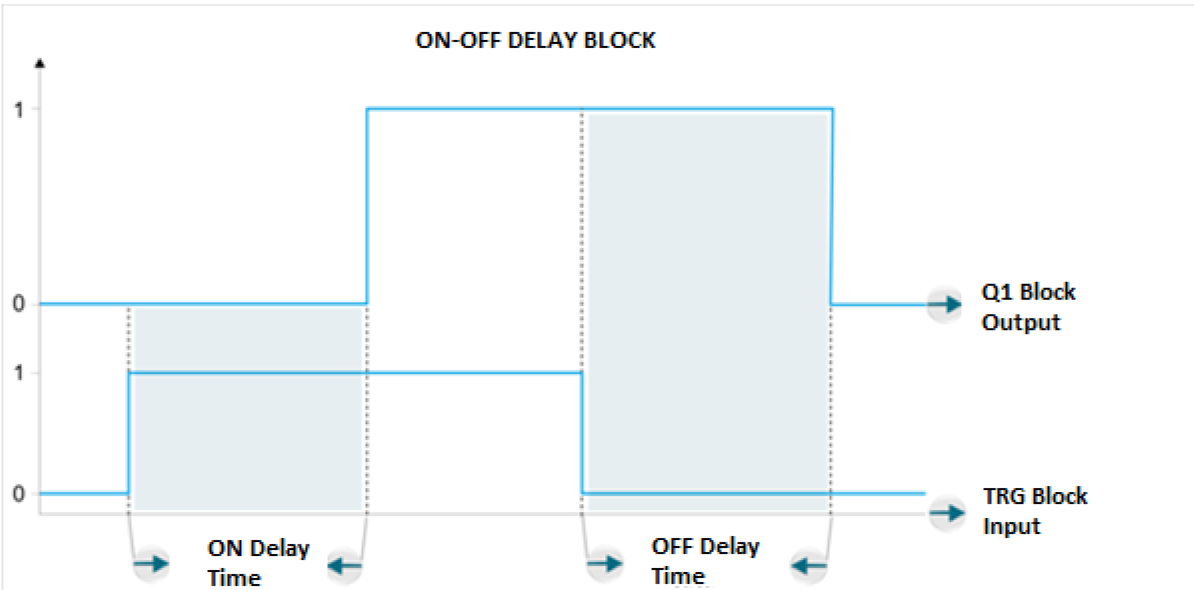
If TRG input change its state to Logic(1) and stay in this state during the determined ON delay time interval, Q1 output signal change its state from logic(0) to logic(1) after end of the ON delay time period.

And same way, If TRG input change its state to Logic(0) and stay in this state during the determined OFF delay time interval, Q1 output signal change its state from logic(1) to logic(0) after end of the OFF delay time period.

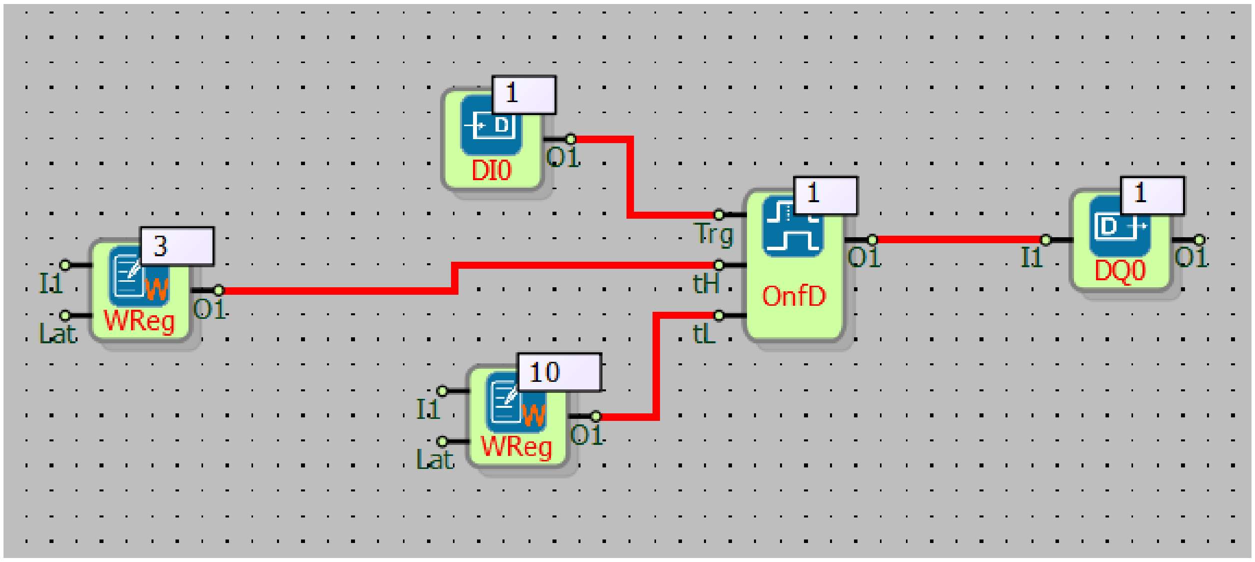
Any changes at the TRG input with shorter duration than user defined delay times does not change the status of the Q1 block output.

TON and TOFF values can be written in block block settings or can be applied by related block inputs. Any type of block signal “word”,”analog” or ”long” can be connected to these inputs. T is number which is between the 0-65535 and be careful about variable type range.

4.3.4.1 Signal Flow Diagram



4.3.5 Sample Application



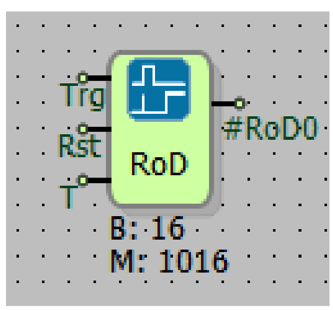
The on / off delay timing scale "seconds" is selected and the tH and tL values are entered from outside the block.

DQ0 becomes logic (1) 3 seconds after the DI0 logic (1) becomes logic (0).

DQ0 is logic (0) after 10 seconds from the logic (1) to the logic (0).

4.4 RETENTIVE ON DELAY

4.4.1 Connections

Trg: The input of block trigger		#RoD0: Block output
Rst: The input of reset		
T: The input of on delay time		

4.4.2 Connection Explanations

<u>Trg: The input of block trigger</u>

It is the block signal input.

<u>Rst: The input of reset</u>

Raising edge at RST input signal resets permanent Logic (1) state and re-initiate the block.

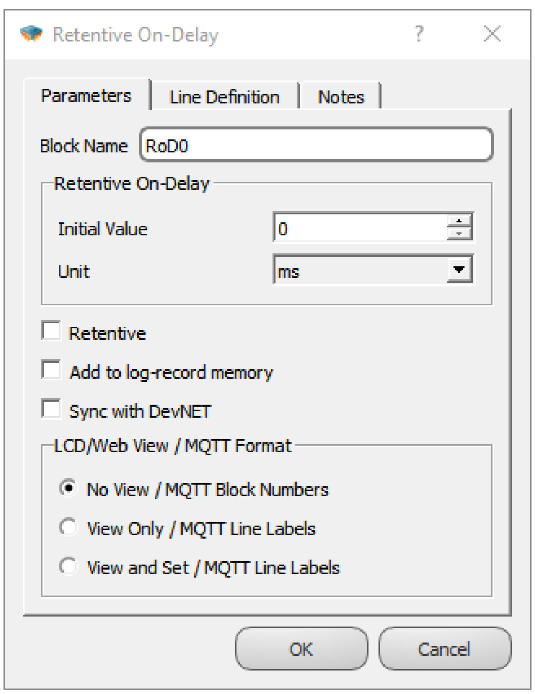
<u>T: The input of on delay time</u>

This is the input is used to set the delay time if you require to change delay time using the block input connection

<u>#RoD0: Block output</u>

Block output signal.

4.4.3 Block Settings

<table>
<tr>
<td rowspan="2">
</td>
<td>İnitial Value (T): The time of on delay is set in the block</td>
</tr>
<tr>
<td>Unit: Unit of time is selected. This selection has following options: milliseconds, seconds, minutes, hours.</td>
</tr>
</table>

4.4.4 Block Explanation

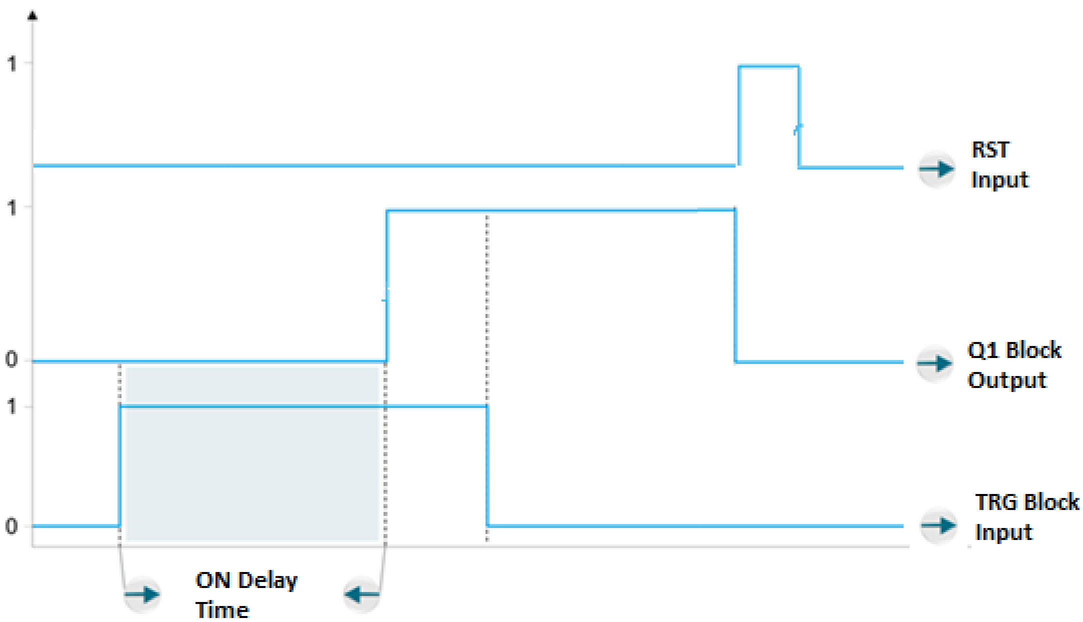
If TRG input change its state to Logic(1) and stay in this state during the determined delay time interval, Q1 output signal change its state from logic(0) to logic(1) after end of the delay time period. After Q1 state goes to Logic(1), Q1 output signal keeps its states as long as receiving a rising edge at RST input.

When a rising edge signal applied to RST input, Q1 Block output goes to Logic(0)

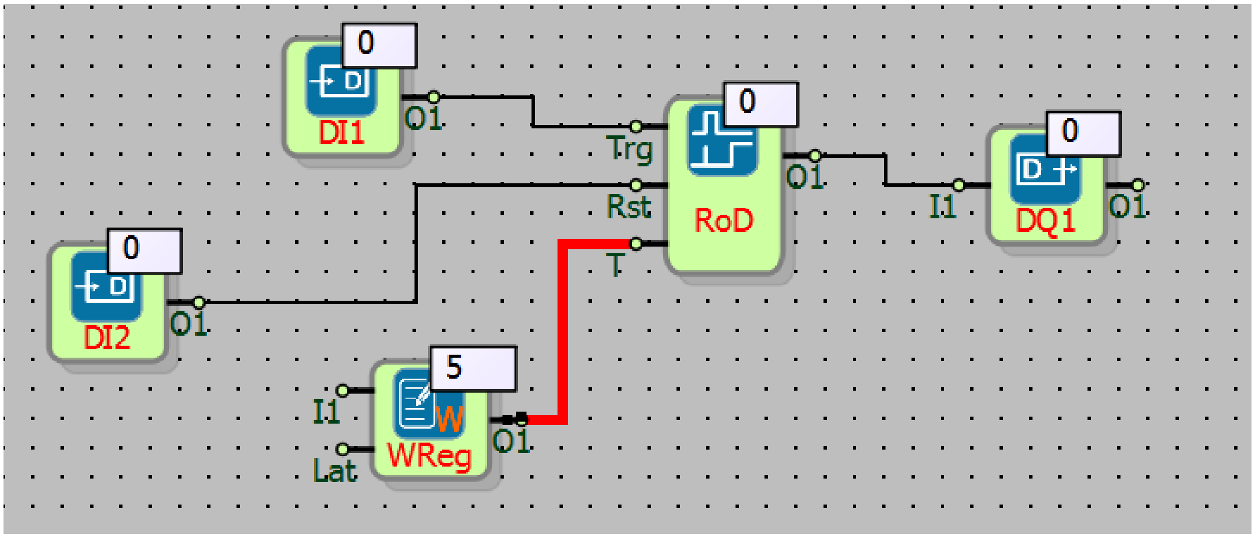
T value can be written in block block settings.

Any type of block signal “word”,”analog” or ”long” can be connected to the T input. T is number which is between the 0-65535 and be careful about variable type range.

4.4.4.1 Signal Flow Diagram



4.4.5 Sample Application



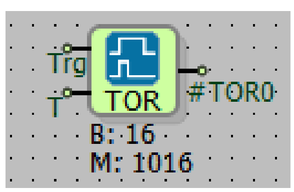
Retentive on delay block timing scale "seconds" is selected and T value is entered from outside the block. After 5 seconds from DI1 logic (1) to logic (0), DQ1 becomes logic (1).

When DQ1 is logic (1), DQ1 maintains the logic (1) position even if DI1 is logic (0).

At the rising edge of the logic (1) signal, called Rst input DI2, DQ1 goes to logic (0)

4.5 TIMER OUTPUT RELAY

4.5.1 Connections

Trg: The input of block trigger		#TOR0: Block output
T: The input of timer parameter		

4.5.2 Connection Explanations

Trg: The input of block trigger

It is the block signal input.

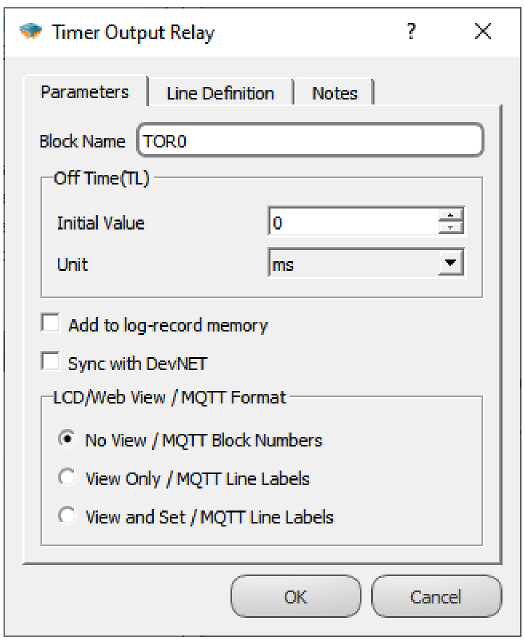
T: The input of timer parameter

This is the input is used to set the delay time if you require to change delay time using the block input connection

#TOR0: Block output

Block output signal.

4.5.3 Block Settings

<table>
<tr><td rowspan="2">
</td><td>Initial Value (T): Timer parameter is set from in the block</td></tr>
<tr><td>Unit: Unit of time is selected. This selection has following options: milliseconds, seconds, minutes, hours.</td></tr>
</table>

4.5.4 Block Explanation

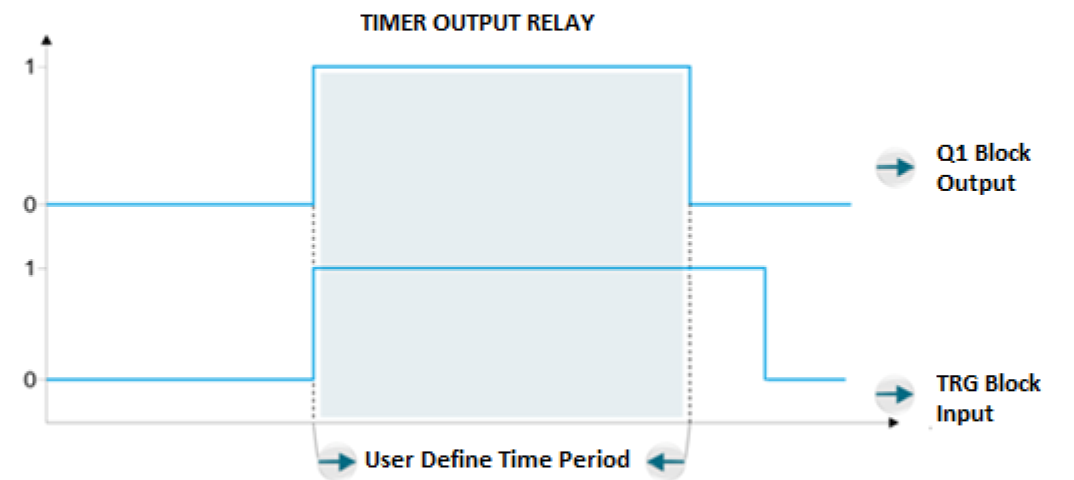
When the Trg input change its state from Logic(0) to Logic(1), Q1 output changes its state immediately to Logic(1). Block keep its Q1 Logic(1) state only user defined duration of time and after that time period expire, Q1 state goes to Logic(0) state automatically.

As soon as received Logic(0) signal at Trg input, Q1 output state is changed to Logic(0)

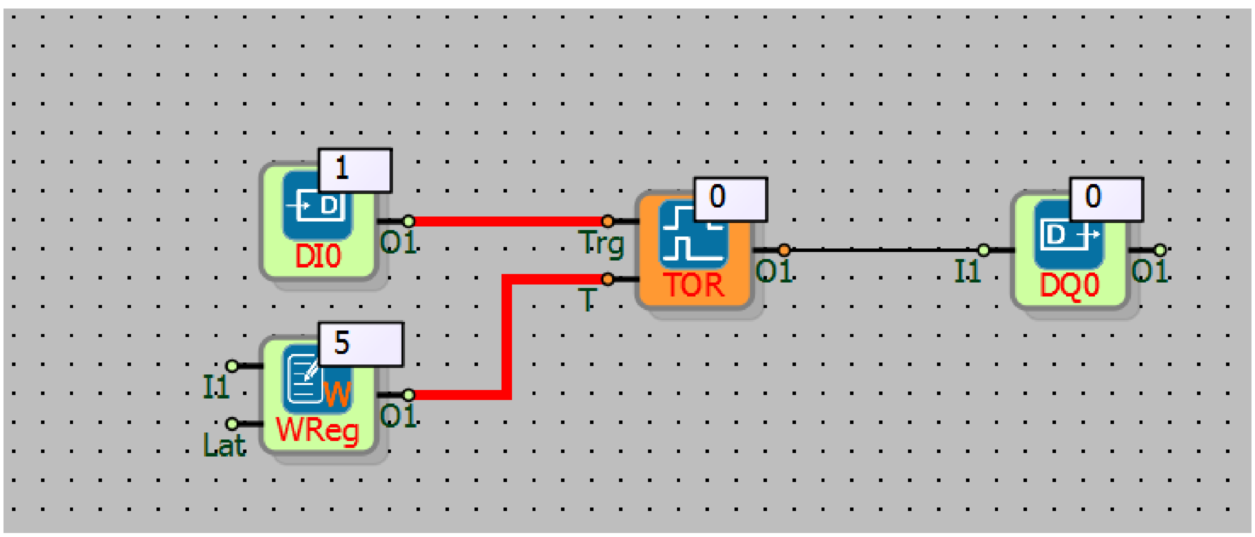
T value can be written in block block settings.

Any type of block signal “word”,”analog” or ”long” can be connected to the T input. T is number which is between the 0-65535 and be careful about variable type range.

4.5.4.1 Signal Flow Diagram



4.5.5 Sample Application

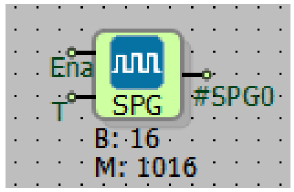


Timer output relay timer period is choosen as seconds from the blocks, T value is entered from the out of block.

When DI0 is logic(1), DQ0 will be logic(1).When DI0 is logic(1), after 5 seconds DQ0 will be logic(0).

4.6 SYMETRIC PULSE GENERATOR

4.6.1 Connections

<table>
<tr><td>Ena: The input of block activation</td><td rowspan="2">
</td><td rowspan="2">#SPG0: Block output</td></tr>
<tr><td>T: The input of timer parameter</td></tr>
</table>

4.6.2 Connection Explanations

Ena: The input of block activation

It is the input of block activation the symmetric pulse generator.

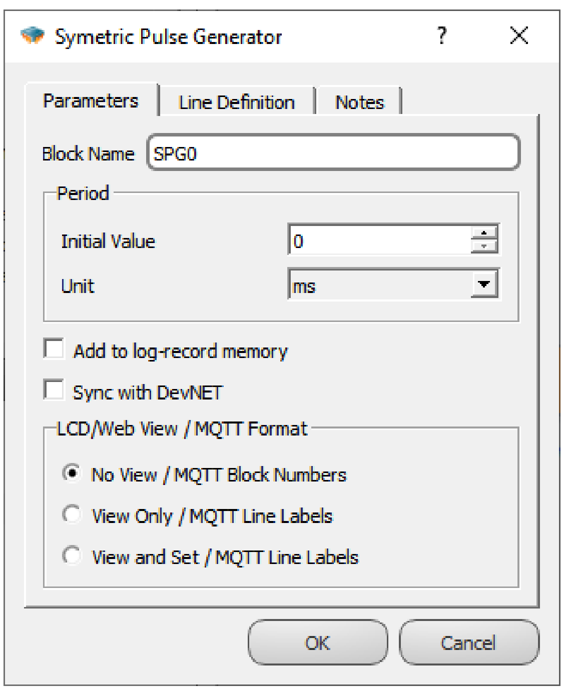
T: The input of timer parameter

The input of the symmetric pulse generator's time parameter from outside the block.

#SPG0: Block output

When Ena input is logic(1), It is block output which is logic(1-0) as symmetric.

4.6.3 Block Settings

<table>
<tr><td rowspan="2">
</td><td>Initial Value (T): Timer parameter is set from in the block.</td></tr>
<tr><td>Unit: Unit of time is selected. This selection has following options: milliseconds, seconds, minutes, hours.</td></tr>
</table>

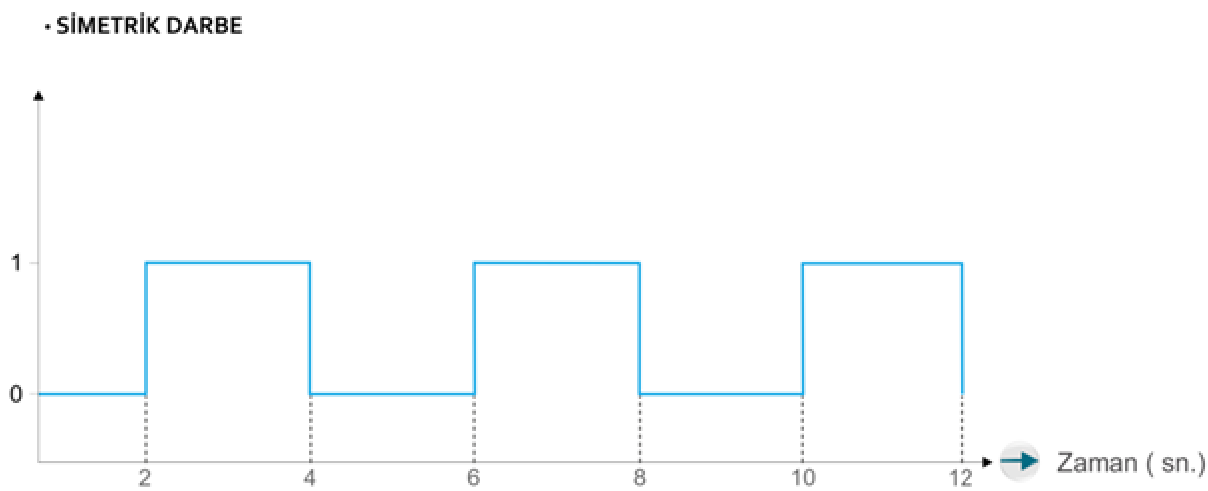
4.6.4 Block Explanation

When Enb input is logic(1), Q1 block output produces periodic symetric pulses in 2*T time period as Logic(0) for T period of time and Logic(1) for T period of time.

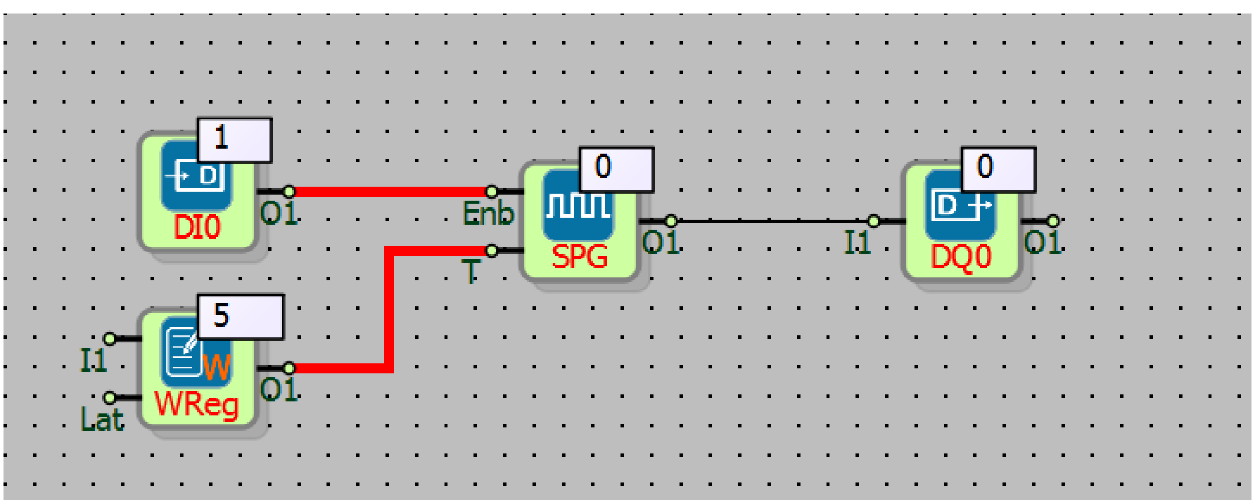
T value can be written in block block settings.

Any type of block signal “word”,”analog” or ”long” can be connected to the T input. T is number which is between the 0-65535 and be careful about variable type range.

4.6.4.1 Signal Flow Diagram



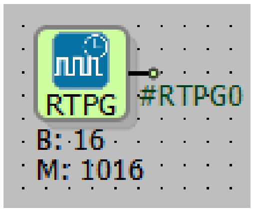
4.6.5 Sample Application



When DI0 is logic(1), DQ0 will be 5 seconds logic(0), 5 seconds logic(1) periodically.

4.7 REAL TIME PULSE GENERATOR

4.7.1 Connections

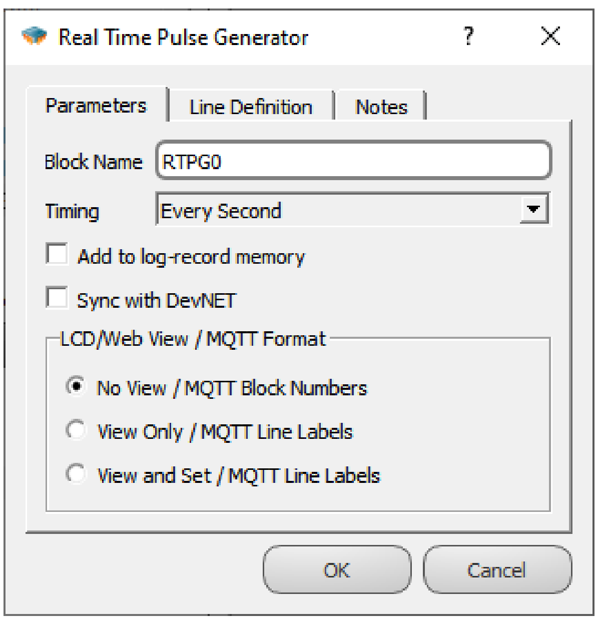
	#RTPG0: Block output

4.7.2 Connection Explanations

<u>#RTPG0: Block output</u>

It is the block output which produce the logic(1) pulse in is described from in the block in the timer period.

4.7.3 Block Settings

	Timing : Pulse period choice can be done from in the block. It cannot be choosen from block inputs.

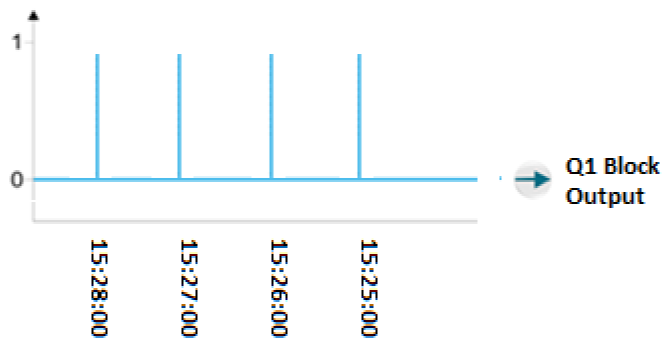
4.7.4 Block Explanation

It periodically generates pulses at the times specified in synchronous with the device's real time clock.

Different time can be chosen from in the block settings.

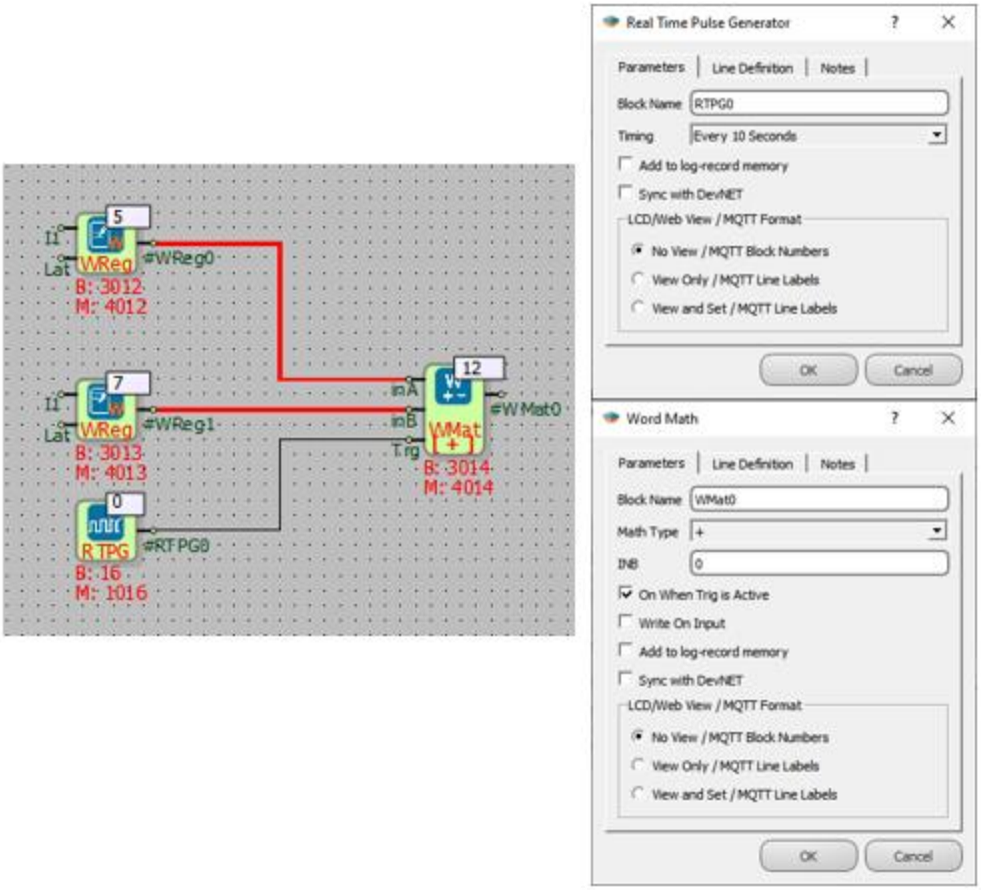
Output of the block is a single cycle time pulse that is generated every specified time events.

4.7.4.1 Signal Flow Diagram



In the example timer parameter is choosen in every minutes in the real time pulse generator And the device is started at time 15:27:12. So, In real time events of minutes was gained logic pulse output.

4.7.5 Sample Application

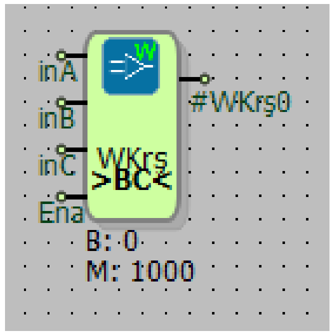


In the example, the Real-Time Pulse Generator (GZDU) block is programmed to generate a pulse every 10 seconds. With the GZDU block connected to the “Trg” input of the Word Math block, the values in the block inputs were collected every 10 seconds and written to the block output. (On When Trig is Active option must be selected in the Word Math block for the trigger of the GZDU block to be available for this example.)

5 MATHEMATICAL OPERATION BLOCKS

5.1 WORD COMPARATOR

5.1.1 Connections

<table>
<tr><td>inA: 1. Word input</td><td rowspan="4">
</td><td rowspan="4">#WKrş0: Output of the block</td></tr>
<tr><td>inB: 2. Word input</td></tr>
<tr><td>inC: 3. Word input</td></tr>
<tr><td>Ena: Enable Input</td></tr>
</table>

5.1.2 Connection Explanations

inA: 1. Word input

Word value to be compared.

inB: 2. Word input

Word value to be compared.

inC: 3. Word input

Word value to be compared.

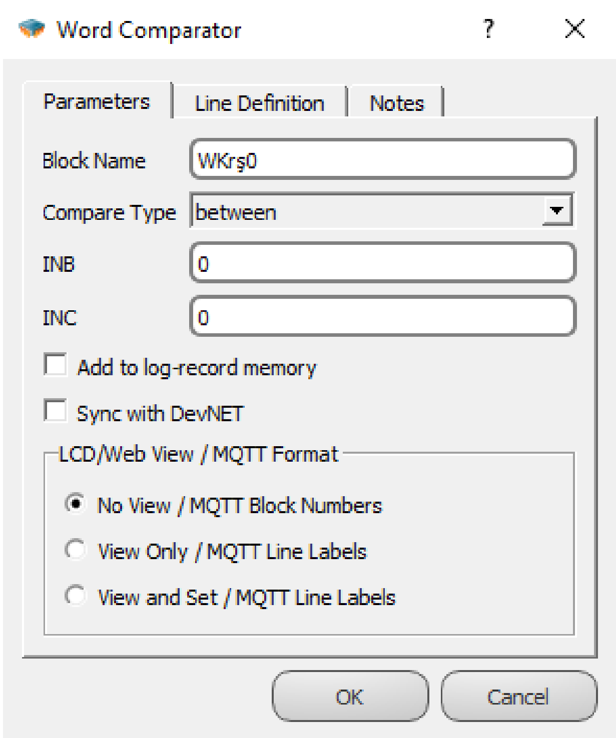
Ena: Block Enable

Block is activated, when Enb input goes to HIGH

#WKrş0: Output of the block

If the conditions are satisfied, output is “1” or HIGH, otherwise is LOW

5.1.3 Block Settings

	Compare Type: Comparison type is specified here.
	INB: Bottom threshold value is entered here in Block Settings.
	INC: Upper threshold value is entered here in Block Settings.

5.1.4 Block Explanation

It is used for comparing 16-bit WORD numbers. (0-65535) The value at the “inA” input of theblock is compared to the values at the “inB” and “inC” inputs of the block according to thecompare type specified in the block settings menu. Block must be activated with sending a HIGH signal to the “Ena” input of the block.

If the comparing condition is satisfied, output of the block becomes “1” or HIGH, otherwise it is “0” or LOW.

Desired threshold values for comparing can be selected in Block Settings menu or they can be adjusted with “inB” and “inC” inputs of the block by connecting a register to the inputs.

With the Word Comparator Block, “greater than”, “smaller than”, “out of range”, “equal to”, “greater than or equal to”, “smaller than or equal to”, “not equal to” operations can be performed.

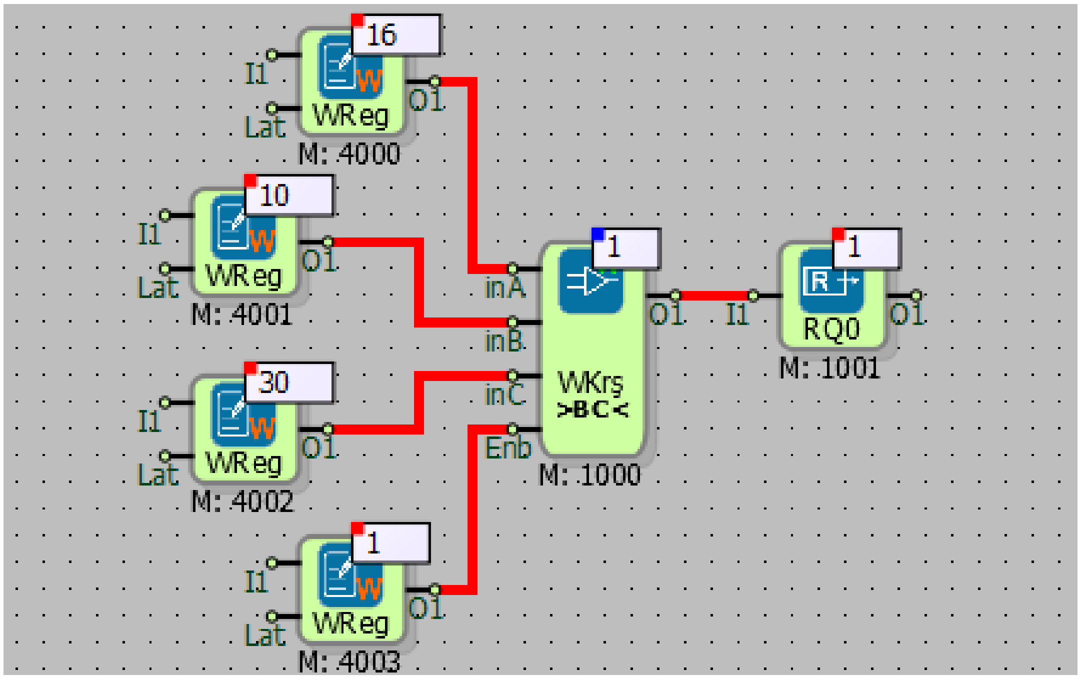
For the operations “greater than”, “smaller than”, “greater than or equal to”, “smaller thanor equal to”, “not equal to”; the value at the “inA” input of the block is compared to the value at the “inB” input of the block.

For the operations “in range” and “out of range”; the value at the inA input of the block is compared to the values at the “inB” and “inC” inputs of the block.

If “in range” or “out of range” operations are going to be used, the value at the “inB” input of the block should be smaller than the value at the “inC” input of the block. (inB < inC) If the signal at the “Ena” input of the block goes to “0” from “1” while the output is equal to “1” or output is HIGH, output of the block will stay the same

Comparison Type	**Used Inputs**	**Enb**	**Explanation**
Equal To	inA, inB	1	If inA = inB then #WKRŞ0 output is “1”.
Greater Than	inA, inB	1	If inA > inB then #WKRŞ0 output is “1”.
Smaller Than	inA, inB	1	If inA < inB then #WKRŞ0 output is “1”.
Greater Than or Equal To	inA, inB	1	If inA ≥ inB then #WKRŞ0 output is “1”.
Smaller Than or Equal To	inA, inB	1	If inA ≤ inB then #WKRŞ0 output is “1”.
Not Equal To	inA, inB	1	If inA ≠ inB then #WKRŞ0 output is “1”.
In Range	inA, inB, inC	1	If inB < inA < inC then #WKRŞ0 output is “1”.
Out of Range	inA, inB, inC	1	If inB < inC < inA or inA < inB < inC then #WKRŞ0 output is “1”.
-	-	0	Previous output preserved; output not updated.

5.1.5 Sample Application

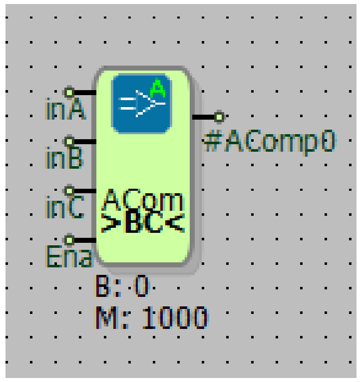


In this example, comparison type is selected as "In Range".

The block is enabled with the HIGH signal at the "Enb" input of the block, while the value at the "inA" input of the block has a value between the value at the "inB" input of the block (bottom threshold) and the value at the "inC" input of the block(upper threshold), the output is HIGH or "1", therefore the Relay Output takes the "1" value.

5.2 ANALOG COMPARATOR

5.2.1 Connections

inA: 1. Analog Input		#AComp0: Output of the block
inB: 2. Analog Input		
inC: 3. Analog Input		
Ena: Enable input		

5.2.2 Connection Explanations

inA: 1. Analog input

Analog value to be compared.

inB: 2. Analog input

Lower analog threshold value to be compared.

inC: 3. Analog input

Upper analog threshold value to be compared.

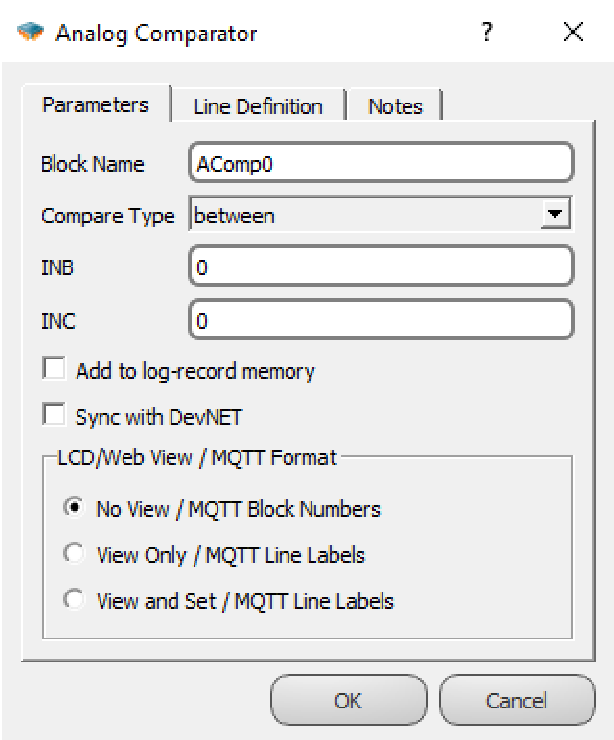
Ena: Enable block

Block is activated with this input.

#AComp0: Output of the block

If the conditions are satisfied, output is “1” or HIGH.

5.2.3 Block Settings

	Compare Type: Comparison type is specified here.
	INB: Bottom threshold value is entered here in Block Settings.
	INC: Upper threshold value is entered here in Block Settings.

5.2.4 Block Explanation

It is used for comparing 32-bit floating point numbers. The value at the “inA” input of the block is compared to the values at the “inB” and “inC” inputs of the block according to the compare type specified in the block settings menu. Block must be activated with sending a HIGH signal to the “Ena” input of the block.

If the comparing condition is satisfied, output of the block becomes “1” or HIGH, otherwise it is “0” or LOW.

Desired threshold values for comparing can be selected in Block Settings menu or they can be adjusted with “inB” and “inC” inputs of the block by connecting a register to the inputs. With the Analog Comparator block, “greater than”, “smaller than”, “out of range”, “equal to”, “greater than or equal to”, “smaller than or equal to”, “not equal to” operations can be performed.

For the operations “greater than”, “smaller than”, “greater than or equal to”, “smaller than or equal to”, “not equal to”; the value at the “inA” input of the block is compared to the value at the “inB” input of the block.

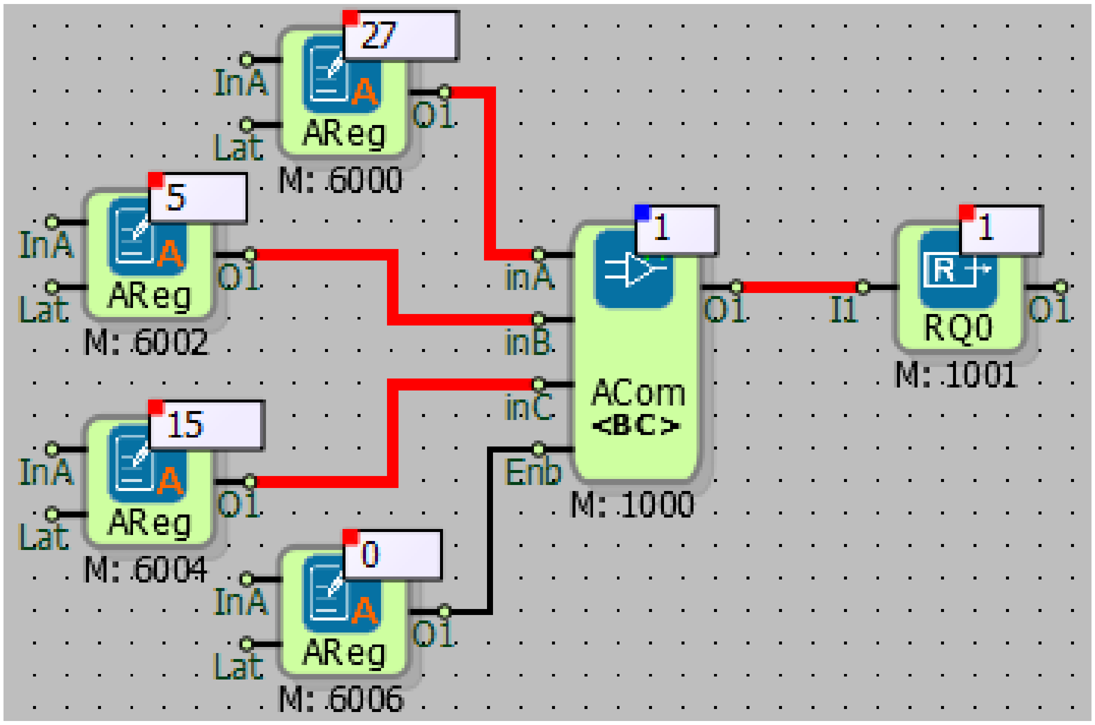
For the operations “in range” and “out of range”; the value at the “inA” input of the block is compared to the values at the “inB” and “inC” inputs of the block.

If “in range” or “out of range” operations are going to be used, the value at the “inB” input of the block should be smaller than the value at the “inC” input of the block. (inB < inC)

If the signal at the “Ena” input of the block goes to “0” from “1” while the output is equal to “1” or output is HIGH, output of the block will stay the same.

Comparison Type	**Used Inputs**	**Ena**	**Explanation**
Equal To	inA, inB	1	If inA = inB then #AComp0 output is “1”.
Greater Than	inA, inB	1	If inA > inB then #AComp0 output is “1”.
Smaller Than	inA, inB	1	If inA < inB then #AComp0 output is “1”.
Greater Than or Equal To	inA, inB	1	If inA ≥ inB then #AComp0 output is “1”.
Smaller Than or Equal To	inA, inB	1	If inA ≤ inB then #AComp0 output is “1”.
Not Equal To	inA, inB	1	If inA ≠ inB then #AComp0 output is “1”.
In Range	inA, inB, inC	1	If inB < inA < inC then #AComp0 output is “1”.
Out of Range	inA, inB, inC	1	If inB< inC < inA or inA < inB < inC then #AComp0 output is “1”.
-	-	0	Previous output preserved; output not updated

5.2.5 Sample Application

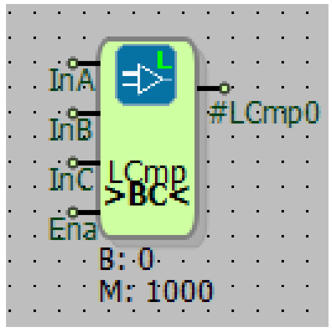


In this example, comparison type is selected as “Out of Range”.

The block is enabled with the HIGH signal at the Enb input of the block, while the value at the inAinput of the block does not have a value between the value at the inB input of the block (bottom threshold) and the value at the inC input of the block (upper threshold), the output is HIGH or “1”, therefore the Relay Output takes the “1” value.

5.3 LONG COMPARATOR

5.3.1 Connections

InA: 1. Long input		#LCmp0: Output of the block
InB: 2. Long input		
InC: 3. Long input		
Ena: Enable input		

5.3.2 Connection Explanations

InA: 1. Long input

Long value to be compared.

InB: 2. Long input

Bottom long threshold value to be compared.

InC: 3. Long input

Upper long threshold value to be compared.

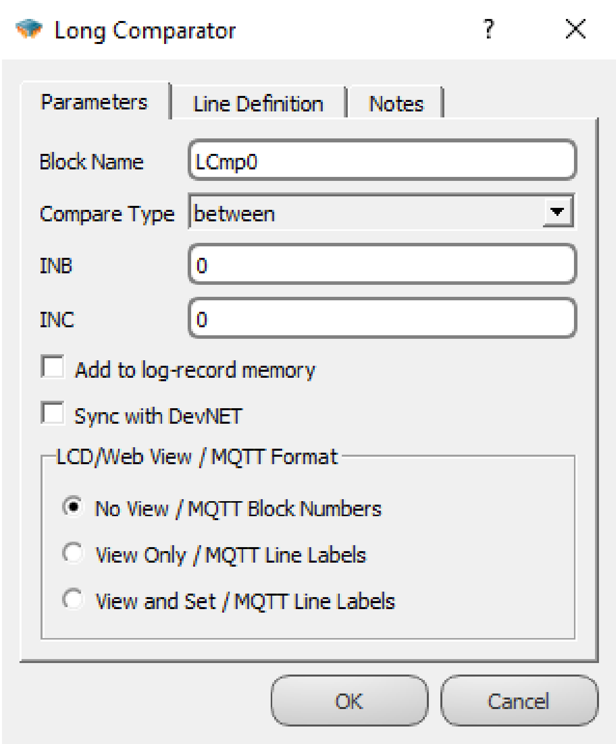
Ena: Enable input

Block is activated with this input.

#LCmp0: Output of the block

IF the conditions are satisfied, output is "1" or HIGH.

5.3.3 Block Settings

<table>
<tr><td rowspan="3">
</td><td>Compare Type: Comparison type is specified here.</td></tr>
<tr><td>INB: Bottom threshold value is entered here in Block Settings.</td></tr>
<tr><td>INC: Upper threshold value is entered here in Block Settings.</td></tr>
</table>

5.3.4 Block Explanation

It is used for comparing 32-bit signed numbers. The value at the “inA” input of the block is compared to the values at the “inB” and “inC” inputs of the block according to the compare type specified in the block settings menu. Block must be activated with sending a HIGH signal to the “Ena” input of the block.

If the comparing condition is satisfied, output of the block becomes “1” or HIGH, otherwise it is “0” or LOW.

Desired threshold values for comparing can be selected in Block Settings menu or they can be adjusted with “inB” and “inC” inputs of the block by connecting a register to the inputs.

With the Long Comparator block, “greater than”, “smaller than”, “out of range”, “equal to”, “greater than or equal to”, “smaller than or equal to”, “not equal to” operations can be performed.

For the operations “greater than”, “smaller than”, “greater than or equal to”, “smaller than or equal to”, “not equal to”; the value at the “inA” input of the block is compared to the value at the “inB” input of the block.

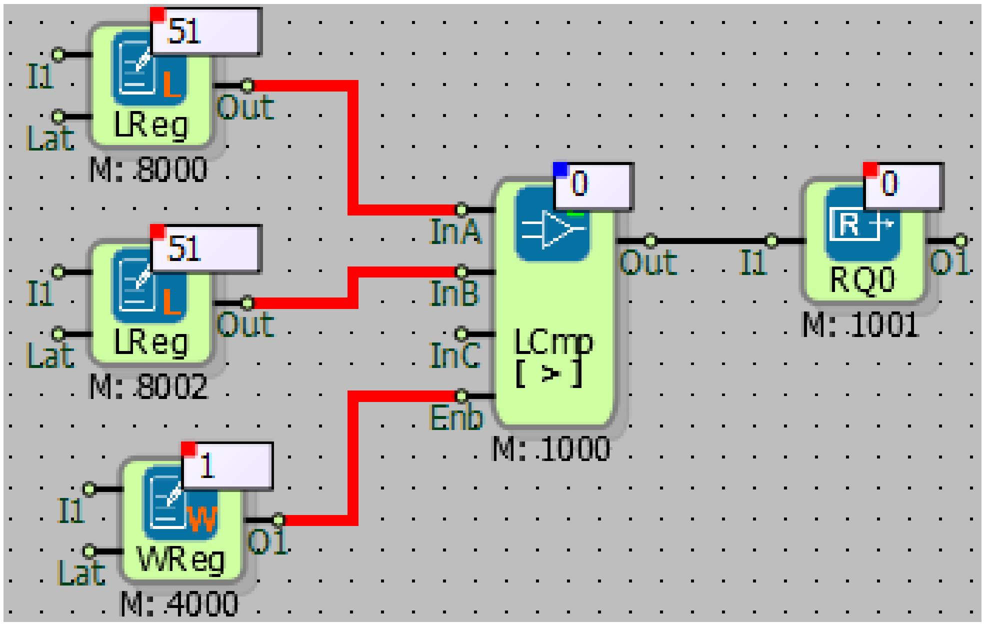
For the operations “in range” and “out of range”; the value at the “inA” input of the block is compared to the values at the “inB” and “inC” inputs of the block.

If “in range” or “out of range” operations are going to be used, the value at the “inB” input of the block should be smaller than the value at the “inC” input of the block. (inB < inC)

If the signal at the “Ena” input of the block goes to “0” from “1” while the output is equal to “1”or output is HIGH, output of the block will stay the same.

Comparison Type	**Used Inputs**	**Enb**	**Explanation**
Equal To	inA, inB	1	If inA = inB then #LCmp0 output is “1”.
Greater Than	inA, inB	1	If inA > inB then #LCmp0 output is “1”.
Smaller Than	inA, inB	1	If inA < inB then #LCmp0 output is “1”.
Greater Than or Equal To	inA, inB	1	If inA ≥ inB then #LCmp0 output is “1”.
Smaller Than or Equal To	inA, inB	1	If inA ≤ inB then #LCmp0 output is “1”.
Not Equal To	inA, inB	1	If inA ≠ inB then #LCmp0 output is “1”.
In Range	inA, inB, inC	1	If inB < inA < inC then #LCmp0 output is “1”.
Out of Range	inA, inB, inC	1	If inB < inC < inA or inA < inB < inC then #LCmp0 output is “1”.
-	-	0	Previous output preserved; output not updated

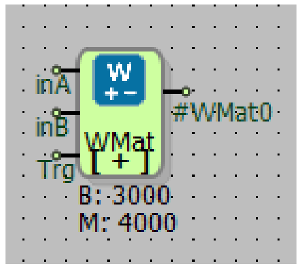
5.3.5 Sample Application



In this example, comparison type is selected as "Greater". The block is enabled with the HIGH signal at the Enb input of the block, while the value at the inA input of the block has a value equal to the value at the inB input of the block(lower threshold), so the output is LOW or "0", therefore the Relay Output takes the "0" value.

5.4 WORD MATH

5.4.1 Connections

inA: WORD data input		#WMat0: WORD output
inB : WORD data input		
Trg: Trigger input		

5.4.2 Connection Explanations

inA: WORD input

WORD value to be processed.

inB: WORD input

WORD value to be processed.

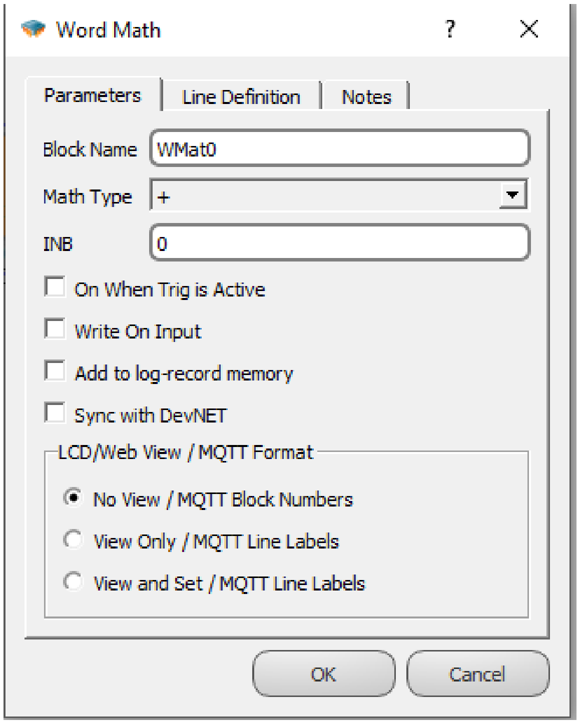
Trg: Trigger input

If the “On When Trig is Active” is selected in Block Settings menu, block is activated at each rising edge detected at the Trg input of the block.

#WMat0: WORD output

16-bit WORD output of the block.

5.4.3 Block Settings

	Math Type: Mathematical operation is specifed here.
	INB: WORD input to be processed.
	On When Trig is Active: If selected, block is activated at each rising edge detected at the Trg input of the block
	Write On Input: If this option is selected, the value at the inA input of the block and the value at the inB input of the block is processed. Result of the operation is written on the inA input of the block. An WORD register should be connected to the inA input of the block. This operation is performed at each PLC cycle by default. If “On When Trig is Active” option is selected, this operation is performed at each rising edge detected on the Trig input of the block.

5.4.4 Block Explanation

It is used for mathematical operations which result in range 0-65535(16-bit). With Word Math block “addition”, “subtraction”, “multiplication”, “division”, “logic AND”, “logic OR”, “logic XOR”, “shift left”, “shift right”, “checkBit”, “LeftShiftCheckFirst”, “RightShiftCheckFirst”, “LeftShiftCheckLast”, “RightShiftCheckLast”, “absolute value”, “bit compare”, “mod”, “bit replace”, “get”, “low limit”, “high limit”, “merge A-B” and “set” operations can be performed.

On When Trig is Active: If this option is selected, with every rising edge on the “Trg” input on the block, specified mathematical operation is performed.

Write on Input: If this option is selected, the value at the “inA” input of the block and the value at the “inB” input of the block is processed. Result of the operation is written on the “inA” input of the block. A WORD register should be connected to the inA input of the block. This operation is performed at each PLC cycle by default.

If “On When Trig is Active” option is selected, this operation is performed at each rising edge detected on the “Trg” input of the block.

Math Types and Explanations:

Math	**Used Inputs**	**Explanation**
ADDITION (+)	inA, inB	The values at the “inA” and the “inB” input are added and the result is written to the “#WMat0” output of the block. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input.
SUBTRACTION (-)	inA, inB	The values at the “inA” and the “inB” input are subtracted and the result is written to the “#WMat0” output of the block. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input.
MULTIPLICATION (*)	inA, inB	The values at the “inA” and the “inB” input are multiplied and the result is written to the “#WMat0” output of the block. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input.
DIVISION (/)	inA, inB	The value at the “inA” is divided to the value at the “inB” and the result is written to the “#WMat0” output of the block. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input.
AND	inA, inB	The values at the “inA” and the “inB” input are bitwise ANDed and the result is written to the “#WMat0” output of the block. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input.
OR	inA, inB	The values at the “inA” and the “inB” input are bitwise ORed and the result is written to the “#WMat0” output of the block. If “Write on Input” is selected, the result is written to the“WMat0” output of the block and the “inA” input.
XOR	inA, inB	The values at the “inA” and the “inB” input are bitwise XOR and the result is written to the “#WMat0” output of the block. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input.
SHIFT LEFT	inA, inB	The bits of the value at the “inA” input are shifted left by the value at the “inB” and the result is written to the “#WMat0” output of the block. . If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input. (Ex: inA =1110b, inB=1 then; #WMat0=1100b)

SHIFT RIGHT	inA, inB	The bits of the value at the "inA" input are shifted right by the value at the "inB" and the result is written to the "#WMat0" output of the block. If "Write on Input" is selected, the result is written to the "WMat0" output of the block and the "inA" input. (Ex: inA=1110b, inB=1 then; #WMat0=0111b)
CHECK BIT	inA, inB	The bit of the value at the "inA" is checked and written to the "#WMat0" output of the block where n is specified by the "inB" input of the block. "inB" must be between 0-15. If "Write on Input" is selected, the result is written to the "WMat0" output of the block and the "inA" input. (Ex: inA=1**1**10, inB=2 then; #WMat0=1)
LEFTSHIFTCHECKFIRST	inA, inB	0th bit of the value at the "inA" is checked and written to the "#Wat0" output of the block. The bits of the value at the "inA" is shifted left by the value at the "inB" input of the block and written to the output "#WMat0" of the block. If "Write on Input" is selected, the result is written to the "inA" input.
RIGHTSHIFTCHECKFIRST	inA, inB	0th bit of the value at the "inA" is checked and written to the "#WMat0" output of the block. The bits of the value at the "inA" is shifted right by the value at the "inB" input of the block and written to the output "#WMat0" of the block. If "Write on Input" is selected, the result is written to the "inA" input.
LEFTSHIFTCHECKLAST	inA, inB	15th bit of the value at the "inA" is checked and written to the "#WMat0" output of the block. The bits of the value at the "inA" is shifted left by the value at the "inB" input of the block and written to the output "#WMat0" of the block. If "Write on Input" is selected, the result is written to the "inA" input.
RIGHTSHIFTCHECKLAST	inA, inB	15th bit of the value at the "inA" is checked and written to the "#WMat0" output of the block. The bits of the value at the "inA" is shifted right by the value at the "inB" input of the block and written to the output "#WMat0" of the block. If "Write on Input" is selected, the result is written to the "inA" input.
ABSOLUTE VALUE	inA	The absolute value of the value at the "inA" is written to the "#WMat0" output of the block. If "Write on Input" is selected, the result is written to the "WMat0" output of the block and the "inA" input.
COMPARE BIT	inA, inB	The bits of the values at the "inA" and the "inB" inputs of the block are compared starting from the left and one more of the value of the first different bit position is written to the "#WMat0" output of the block. If all the bits are the same, 0 is written to the "#WMat0" output. If "Write on Input" is selected, the result is written to the "WMat0" output of the block and the "inA" input. (Ex: If 0th bit is different, 1 is written to the #WMat0.)
MOD	inA, inB	Modular arithmetic operation. Mod(inB) of the value at the "inA" is written to the "#WMat0" output of the block. The value at the "inA" is divided by the value at the "inB" and the remainder is written to the "#WMat0" output. If "Write on Input" is selected, the result is written to the "WMat0" output of the block and the "inA" input. (Ex: inA = 253, inB = 10 then O1 = 4)

BIT REPLACE	inA, inB, INB	It is used to set the index of the “inA” block input value bits in the “INB” value in the block options section to 0 or 1. The bit value to be written is determined by the “inB” block input value. The result of the operation is written to the “#WMat0” output. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input.
GET	inB	It is used for reading a Word Register block’s or a block’s value present in the logic project. The block number to be read is specified with “inB” input of the block. Read value is written to output “#WMat0”. If “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input. It is also used for some special commands. These commands can be seen in diagram below.
LOW LIMIT	inA, inB	Specifies the minimum value that “#WMat0” output can take. Desired minimum value is written to the “inA” input. If “inB” has a greater value than “inA” input, the value at the “inB” is written to the “#WMat0” output. Otherwise, the value at the “inA” is written to the “#WMat0” output. “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input. (Ex: inA = 10, inB = 8 then; #WMat0 = 10)
HIGH LIMIT	inA, inB	Specifies the maximum value that “#WMat0” output can take. Desired maximum value is written to the “inA” input. If “inB” has a smaller value than “inA” input, the value at the “inB” is written to the “#WMat0” output. Otherwise, the value at the “inA” is written to the “#WMat0” output. “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input. (Ex: inA = 10, inB = 12 then; #WMat = 10)
MERGE A-B	inA, inB	The value at the “inB” is shifted left by 8 bits and added to the value at the “inA”. “Write on Input” is selected, the result is written to the “WMat0” output of the block and the “inA” input. (Two of the 8 bit merge blocks can be used for 16 bit merging.)
SET	inA, inB	It is used for write to a Word Register block or to a block present in the logic project. “inA” block input value is the value to be written. The block number to be written is specified with inB input of the block. The “inA” block input value is written both to the “#WMat0” block output and to the block to be written. (Ex: inA = 10, inB = 3001 then; 10 is written to the block which has block number 3001.)

5.4.4.1 GET Operation Special Commands

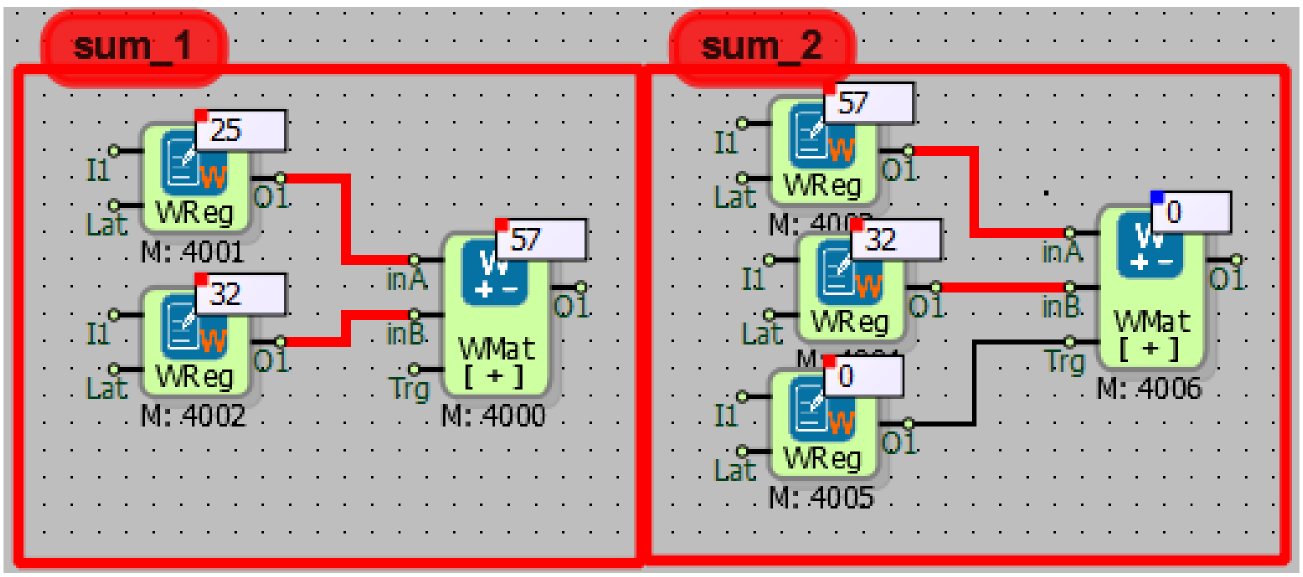
When performing GET operation, if some special values are entered to the “inB” input of the block then some special operations are performed by the block.

Diagram below shows the commands and the related operations to the commands.

inB Value	Function Explanation
20000	Resets the device using software.
31000	Sends the value at the “inA”as DTMF code. (Only available for GSM devices.)

5.4.5 Sample Application

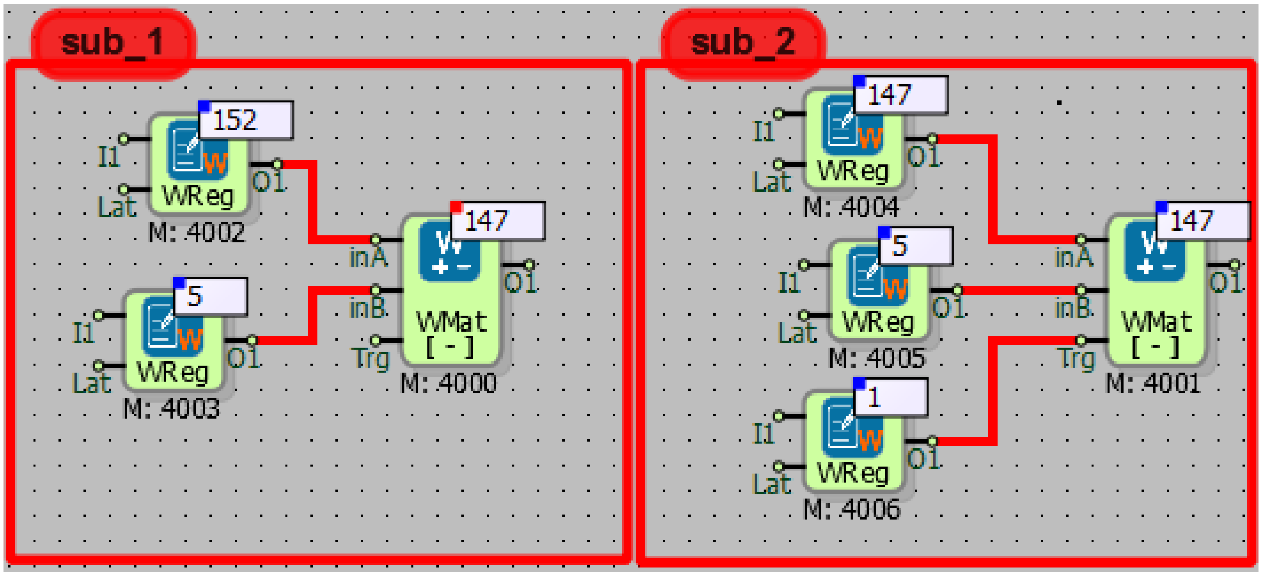
Addition examples:



In sum_1 example, the values at the “inA” and “inB” input of the block are added and the result is written to the “O1” output of the block.

In sum_2 example, “On When Trig is Active” and “Write on Input” is selected. Hence, the lues at the “inA” and the “inB” are added and the result is written to the “inA” input at each detected rising edge on the “Trg” input of the block.

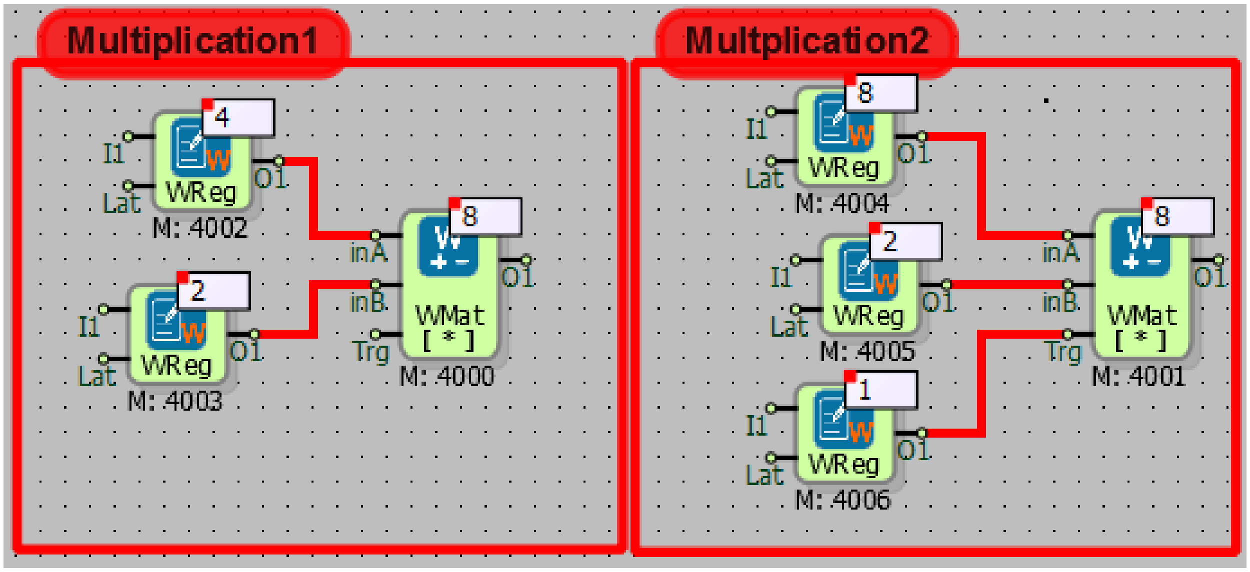
Subtraction examples:



In sub_1 example, the value at the “inA” of the block is subtracted from the “inB” input of the block and the result is written to the “O1” output of the block.

In sub_2 example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the “inA” of the block is subtracted from the “inB” input of the block and the result is written to the “inA” input at each detected rising edge on the “Trg” input of the block.

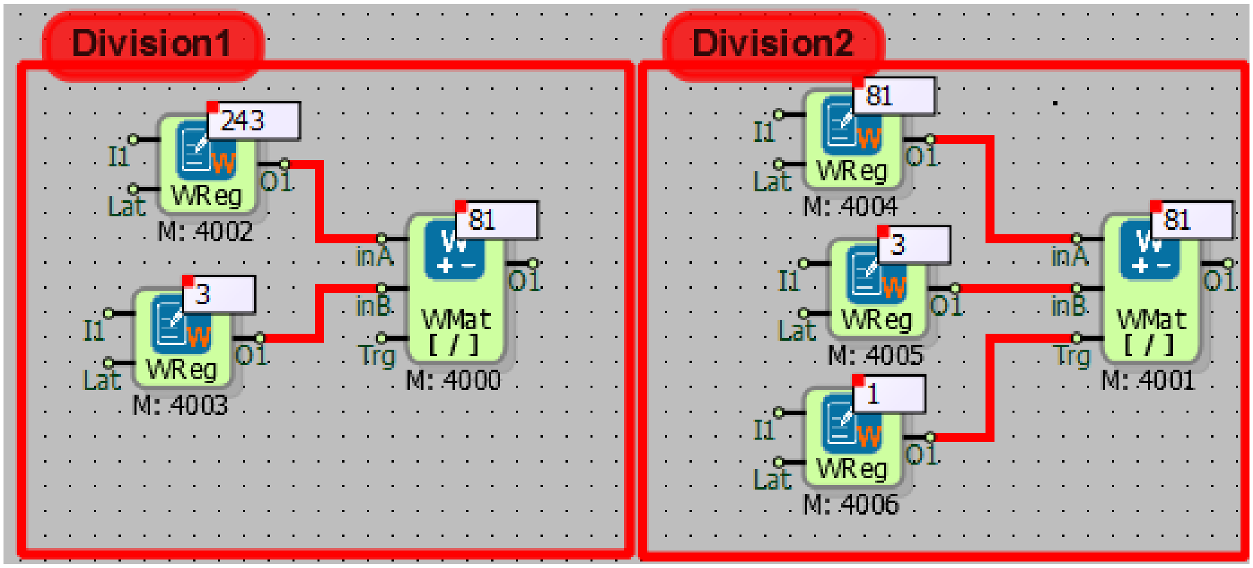
Multiplication examples;



In Multiplication1 example, the value at the “inA” input of the block is multiplied by the “inB” input of the block and the result is written to the “O1” output of the block.

In Multiplication2 example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the “inA” of the block is multiplied by the “inB” input of the block and the result is written to the “inA” input at each detected rising edge on the “Trg” input of the block.

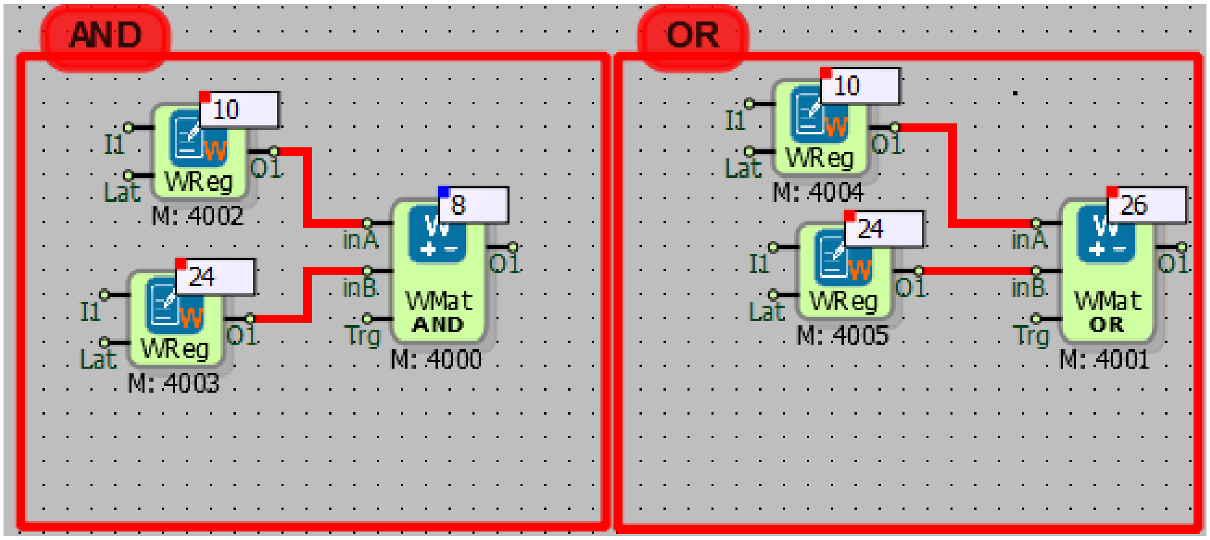
Division examples;



In Division1 example, the value at the “inA” input of the block is divided by the “inB” input of the block and the result is written to the “O1” output of the block.

In Division2 example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the “inA” of the block is divided by the “inB” input of the block and the result is written to the “inA” input at each detected rising edge on the “Trg” input of the block.

AND and OR examples:



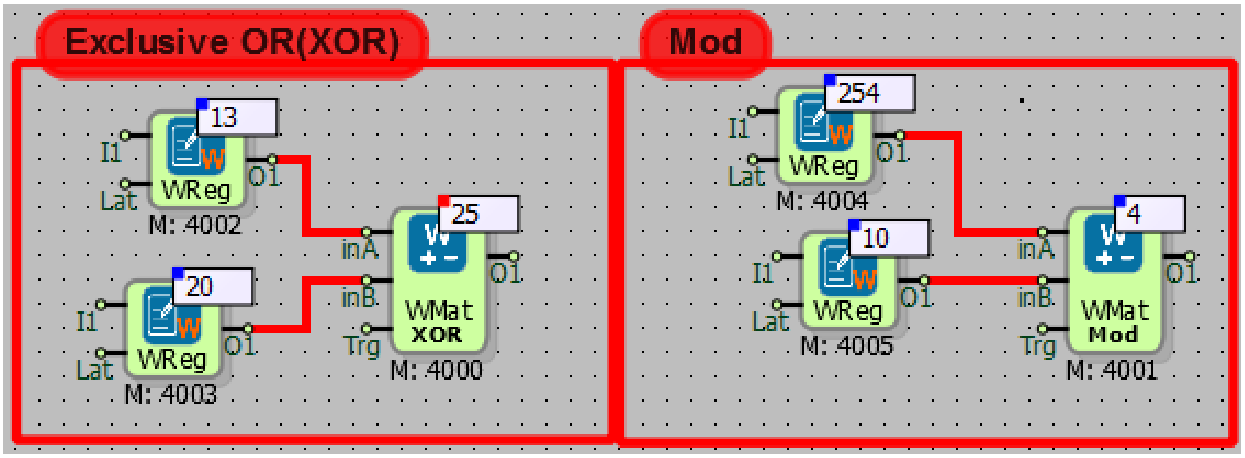
Corresponding binary value of the decimal value at the inA: $(10)_{10}$=$(01010)_2$

Corresponding binary value of the decimal value at the inB: $(24)_{10=}$ $(11000)_2$

The result of bitwise AND operation between inA and inB is: $(8)_{10}$=$(01000)_2$

The result of bitwise OR operation between inA and inB is: $(26)_{10}$=$(11010)_2$

Exclusive OR(XOR) and Mod examples;



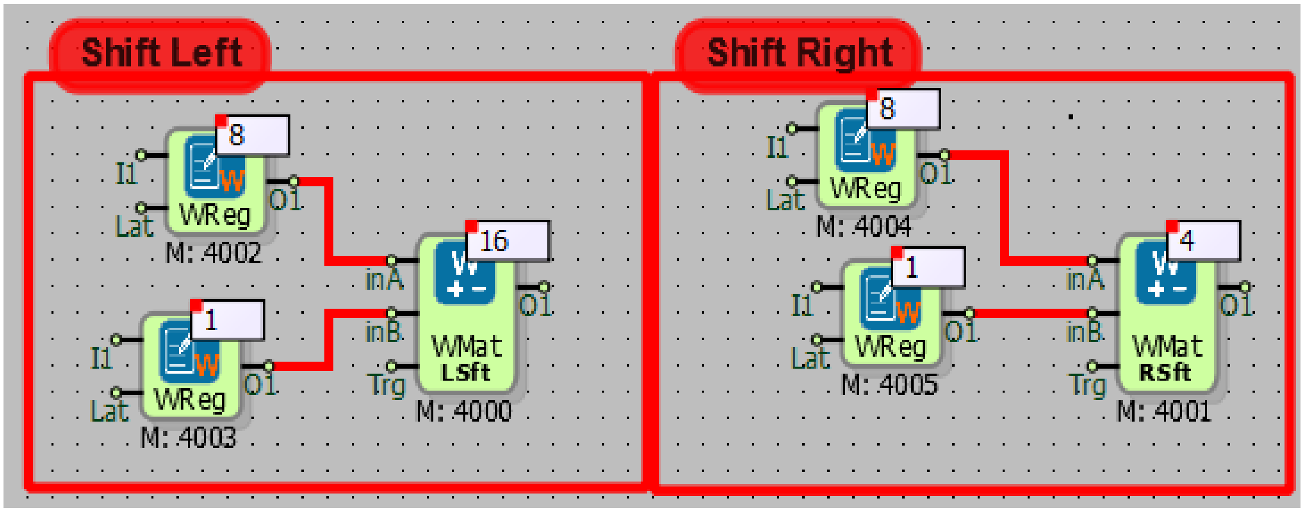
Corresponding binary value of the decimal value at the inA: $(13)_{10}=(01101)_2$

Corresponding binary value of the decimal value at the inB: $(20)_{10=}$ $(10100)_2$

The result of bitwise XOR operation between inA and inB is: $(25)_{10}=(11001)_2$

In Mod example, the value at the “inA” is divided by the value at “inB” and the remainder of the operation is written to the “O1” output of the block.

Shift Left and Shift Right examples:



Corresponding binary value of the decimal value at the inA: $(8)_{10}=(0\mathbf{1}000)_2$

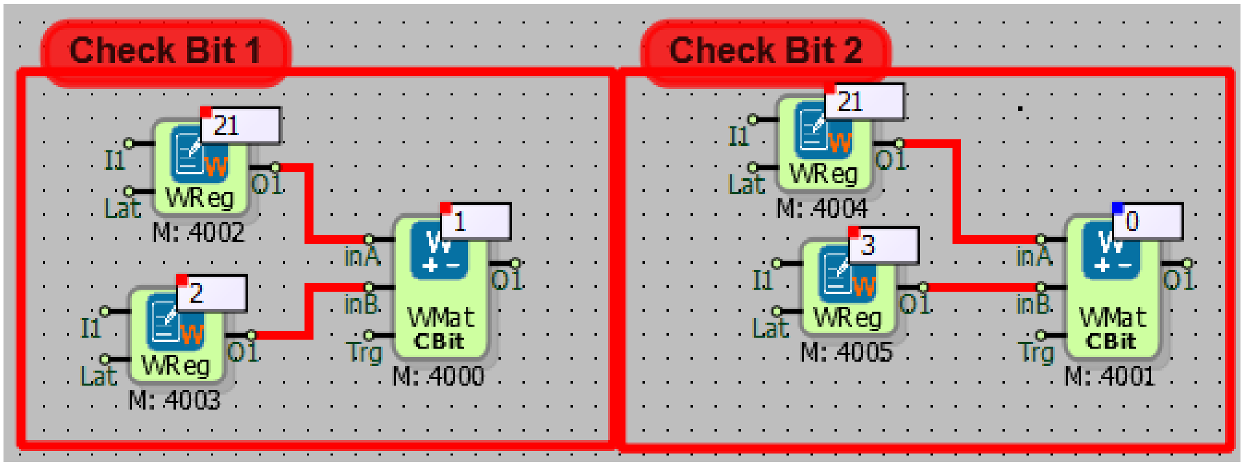
The value at the “inB” input specifies the number of bits which “inA” is going to be shifted by.

After the shifting operation, result is written to the “O1” output of the block.

Shift Left: When 8 is shifted left by 1: $(16)_{10}=(\mathbf{1}0000)_2$ is obtained.

Shift Right: When 8 is shifted right by 1: $(4)_{10}=(00\mathbf{1}00)_2$ is obtained.

Check Bit examples;



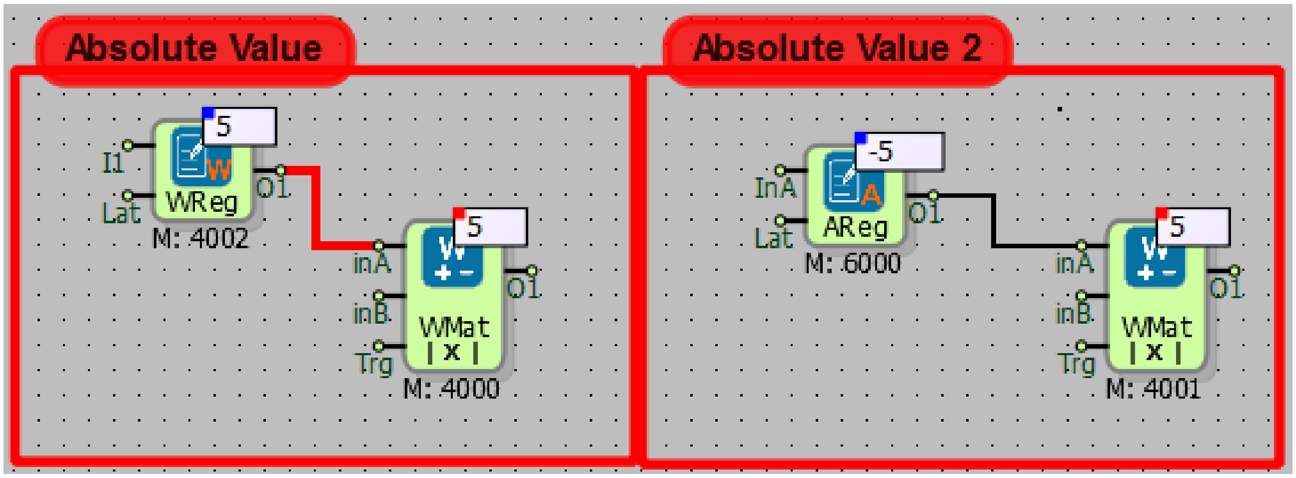
Corresponding binary value of the decimal value at the inA: $(21)_{10}=(10101)_2$

The value at the “inB” input specifies the index of the bit which is going to be checked. After the checking process, checked bit is written to the “O1” output of the block.

In Check Bit 1 example, the value of the checked bit is $(10\underline{\mathbf{1}}01)_2$:1.

In Check Bit 2 example, the value of the checked bit is $(1\underline{\mathbf{0}}101)_2$:0.

Absolute value examples:

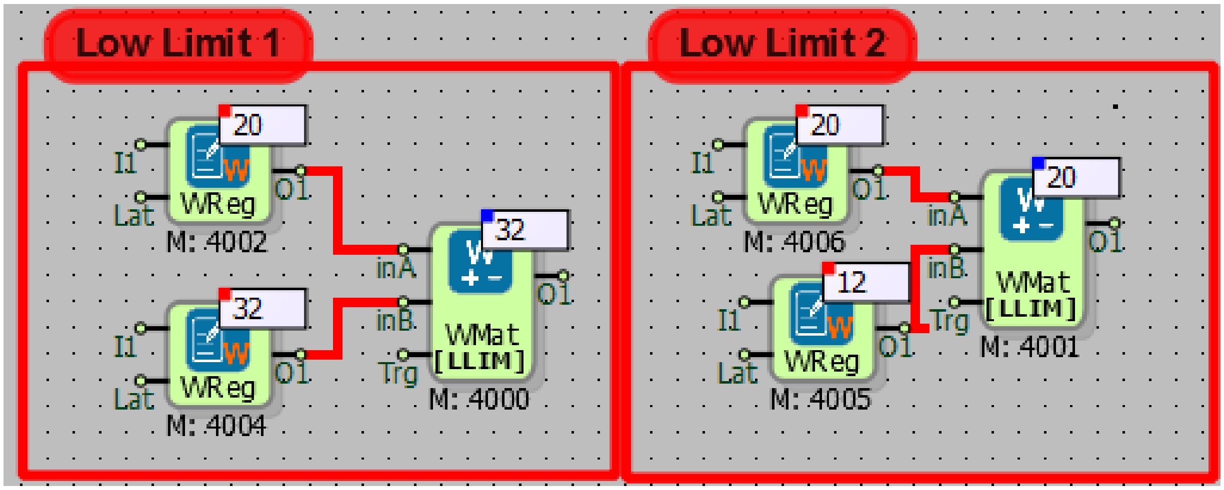


Distance of the value at the inA to the origin is written to the O1 output.

In “AbsoluteValue1” example, distance of 5 to the origin is 5.

In “AbsoluteValue2” example, distance of -5’ to the origin is 5.

Low Limit examples:

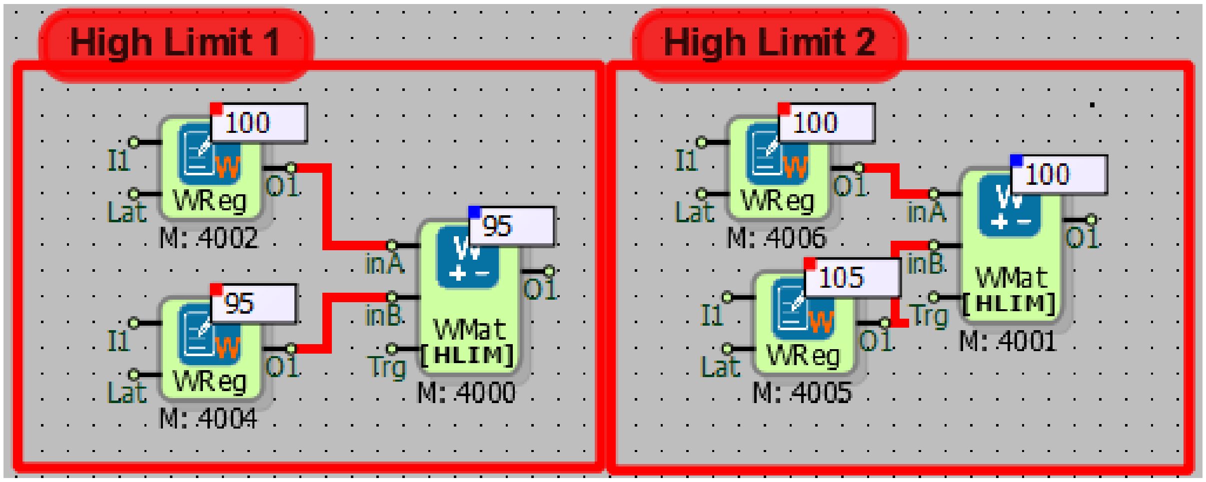


Low limit value is connected to the “inA” input of the block using a WORD register.

In Low Limit 1 example, low limit is not activated. Since the value at the “inB” input is greater than the low limit, the value at the “inB” is written to the “O1” output of the block.

In Low Limit 2 example, low limit is activated. Since the value at the “inB” input is smaller than the low limit, the value at the “inA” is written to the “O1” output of the block.

High limit examples:

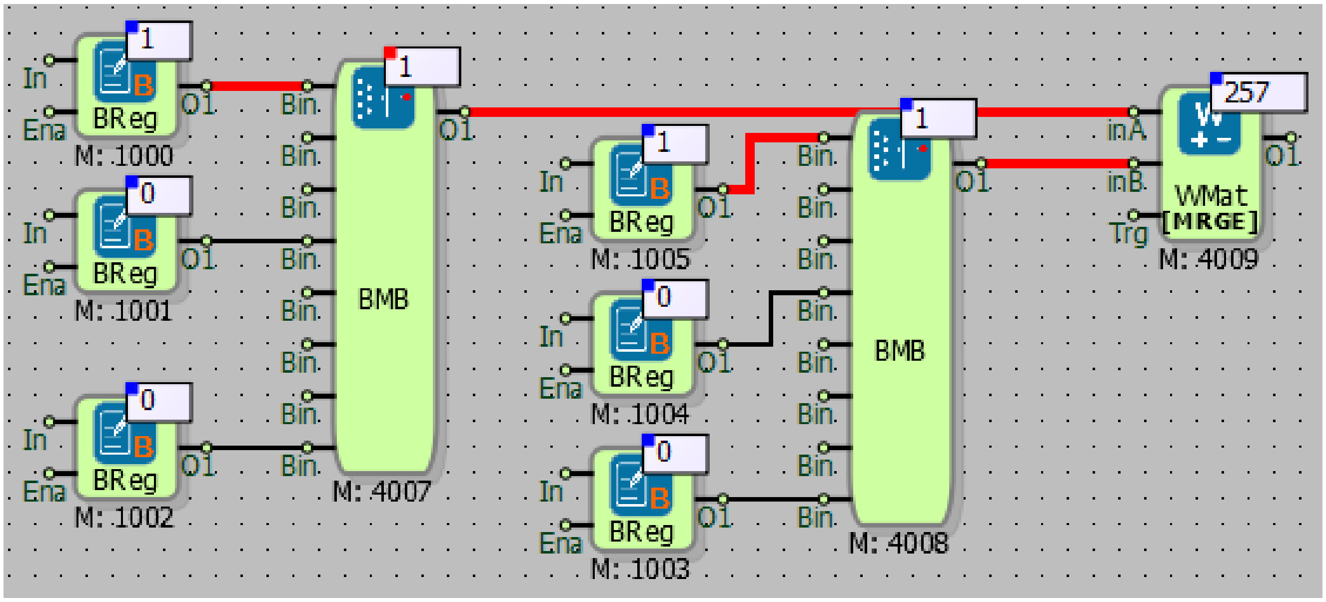


High limit value is connected to the “inA” input of the block using a WORD register.

In High Limit 1 example, high limit is not activated. Since the value at the “inB” input is smaller than the low limit, the value at the “inB” is written to the “O1” output of the block.

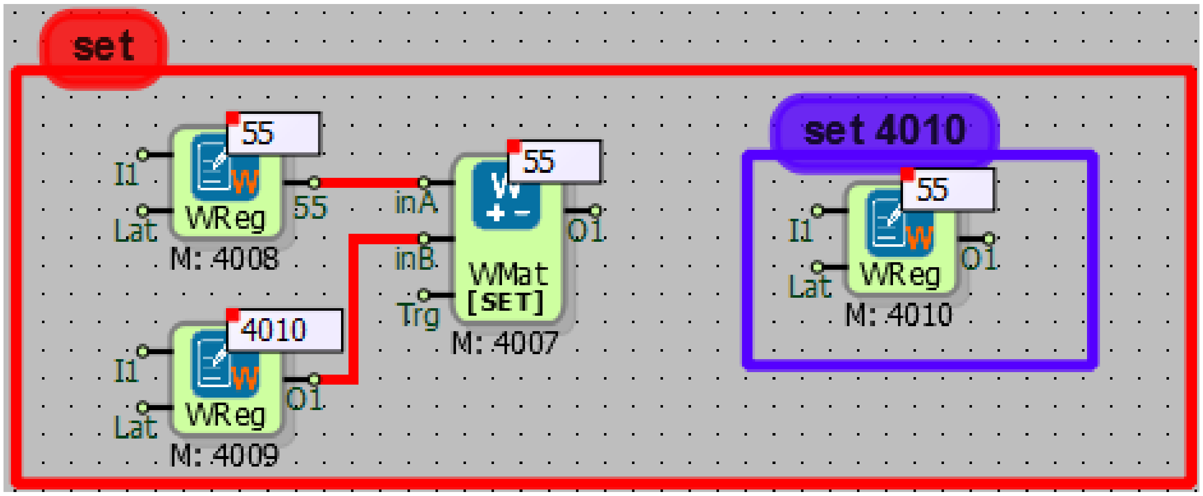
In High Limit 2 example, high limit is activated. Since the value at the “inB” input is greater than the low limit, the value at the “inA” is written to the “O1” output of the block.

Merge A-B example:



An 8 bit merge block is connected to “inA” input of the block and an other 8 bit merge block is connected to “inB” input of the block. The value at the “inB” block is shifted left by 8 bits and added to the value at the “inA” input of the block. That way, a merge 16 bit merge block is designed with 0-8 bits are connected to “inA” input and 9-15 bits are connected to “inB” input.

Set example;



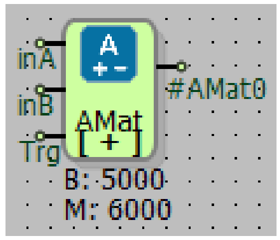
The value to be set is connected to the “inA” input of the block.

Number of the target block is connected to the “inB” input of the block.

The value at the “inA” input, 55, is set to the block with number 4010.

5.5 ANALOG MATH

5.5.1 Connections

inA: Analog data input		#AMat0: Output of the Block
inB : Analog data input		
Trg: Trigger input		

5.5.2 Connection Explanations

inA: Analog data input

Analog value to be processed.

inB: Analog data input

Analog value to be processed.

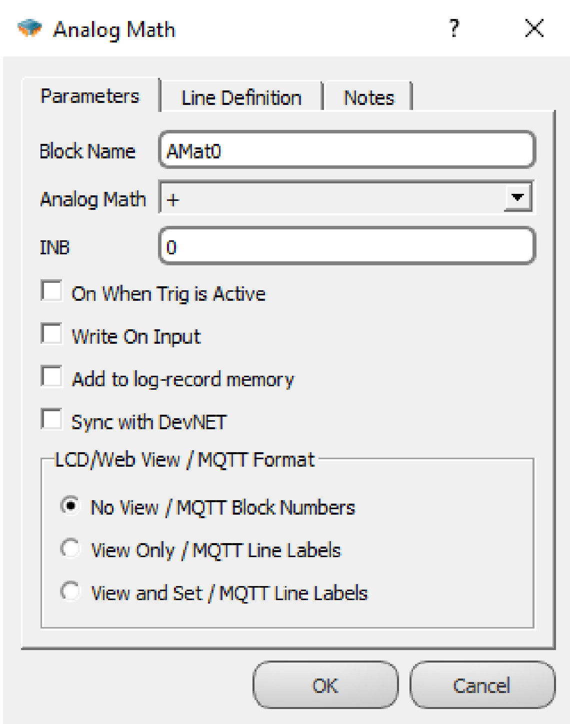
Trg: Trigger input

If the “On When Trig is Active” is selected in Block Settings menu, block is activated at each rising edge detected at the Trg input of the block.

#AMat0: Block of the Output

32-bit floating point output of the block.

5.5.3 Block Settings

	Analog Math Type: Mathematical operation is specifed here
	INB: Analog input to be processed.
	On When Trig is Active: If selected, block is activated at each rising edge detected at the Trg input of the block
	Write On Input: If this option is selected, the value at the inA input of the block and the value at the inB input of the block is processed. Result of the operation is written on the inA input of the block. An Analog register should be connected to the inA input of the block. This operation is performed at each PLC cycle by default. If “On When Trig is Active” option is selected, this operation is performed at each rising edge detected on the Trig input of the block.

5.5.4 Block Explanation

It is used for IEE754 floating point number mathematical operations. With Analog Math block, “addition”, “subtraction”, “multiplication”, “division”, “absolute value”, “square root”, “sin”, “cos”, “tan”, “asin”, “acos”, “atan1”, “atan2”, “get”, “low limit”, “high limit”, “set” and “Word to Signed” operations can be performed.

On When Trig is Active: If this option is selected, with every rising edge on the Trig input of the block, specified mathematical operation is performed.

Write on Input: If this option is selected, the value at the “inA” input of the block and the value at the “inB” input of the block is processed. Result of the operation is written on the “inA” input of the block. An Analog Register block should be connected to the “inA” input of the block. This operation is performed at each PLC cycle by default. If “On When Trig is Active” option is selected, this operation is performed at each rising edge detected on the “Trg” input of the block.

Math Types and Explanations:

Math	Used Inputs	Explanation
ADDITION (+)	inA, inB	The values at the “inA” and the “inB” input are added and the result is written to the “#AMat0” output of the block. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
SUBTRACTION (-)	inA, inB	The values at the inA and the inB input are subtracted and the result is written to the “#AMat0” output of the block. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
MULTIPLICATION (*)	inA, inB	The value at the “inA” input of the block is multiplied by the “inB” input of the block and the result is written to the “#AMat0” output of the block. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the inA input.
DIVISION (/)	inA, inB	The value at the “inA” input of the block is divided by the “inB” input of the block and the result is written to the “#AMat0” output of the block. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
ABSOLUTE VALUE	inA	The absolute value of the value at the “inA” is written to the “#AMat0” output of the block. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input. (Ex: inA=-15 then; #AMat0=15)
SQUARE ROOT	inA	Takes the square root of the value at the “inA” input and the result is written to the “#AMat0” output of the block. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input. (Ex: inA=81 then; O1=9)
SIN	inA	Trigonometric sine function Sin(inA). The result is written to the "#AIsm0" block output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
COS	inA	Trigonometric cosine function Cos(inA). The result is written to the "#AIsm0" block output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
TAN	inA	Trigonometric tangent function Tan(inA). The result is written to the "#AIsm0" block output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
ASIN	inA	Trigonometric arcsine function Asin(inA). The result is written to the "#AIsm0" block output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
ACOS	inA	Trigonometric arccosine function Acos(inA). The result is written to the "#AIsm0" block output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
ATAN1	inA	Trigonometric arctangent function Atan(inA). The result is written to the "#AIsm0" block output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input.
ATAN2	inA, inB	Trigonometric arctangent (inB/ inA) function Atan2(inA, inB). The result is written to the "#AIsm0" block output. If “Write on Input” is selected,

		the result is written to the “#AMat0” output of the block and the “inA” input.
GET	inA, inB	It is used for reading a Word Register block’s or a block’s value present in the logic project. The block number to be read is specified with “inB” input of the block. The read value is written to the "#AIsm0" block output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input. It is also used for some special commands. These commands can be seen in diagram below.
LOW LIMIT	inA, inB	Specifies the minimum value that “#AMat0” output can take. Desired minimum value is written to the “inA” input. If “inB” has a greater value than “inA” input, the value at the “inB” is written to the “#AMat0” output. Otherwise, the value at the “inA” is written to the “#AMat0” output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input. (Ex: inA = 10, inB = 8 then; O1 = 10)
HIGH LIMIT	inA, inB	Specifies the maximum value that “#AMat0” output can take. Desired maximum value is written to the “inA” input. If “inB” has a smaller value than “inA” input, the value at the “inB” is written to the “#AMat0” output. Otherwise, the value at the “inA” is written to the “#AMat0” output. If “Write on Input” is selected, the result is written to the “#AMat0” output of the block and the “inA” input. (Ex: inA = 10, inB = 12 then; O1 = 10)
SET	inA, inB	It is used for write to a Word Register block’s or to a block present in the logic project. “inA” block input value is the value to be written. The block number to be written is specified with inB input of the block. The “inA” block input value is written both to the “#AMat0” block output and to the block to be written. (Ex: inA = 10, inB = 3001 then; 10 is written to the block which has block number 3001.)
WORD TO SIGNED	inA	A Word Register block containing 16-bit unsigned number is connected to “inA” input of the block and converted to the 16-bit signed number and written to the “#AMat0” output of the block. (Ex: inA=65535 then; #AMat0=-1, inA=65534 then; #AMat0=-2 inA=32768 then; #AMat0=-32768, inA=32769 then; #AMat0=-32767, inA=1 then; #AMat0=1, inA=32766 then; #AMat0=32766, inA=32767 then; #AMat0=32767))

5.5.4.1 GET Operation Special Commands

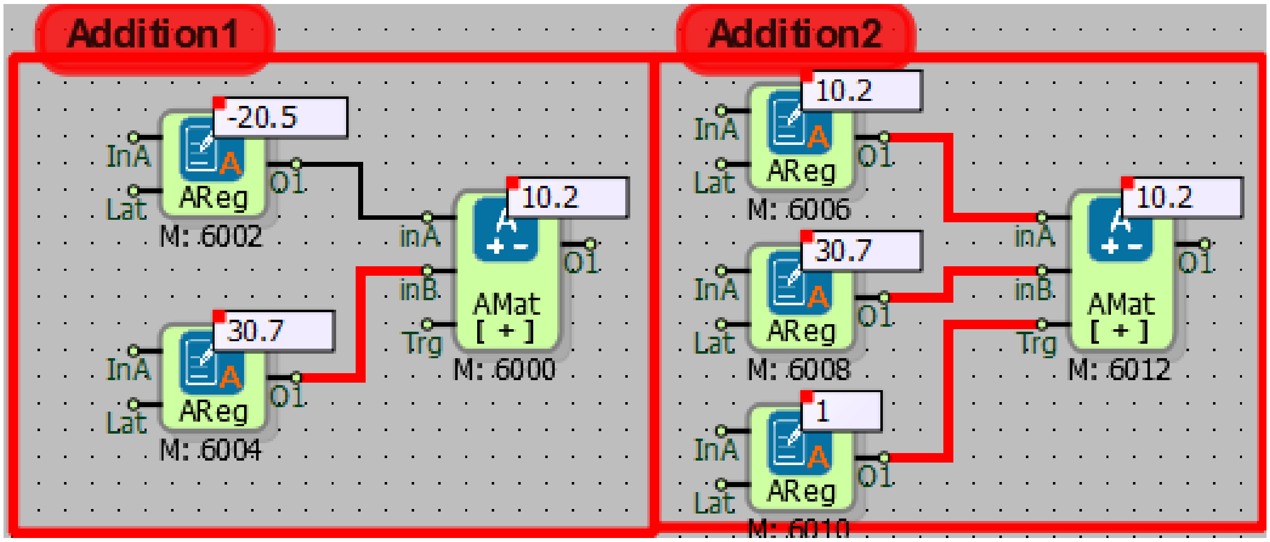
When performing GET operation, if some special values are entered to the “inB” input of the block then some special operations are performed by the block.

Diagram below shows the commands and the related operations to the commands

inB Value	Function Explanation
10000	Reads the temperature value from the integrated temperature sensor SHT21. Aavilable only for devices that have the integrated temperature sensor.
10001	Reads the humidity value from the integrated temperature sensor SHT21. Aavilable only for devices that have the integrated humidity sensor.
20000	Reads the RMC geographic latitude data from GPS.
20001	Reads the RMC geographic longitude data from GPS.
20002	Reads the geographic speed data from GPS.(km/h)
20003	Reads the GLL geographic latitude data from GPS.
20004	Reads the GLL geographic longitude data from GPS.
20005	Reads the HEH degree data from GPS.
30001	Real time clock, VBAT – Battery voltage in Volts

5.5.5 Sample Application

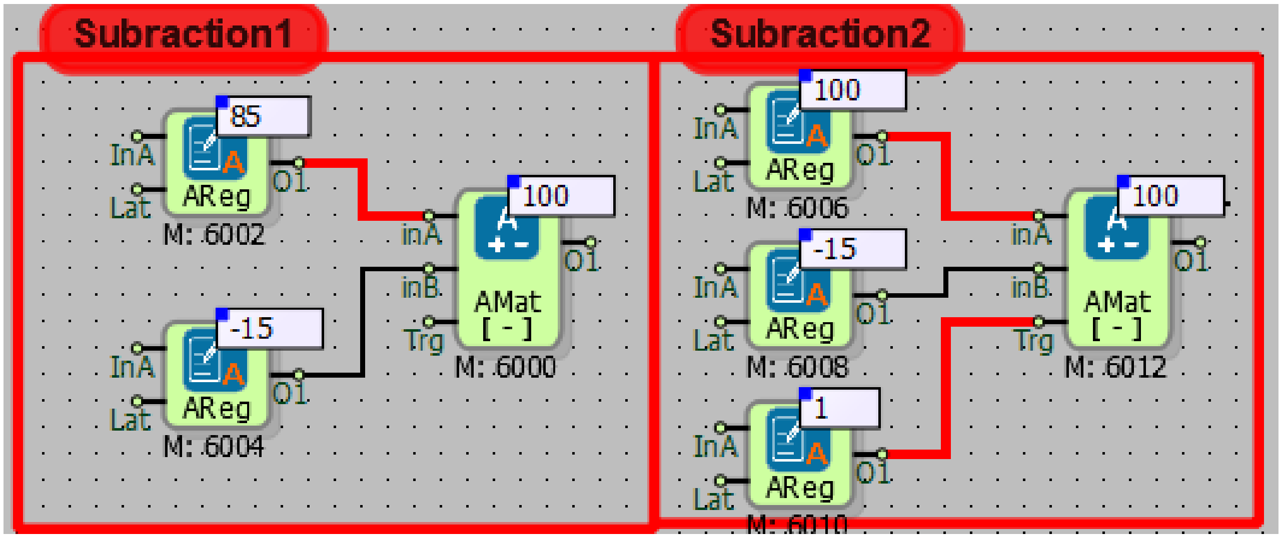
Addition examples:



In "Addition1" example, the values at the inA and inB input of the block are added and the result is written to the O1 output of the block.

In "Addition2" example, "On When Trig is Active" and "Write on Input" is selected. Hence, the values at the inA and the inB are added and the result is written to the inA input at each detected rising edge on the Trig input of the block.

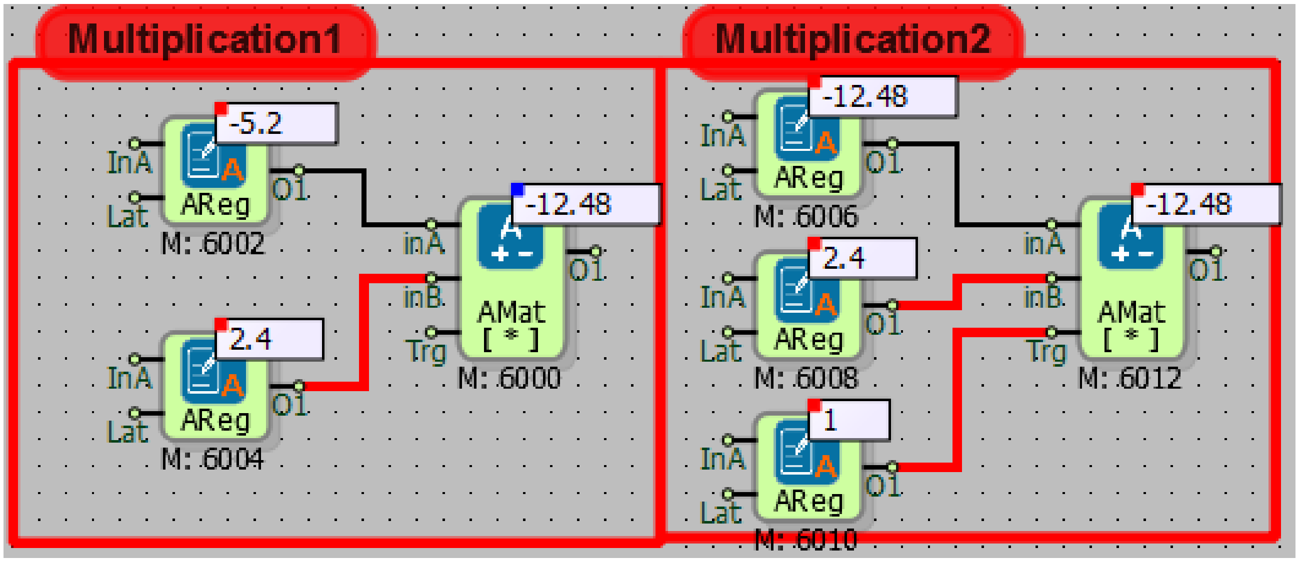
Subtraction examples:



In “Subtraction1” example, the value at the inA of the block is subtracted from the inB input of the block and the result is written to the O1 output of the block.

In “Subtraction2” example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the inA of the block is subtracted from the inB input of the block and the result is written to the inA input at each detected rising edge on the Trig input of the block.

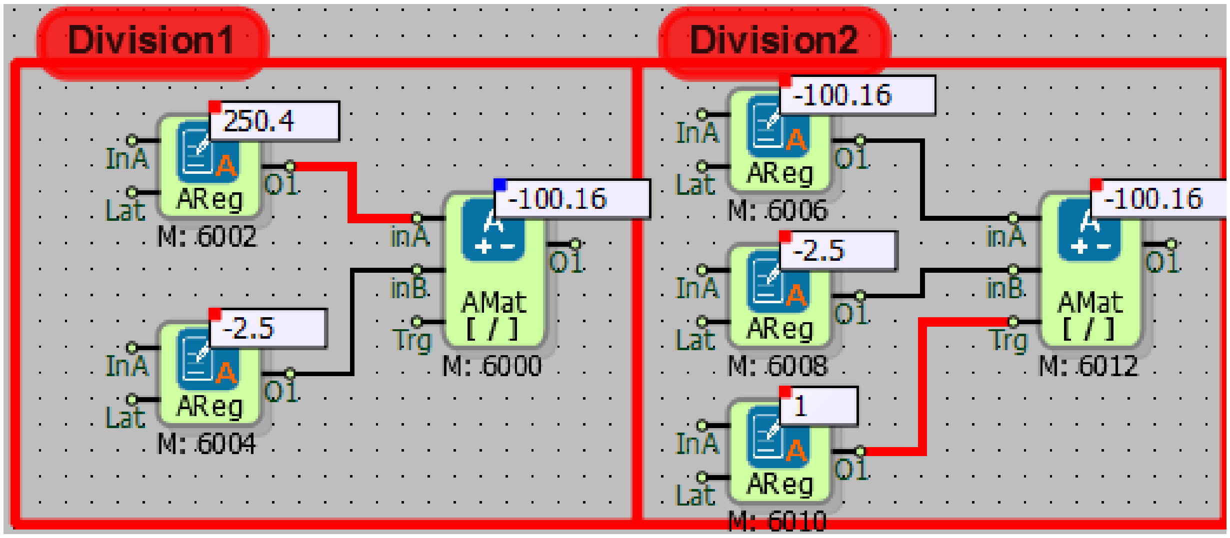
Multiplication examples:



In “Multiplication1” example, the value at the inA input of the block is multiplied by the inB input of the block and the result is written to the O1 output of the block.

In “Multiplication2” example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the inA of the block is multiplied by the inB input of the block and the result is written to the inA input at each detected rising edge on the Trig input of the block.

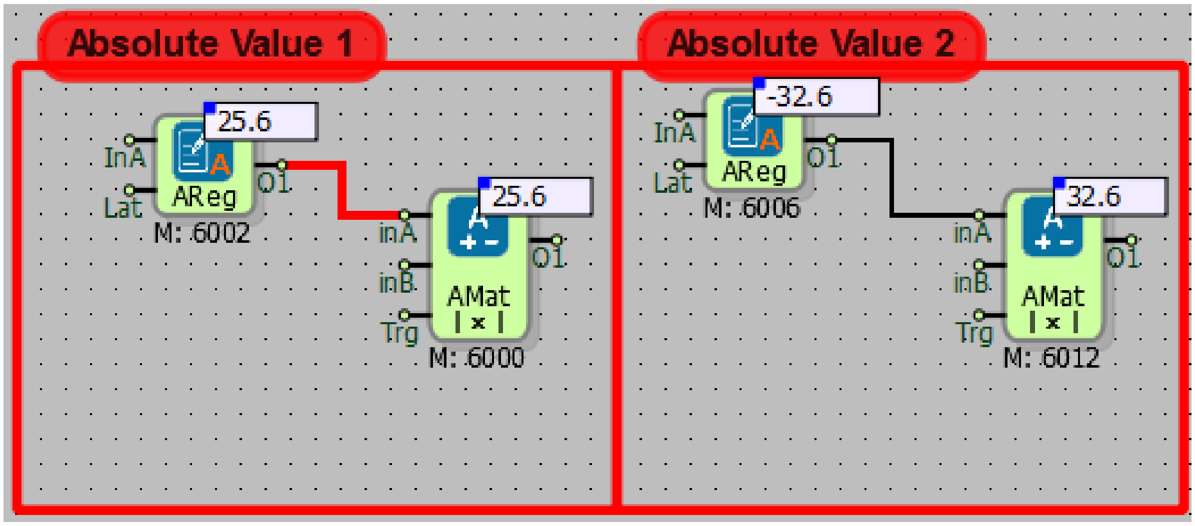
Division examples:



In “Division1” example, the value at the inA input of the block is divided by the inB input of the block and the result is written to the O1 output of the block.

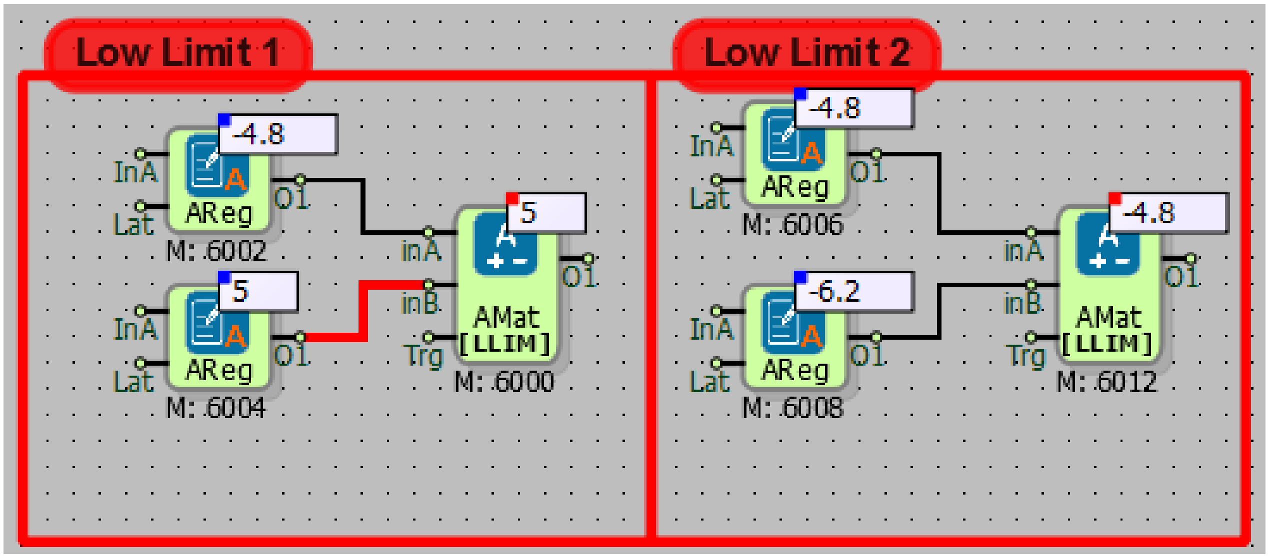
In “Division2” example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the inA of the block is divided by the inB input of the block and the result is written to the inA input at each detected rising edge on the Trig input of the block.

Absolute value examples:



Distance of the value at the inA to the origin is written to the O1 output. In “AbsoluteValue1” example, distance of 25.6 to the origin is 25.6. In “AbsoluteValue2” example, distance of -32.6 to the origin is 32.6.

Low limit examples:

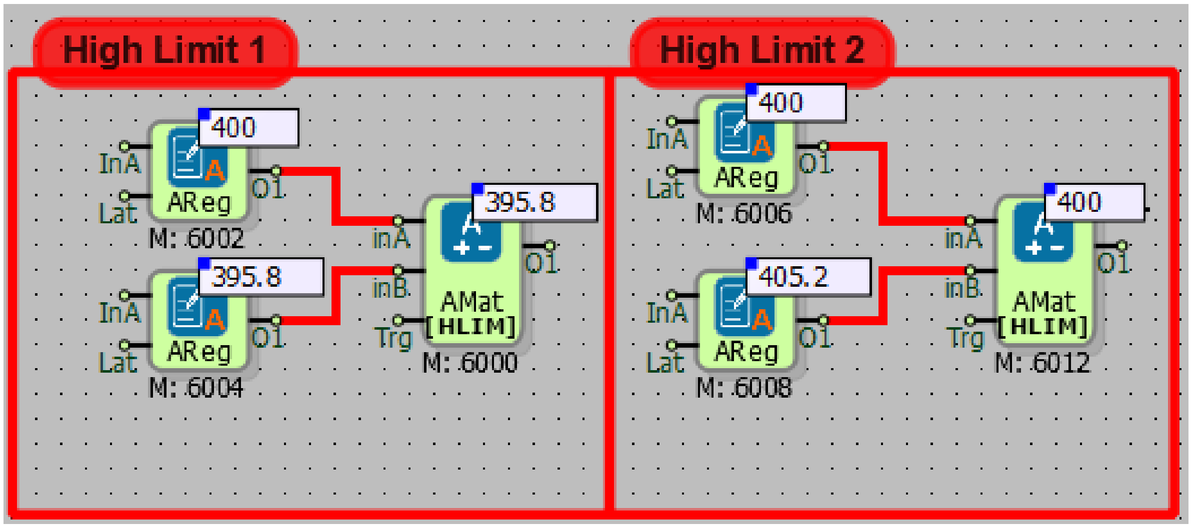


Low limit value is connected to the inA input of the block using a WORD register.

In “Low Limit 1” example, low limit is not activated. Since the value at the inB input is greater than the low limit, the value at the inB is written to the O1 output of the block.

In “Low Limit 2” example, low limit is activated. Since the value at the inB input is smaller than the low limit, the value at the inA is written to the O1 output of the block.

High limit examples:

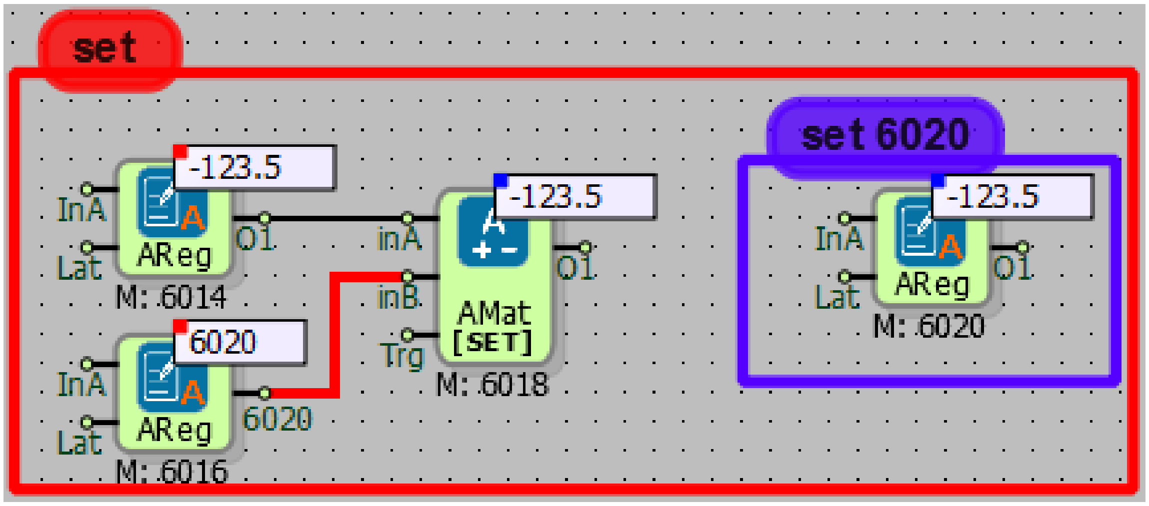


High limit value is connected to the inA input of the block using a WORD register.

In “High Limit 1” example, high limit is not activated. Since the value at the inB input is smaller than the low limit, the value at the inB is written to the O1 output of the block.

In “High Limit 2” example, high limit is activated. Since the value at the inB input is greater than the low limit, the value at the inA is written to the O1 output of the block.

Set example:



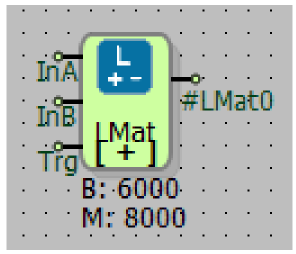
The value to be set is connected to the inA input of the block.

Number of the target block is connected to the inB input of the block.

The value at the inA input, -123.5, is set to the block with number 6020.

5.6 LONG MATH

5.6.1 Connections

InA: Long data input		#LMat0: Output of the Block
InB : Long data input		
Trg: Trigger input		

5.6.2 Connection Explanations

<u>InA: Long data input</u>

Long value to be processed.

<u>InB : Long data input</u>

Long value to be processed.

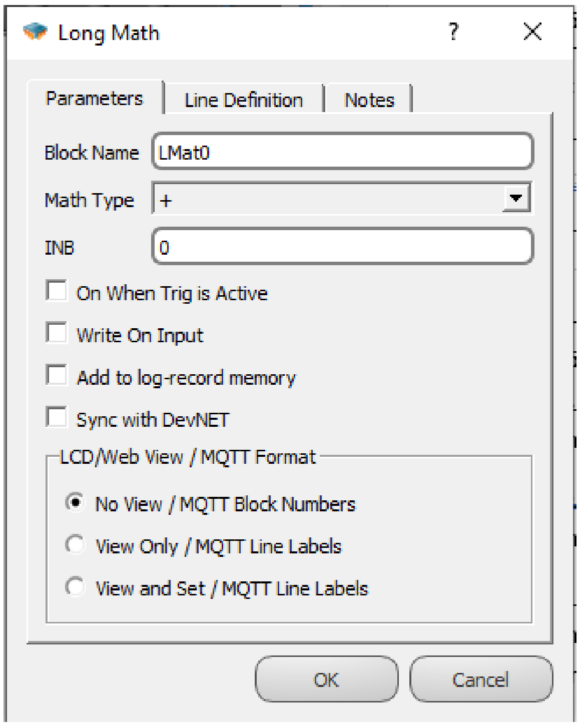
<u>Trg: Trigger input</u>

If the "On When Trig is Active" is selected in Block Settings menu, block is activated at each rising edge detected at the Trg input of the block.

<u>#LMat0: Output of the Block</u>

32-bit signed output of the block.

5.6.3 Block Settings



Math Type: Mathematical operation is specifed here
INB: Second long input to be processed can be entered in Block Settings menu.
On When Trig is Active: If selected, block is activated at each rising edge detected at the Trg input of the block
Write On Input: If this option is selected, the value at the inA input of the block and the value at the inB input of the block is processed. Result of the operation is written on the inA input of the block. A long register should be connected to the inA input of the block. This operation is performed at each PLC cycle by default. If “On When Trig is Active” option is selected, this operation is performed at each rising edge detected on the Trig input of the block.

5.6.4 Block Explanation

It is used for mathematical operations which result in 32-bit signed integers. With Long Math block “addition”, “subtraction”, “multiplication”, “division”, “logic AND”, “logic OR”, “logic XOR”, “shift left”, “shift right”, “checkBit”, “LeftShiftCheckFirst”, “RightShiftCheckFirst”, “LeftShiftCheckLast”, “RightShiftCheckLast”, “absolute value”, “bit compare”, “mod”, “bit replace”, “get”, “low limit”, “high limit”, “merge A-B”, “WORD to signed” and “set” operations can be performed.

On When Trig is Active: If this option is selected, with every rising edge on the “Trg” input of the block, specified mathematical operation is performed.

Write on Input: If this option is selected, the value at the “inA” input of the block and the value at the “inB” input of the block is processed. Result of the operation is written on the “inA” input of the block. An Analog Register block should be connected to the “inA” input of the block. This operation is performed at each PLC cycle by default. If “On When Trig is Active” option is selected, this operation is performed at each rising edge detected on the “Trg” input of the block.

Math Types and Explanations:

Math	Used Inputs	Explanation
ADDITION (+)	InA, InB	The values at the inA and the inB input are added and the result is written to the OUT output of the block. If “Write on Input” is selected, the result is written to the inA input.
SUBTRACTION (-)	InA, InB	The values at the inA and the inB input are subtracted and the result is written to the OUT output of the block. If “Write on Input” is selected, the result is written to the inA input.
MULTIPLICATION(*)	InA, InB	The value at the inA input of the block is multiplied by the inB input of the block and the result is written to the OUT output of the block. If “Write on Input” is selected, the result is written to the inA input.
DIVISION(/)	InA, InB	The value at the inA input of the block is divided by the inB input of the block and the result is written to the OUT output of the block. If “Write on Input” is selected, the result is written to the inA input.
AND	InA, InB	The values at the inA and the inB input are bitwise ANDed and the result is written to the OUT output of the block. If “Write on Input” is selected, the result is written to the inA input. (Ex: InA=0110, InB=1011 then Out=0010)
OR	InA, InB	The values at the inA and the inB input are bitwise ORed and the result is written to the OUT output of the block. If “Write on Input” is selected, the result is written to the inA input. (Ex: InA=0110, InB=0101 then; Out=0111)
XOR	InA, InB	The values at the inA and the inB input are bitwise XORed and the result is written to the OUT output of the block. If “Write on Input” is selected, the result is written to the inA input. (Ex: InA=0101, InB=1001 then; Out=1100)
SHIFT LEFT	InA, InB	The bits of the value at the inA input are shifted left by the value at the inB and the result is written to the OUT output of the block. . If

		"Write on Input" is selected, the result is written to the inA input. (Ex: inA =1110b, inB=1 then; OUT=1100b)
SHIFT RIGHT	InA, InB	The bits of the value at the inA input are shifted right by the value at the inB and the result is written to the OUT output of the block. . If "Write on Input" is selected, the result is written to the inA input. (Ex: inA=1110b, inB=1 then; OUT=0111b)
CHECK BIT	InA, InB	The n'th bit of the value at the inA is checked and written to the OUT output of the block where n is specified by the inB input of the block. inB must be between 0-15. (Ex: inA=1110, inB=2 then; OUT=1)
LEFTSHIFTCHECKFIRST	InA, InB	0th bit of the value at the inA is checked and written to the OUT output of the block. The bits of the value at the inA is shifted left by the value at the inB input of the block and written to the output OUT of the block. If "Write on Input" is selected, the result is written to the inA input.
RIGHTSHIFTCHECKFIRST	InA, InB	0th bit of the value at the inA is checked and written to the OUT output of the block. The bits of the value at the inA is shifted right by the value at the inB input of the block and written to the output OUT of the block. If "Write on Input" is selected, the result is written to the inA input.
LEFTSHIFTCHECKLAST	InA, InB	15th bit of the value at the inA is checked and written to the OUT output of the block. The bits of the value at the inA is shifted left by the value at the inB input of the block and written to the outputOUT of the block. If "Write on Input" is selected, the result is written to the inA input.
RIGHTSHIFTCHECKLAST	InA, InB	15th bit of the value at the inA is checked and written to the OUT output of the block. The bits of the value at the inA is shifted right by the value at the inB input of the block and written to the output OUT of the block. If "Write on Input" is selected, the result is written to the inA input.
ABSOLUTE VALUE	InA	The absolute value of the value at the inA is written to the OUT output of the block. (Ex: InA=-5 then; Out=5 or InA=22 then; Out=22)
COMPARE BIT		The bits of the values at the inA and the inB inputs of the block are compared starting from the left and the first different bits position

		is written to the OUT output of the block. If all the bits are the same, 0 is written to the OUT output. One more of the value of the different bit's index is written to the OUT. (Ex: If 0th bit is different, 1 is written to the OUT.)
MOD	InA, InB	Modular arithmetic operation. Mod(inB) of the value at the inA is written to the OUT output of the block.The value at the inA is divided by the value at the inB and the remainder is written to the OUT output.(Ex: inA = 253, inB = 10 then OUT = 4)
BIT REPLACE		It is used for replacing a bit of the value of the inA with 0 or 1. The value at the inB specifies the target bit
GET	InA, InB	It is used for reading a WORD register's or a block's value present in the logic project. The block to be read is specified with inB input of the block. It is also used for some special commands. These commands can be seen in diagram below.
LOW LIMIT	InA, InB	Specifies the minimum value that OUT output can take. Desired minimum value is written to the inA input. If inB has a greater value than inA input, the value at the inB is written to the OUT output. Otherwise, the value at the inA is written to the OUT output.(Ex: inA = 10, inB = 8 then; OUT = 10)
HIGH LIMIT	InA, InB	Specifies the maximum value that OUT output can take. Desired maximum value is written to the inA input. If inB has a smaller value than inA input, the value at the inB is written to the OUT output. Otherwise, the value at the inA is written to the OUT output.(Ex: inA = 10, inB = 12 then; OUT = 10)
MERGE A-B	InA, InB	The value at the inB is shifted left by 8bits and added to the value at the inA.
SET	InA, InB	It is used for write to a WORD register or to a block present in the logic project. The block to be written is specified with inB input of the block.(Ex: inA = 10, inB = 3001 then; 10 is written to the block which has block number 3001.)
WORD TO SIGNED		A WORD register containing 16-bit unsigned number is connected to inA input of the block and converted to the 16-bit signed number and written to the OUT output of the block. (Ex: inA=65535 then; OUT=-1, inA=65534 then; OUT=-2)

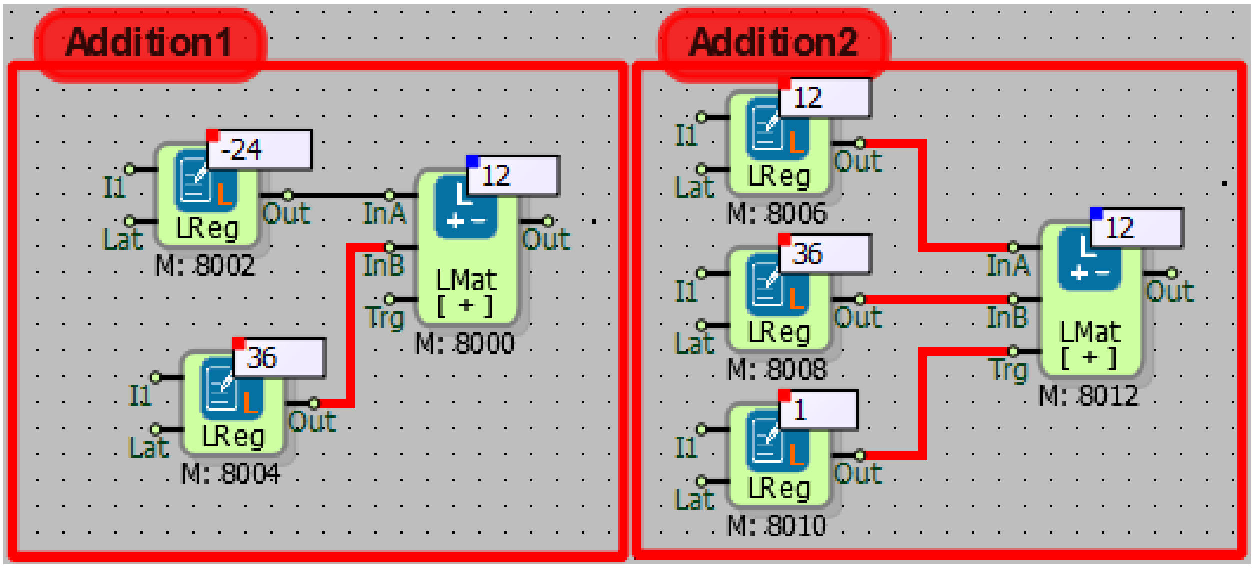
5.6.4.1 GET Operation Special Commands

When performing GET operation, if some special values are entered to the inB input of the block then some special operations are performed by the block. Diagram below shows the commands and the related operations to the commands.

inB Value	Function Explanation
10000	Reads the temperature value from the integrated temperature sensor SHT21. Aavilable only for devices that have the integrated temperature sensor.
10001	Reads the humidity value from the integrated temperature sensor SHT21. Aavilable only for devices that have the integrated humidity sensor.
20000	Reads the RMC geographic latitude data from GPS.
20001	Reads the RMC geographic longitude data from GPS.
20002	Reads the geographic speed data from GPS.(km/h)
20003	Reads the GLL geographic latitude data from GPS.
20004	Reads the GLL geographic longitude data from GPS.
20005	Reads the HEH degree data from GPS.
30001	Real time clock, VBAT – Battery voltage in Volts

5.6.5 Sample Applications

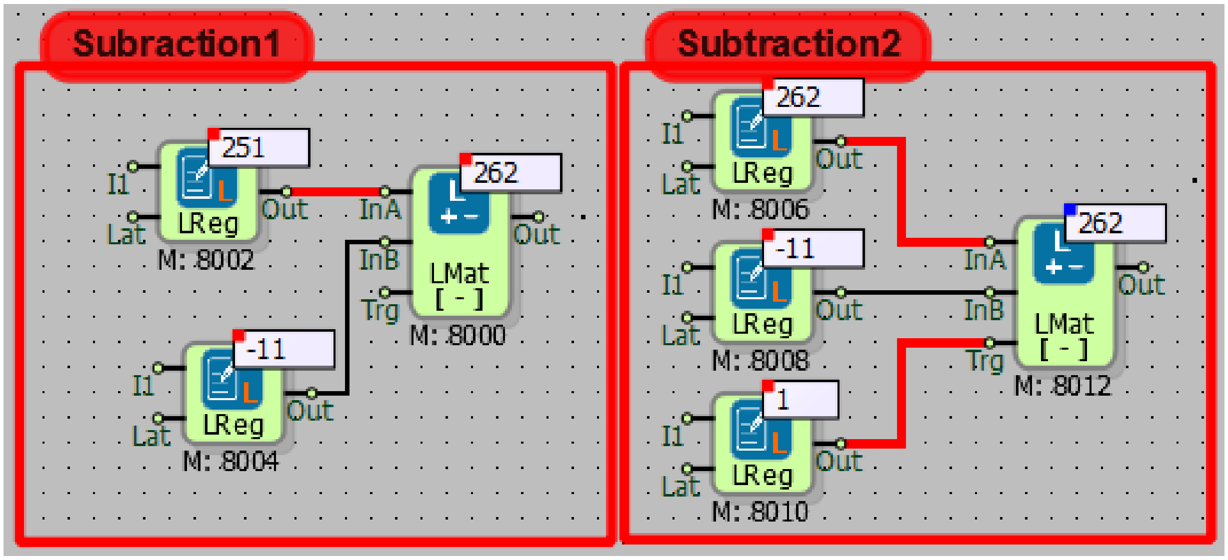
Addition examples:



In “Addition1” example, the values at the inA and inB input of the block are added and the result is written to the OUT output of the block.

In “Addition2” example, “On When Trig is Active” and “Write on Input” is selected. Hence, the values at the inA and the inB are added and the result is written to the inA input at each detected rising edge on the Trig input of the block.

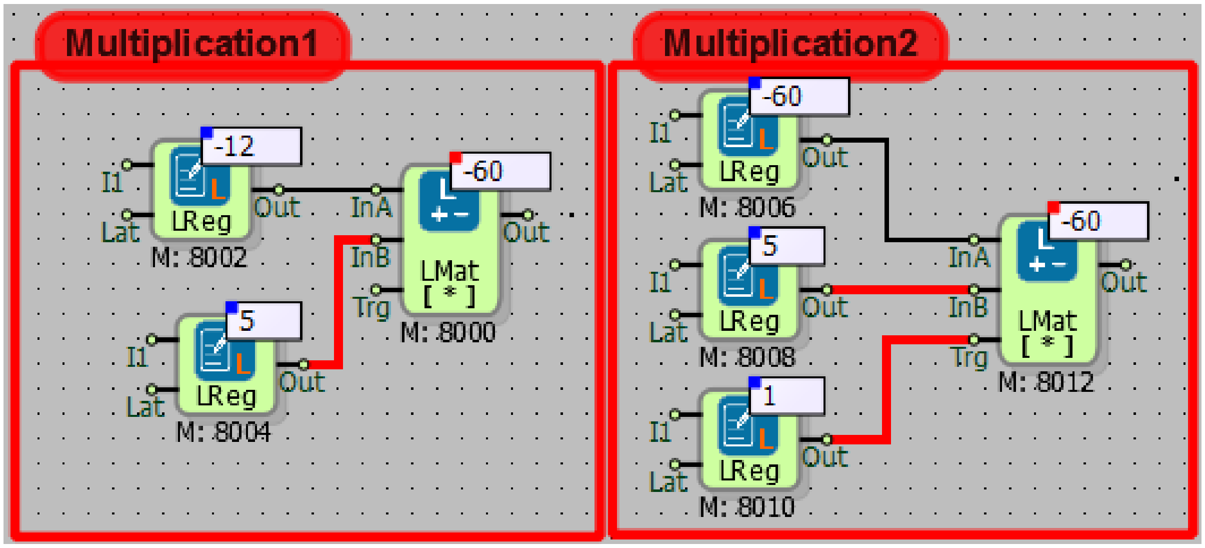
Subtraction examples:



In “Subtraction1” example, the value at the inA of the block is subtracted from the inB input of the block and the result is written to the OUT output of the block.

In “Subtraction2” example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the inA of the block is subtracted from the inB input of the block and the result is written to the inA input at each detected rising edge on the Trig input of the block.

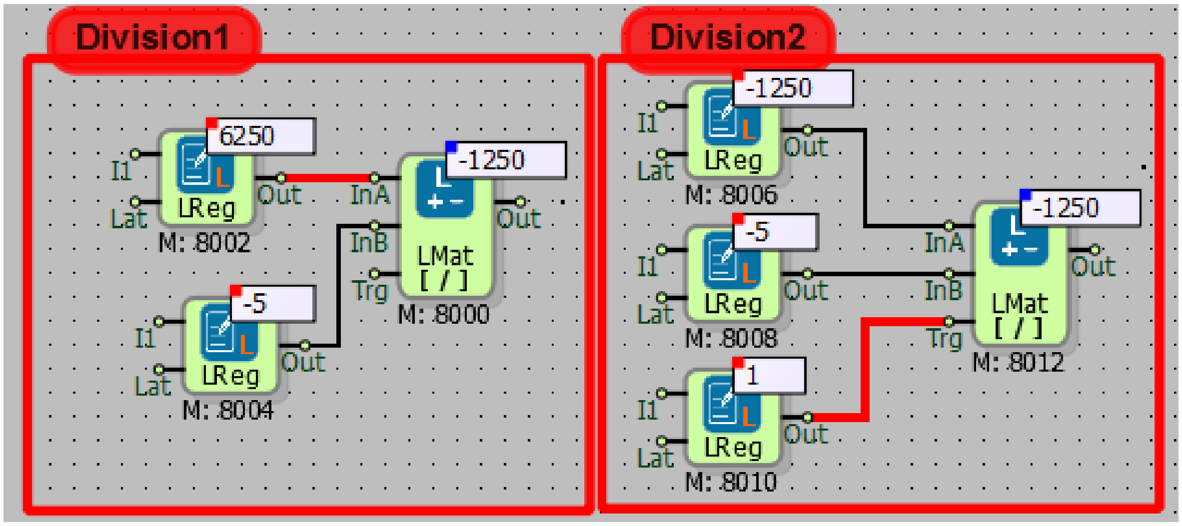
Multiplication examples:



In “Multiplication1” example, the value at the inA input of the block is multiplied by the inB input of the block and the result is written to the OUT output of the block.

In “Multiplication2” example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the inA of the block is multiplied by the inB input of the block and the result is written to the inA input at each detected rising edge on the Trig input of the block.

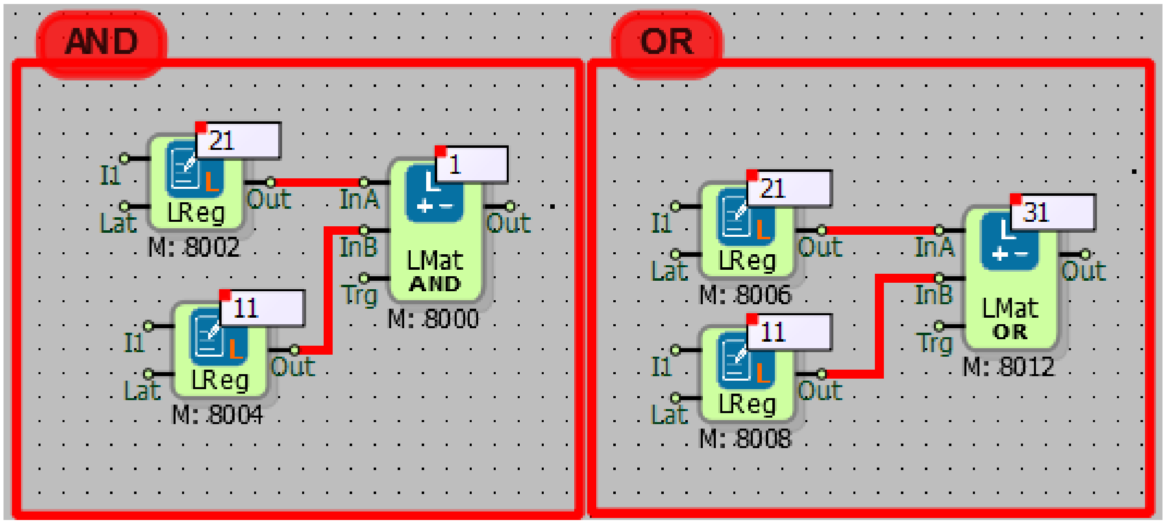
Division examples:



In “Division1” example, the value at the inA input of the block is divided by the inB input of the block and the result is written to the OUT output of the block.

In “Division2” example, “On When Trig is Active” and “Write on Input” is selected. Hence, the value at the inA of the block is divided by the inB input of the block and the result is written to the inA input at each detected rising edge on the Trig input of the block.

AND and OR examples:



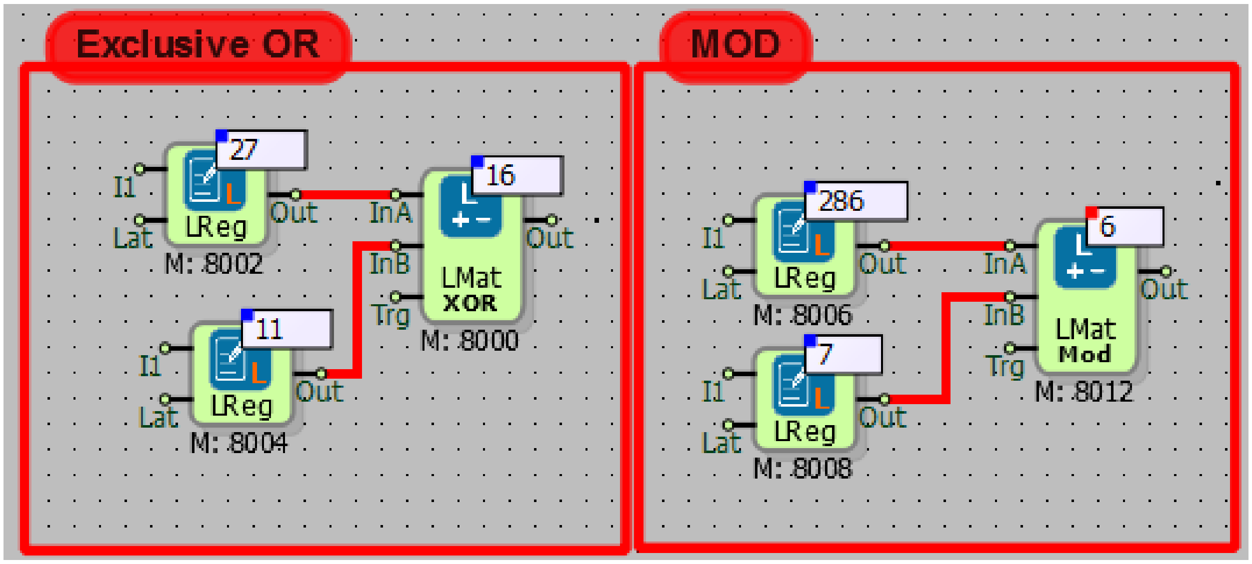
Corresponding binary value of the decimal value at the inA: ; $(21)_{10}=(10101)_2$

Corresponding binary value of the decimal value at the inB: $(11)_{10=}$ $(01011)_2$

The result of bitwise AND operation between inA and inB is: $(1)_{10}=(00001)_2$

The result of bitwise OR operation between inA and inB is: $(31)_{10}=(11111)_2$

Exclusive OR(XOR) and MOD example:



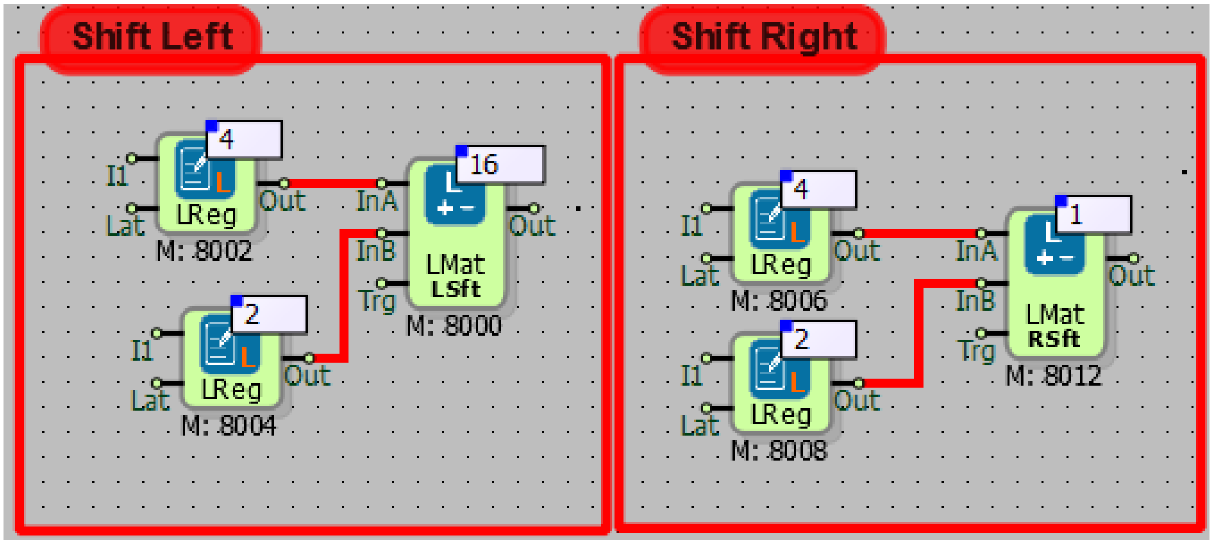
Corresponding binary value of the decimal value at the inA: $(27)_{10}=(\underline{\mathbf{1}}1011)_2$

Corresponding binary value of the decimal value at the inB: $(20)_{10=}$ $(\mathbf{0}1011)_2$

The result of bitwise XOR operation between inA and inB is: $(16)_{10}=(\underline{\mathbf{1}}0000)_2$

In Mod example, the value at the inA is divided by the value at inB and the remainder of the operation is written to the OUT output of the block.

Shift Left and Shift Right examples:



InA'daki değerin bitlerine ayrılmış hali; $(4)_{10}=(00\underline{\mathbf{1}}00)_2$'dir.

InB'deki değer kaç bit kaydırma yapılacağını gösterir.

Out çıkışına InA'daki değerin bitleri kaydırıldıktan sonraki long değeri yazılır.

Sola Kaydır; 4 değeri 2 bit sola kaydırıldığında; $(16)_{10}=(\underline{\mathbf{1}}0000)_2$ değeri elde edilir.

Sağa Kaydır; 4 değeri 2 bit sağa kaydırıldığında; $(1)_{10}=(0000\underline{\mathbf{1}})_2$ değeri elde edilir.

Corresponding binary value of the decimal value at the inA: ; $(4)_{10}=(00\underline{\mathbf{1}}00)_2$

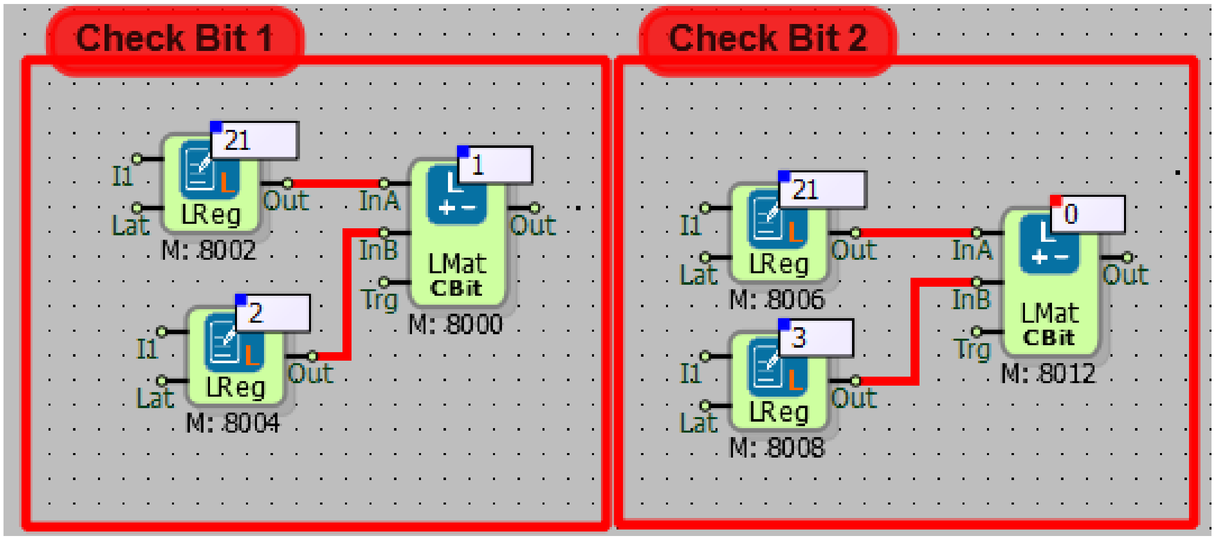
The value at the inB input specifies the number of bits which inA is going to be shifted by.

After the shifting operation, result is written to the OUT output of the block.

Shift Left: When 8 is shifted left by 1: $(16)_{10}=(\underline{\mathbf{1}}0000)_2$ is obtained.

Shift Right: When 8 is shifted right by 1: $(1)_{10}=(0000\underline{\mathbf{1}})_2$ is obtained.

Check Bit examples:



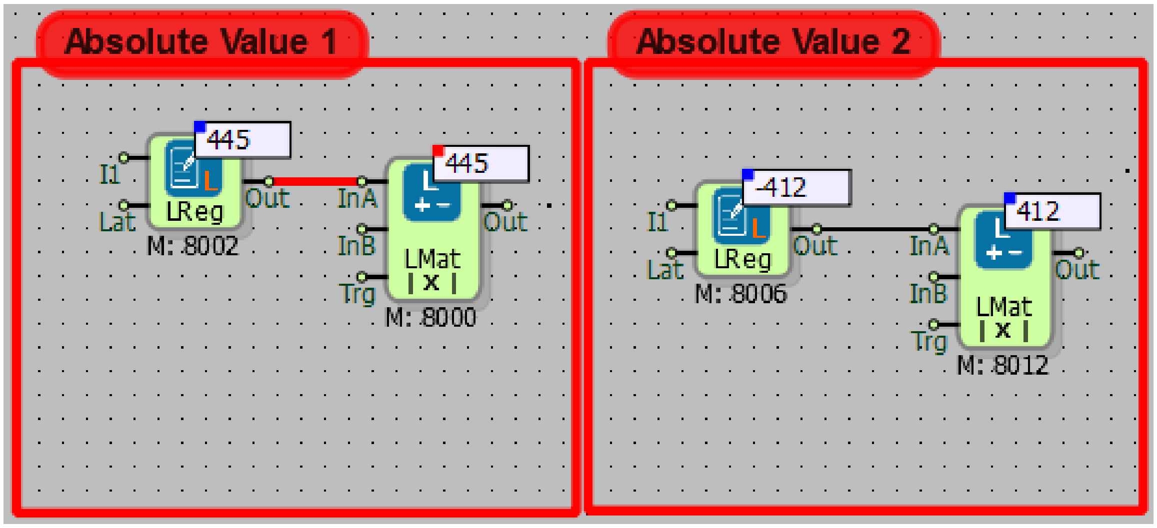
Corresponding binary value of the decimal value at the inA: $(21)_{10}=(10101)_2$

The value at the inB input specifies the index of the bit which is going to be checked. After the checking process, checked bit is written to the OUT output of the block.

In "Check Bit 1" example, the value of the checked bit is $(10\underline{\mathbf{1}}01)_2=1$

In "Check Bit 2" example, the value of the checked bit is $(1\underline{\mathbf{0}}101)_2=0$

Absolute Value examples:

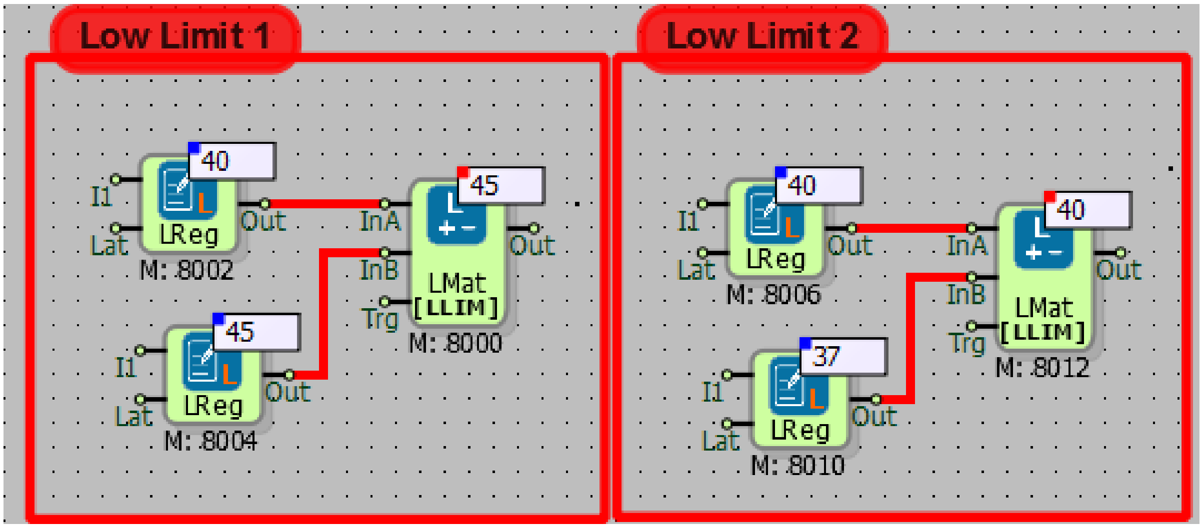


Distance of the value at the inA to the origin is written to the OUT output.

In “AbsoluteValue1” example, distance of 445 to the origin is 445.

In “AbsoluteValue2” example, distance of -412 to the origin is 412.

Low Limit examples:

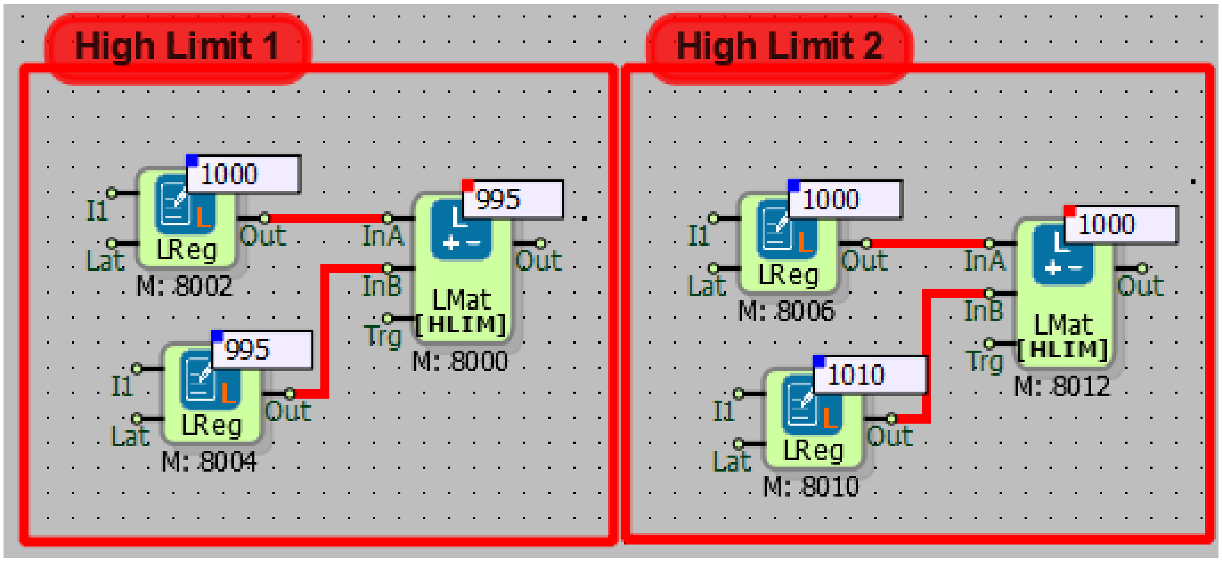


Low limit value is connected to the inA input of the block using a long register.

In “Low Limit 1” example, low limit is not activated. Since the value at the inB input is greater than the low limit, the value at the inB is written to the OUT output of the block.

In “Low Limit 2” example, low limit is activated. Since the value at the inB input is smaller than the low limit, the value at the inA is written to the OUT output of the block.

High Limit examples:

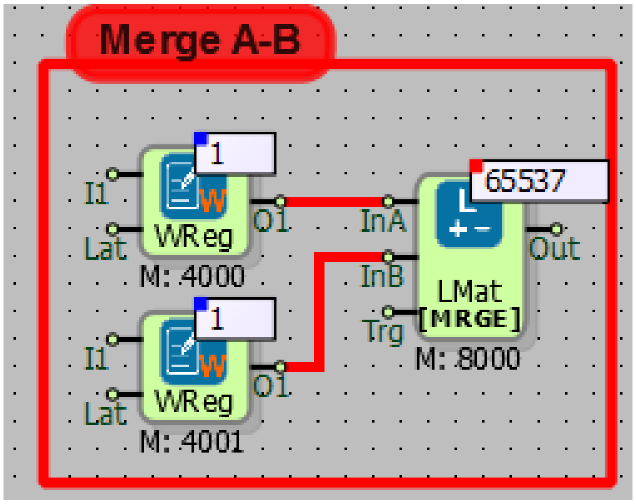


High limit value is connected to the inA input of the block using a WORD register.

In “High Limit 1” example, high limit is not activated. Since the value at the inB input is smaller than the low limit, the value at the inB is written to the OUT output of the block.

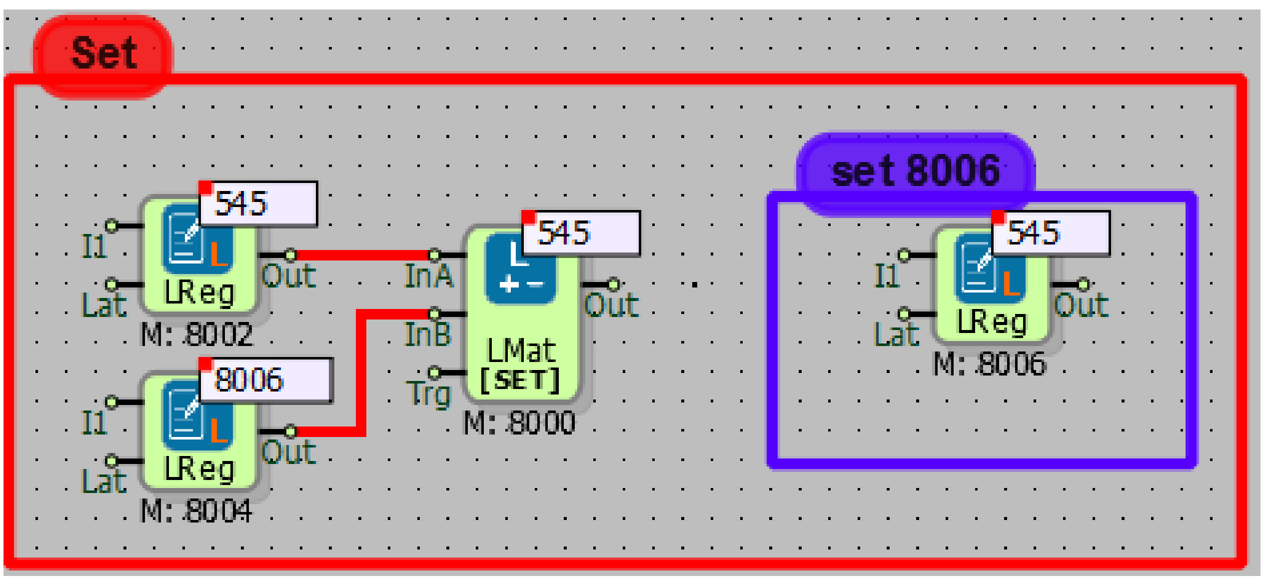
In “High Limit 2” example, high limit is activated. Since the value at the inB input is greater than the low limit, the value at the inA is written to the OUT output of the block.

Merge A-B example:



The value at the inB block is shifted left by 8bits and added to the value at the inA input of the block.The result is written to the Out output of the block. Two 16-bit word register's bits are concetanated with Long Math block.

Set example:



The value to be set is connected to the inA input of the block.

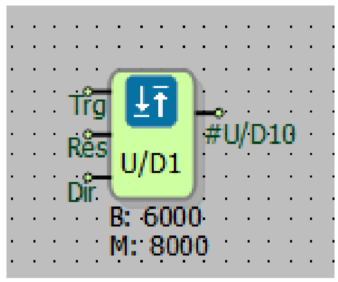
Number of the target block is connected to the inB input of the block.

The value at the inA input, 545, is set to the block with number 6003.

6 COUNTER BLOCKS

6.1 UP/DOWN COUNTER 1

6.1.1 Connections

Trg: Trigger input		#U/D10: Block output
Res: Reset input		
Dir: Direction input		

6.1.2 Connection Explanations

Trg: Trigger input

It is the trigger input.

Res: Reset input

The counter's reset input.

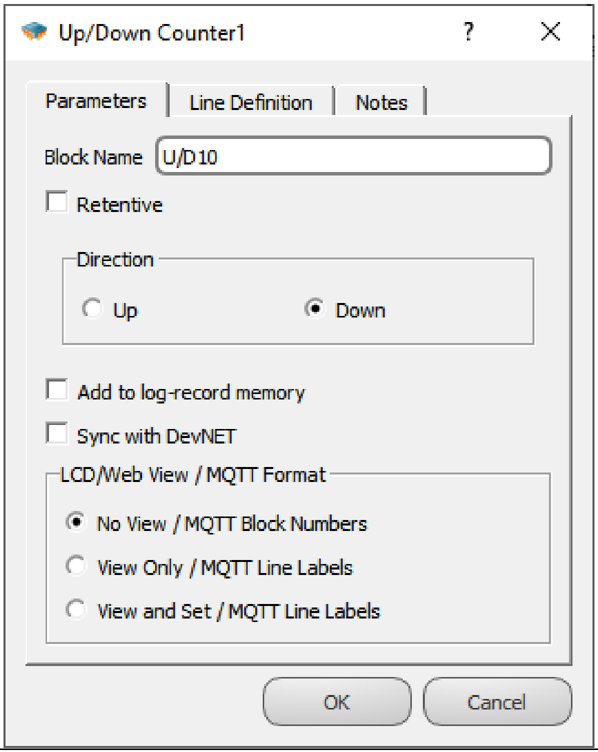
Dir: Direction input

Counter direction binary input.

#U/D10: Block output

Counter value output.

6.1.3 Block Settings

	Up: If selected; Counter increases in the positive (+) direction. If it is desired to select from outside the block, Logic high(1) should be applied to "Dir" input.
	Down: If selected; Counter increases in the negative (-) direction. If you want to select from outside the block, logic low(0) should be applied to "Dir" input.
	Retentive (Persistence): If selected; the counter keeps the last value when the power of the device is interrupted or reset.

6.1.4 Block Explanation

It is used to increment the counting process from any value in positive (+) direction one by one, or to reduce a value in negative (-) direction one by one.

If the counter direction is to be determined from outside the block;

"Dir" input of the counter is logic high(1) => the counter has positive (+) direction

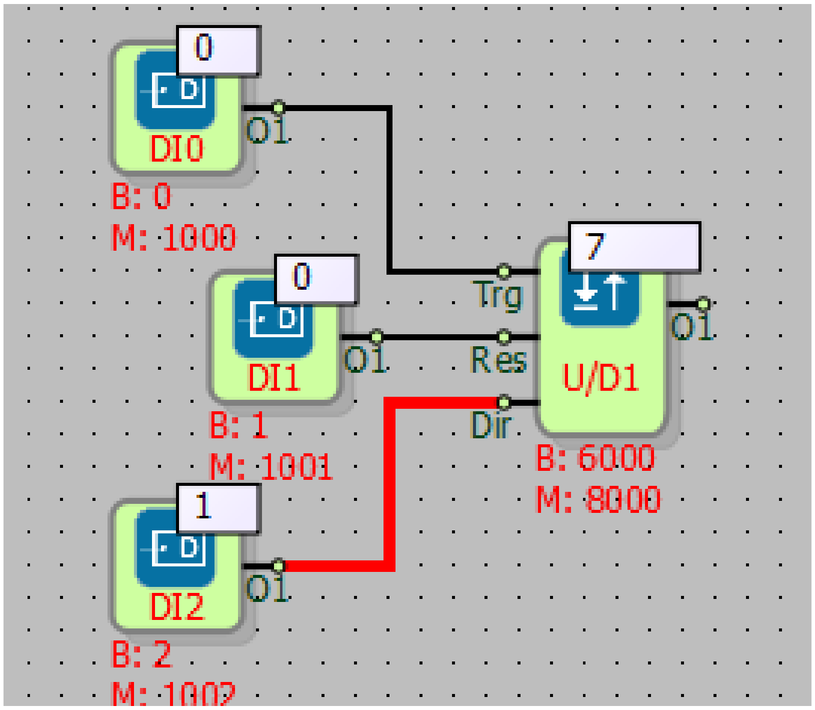
"Dir" input of the counter is logic low(0) => the counter has positive (-) direction

The counter Increases/decreases value by 1 on the rising edge of the logic high(1) signal applied to “Trg” input.

The reference point from which the counting process starts can be specified by overwriting the block register.

It can count 32 bits signed integers.

6.1.5 Sample Application

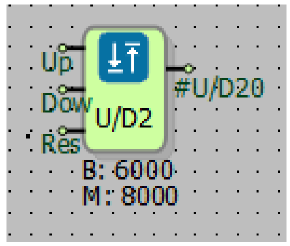


In the example, on the rising edge of each logic high(1) signal coming from DI0; If the DI2 input is logic high(1), it performs counting upwards, else if the DI2 input is logic low(0), it performs counting downwards.

Logic high(1) from DI1 input is used for resetting the counter.

6.2 UP/DOWN COUNTER 2

6.2.1 Connections

Up: Up input		#U/D20: Block output
Dow: Down input		
Res: Reset input		

6.2.2 Connection Explanations

Up: Up input

The counter value increases by 1, when “Up” input triggered.

Dow: Down input

The counter value decreases by 1, when “Dow” input triggered.

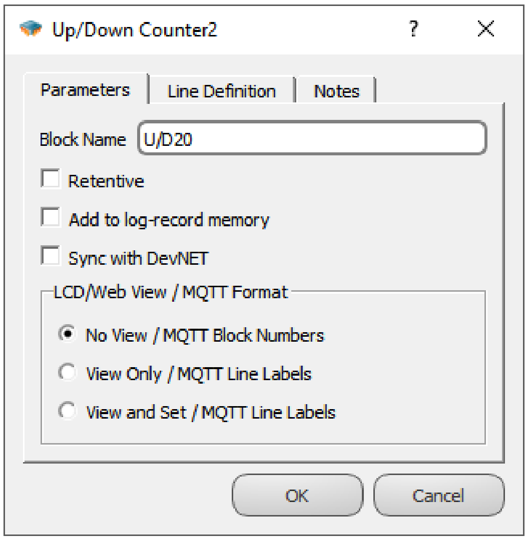
Res: Reset input

It is counter’s reset input.

#U/D20: Block output

It is counter’s value output.

6.2.3 Block Settings

	Retentive (Persistence): If selected; the counter keeps the last value when the power of the device is interrupted or reset.

6.2.4 Block Explanation

It is used when positive (+) direction and negative (-) direction counting is done from two different inputs on the block.

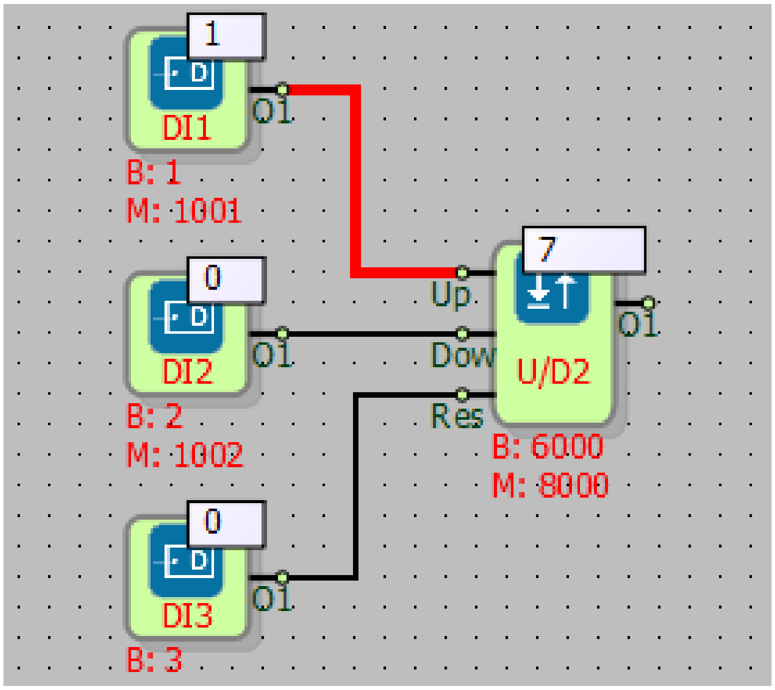
The counter value increases by 1 when the rising edge applied at the "Up" input.

When the rising edge applied the "Dow" input, the counter value 1 is decremented.

The reference point from which counting starts can be specified by writing on the block register.

Up to 32-bit counting can be performed.

6.2.5 Sample Application



In the example;

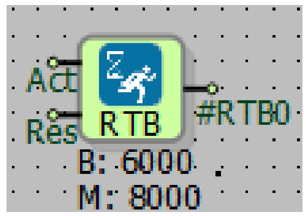
At the rising edge of each logic high(1) signal DI1, the counter value is incremented by 1.

At the rising edge of each logic high(1) signal DI2, the counter value is decremented by 1.

DI3 input logic high(1) is used to reset the counter.

6.3 RUN TIME

6.3.1 Connections

Act: Activation input		#RTB0: Block output
Res: Reset input		

6.3.2 Connection Explanations

Act: Activation input

Block enable input.

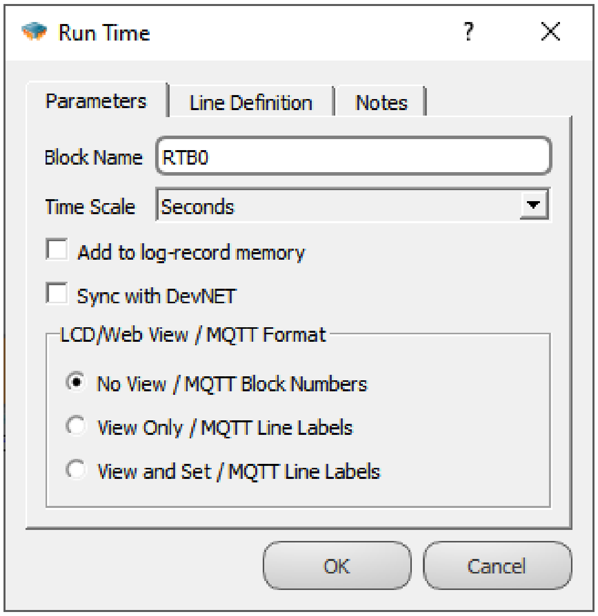
Res: Reset

Run-time counter’s reset input.

#RTB0: Block output

Runtime value.

6.3.3 Block Settings

	Time Scale: “Seconds, minutes, hours" can be selected from the time scales.

6.3.4 Block Explanation

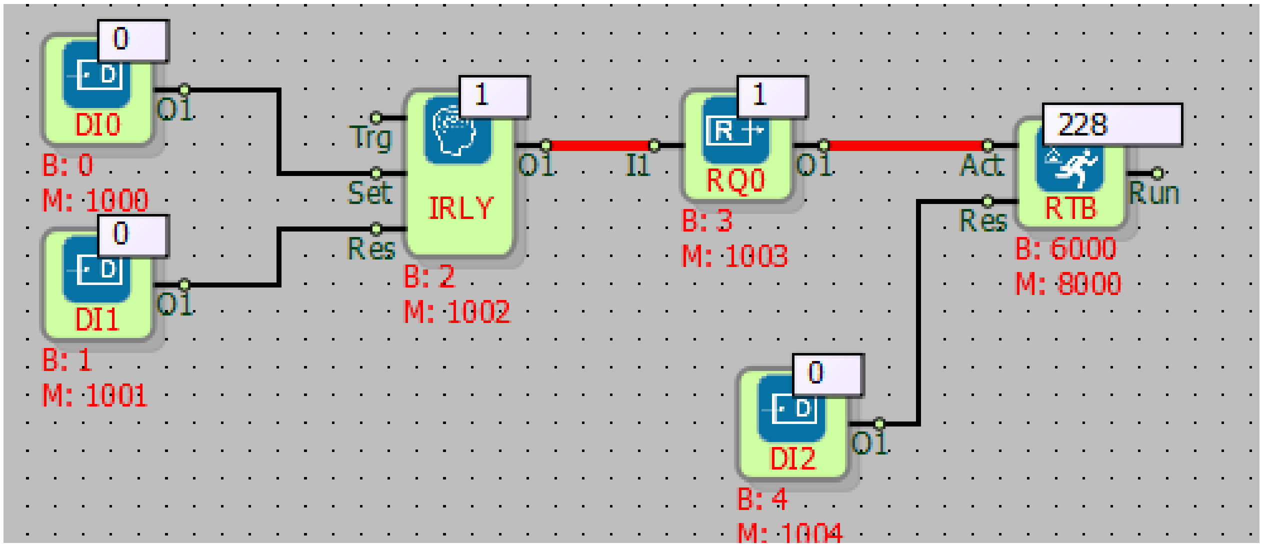
Run Time block is used to save the runtime.

When the "Act" input is logic(1), it counts the time in selected time scale (seconds, minutes, hours and writes to the output.

On every logic(1) signal applied on the "Act" input, it continues to count from the last value.

The counter value is reset when the rising edge is applied on the block "Res" input.

6.3.5 Sample Application



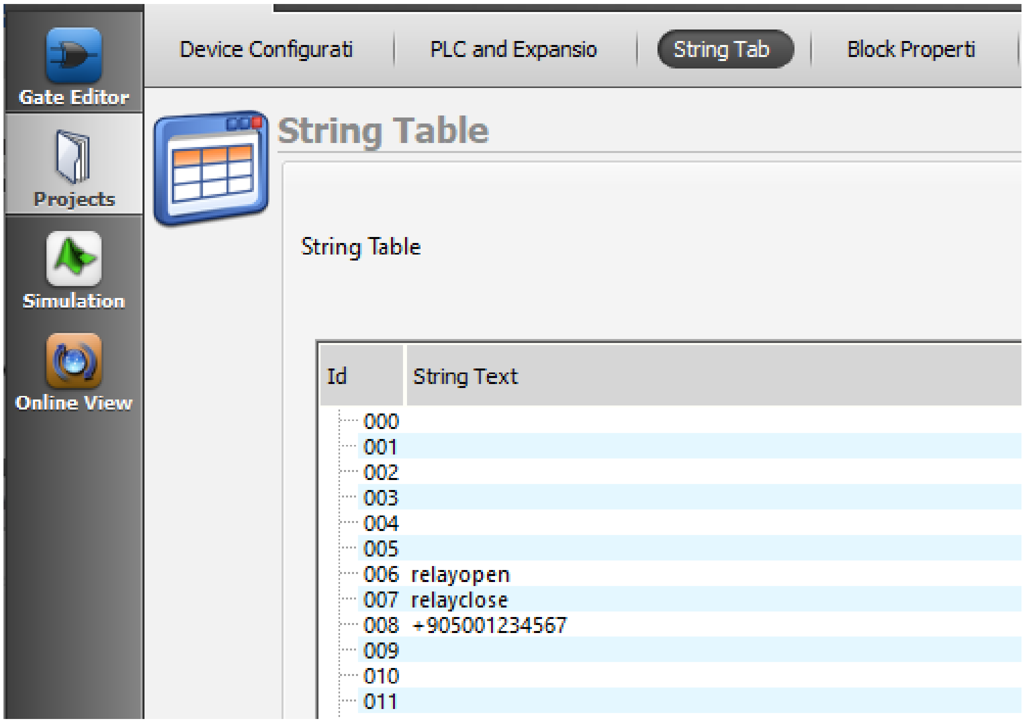
In the example, RQ0 is started from DI0 input and stopped from DI1 input.

With the RTB block, the duration when the RQ0 is logic high(1) will be monitored.

DI2 input will reset the run time.

7 GSM BLOCKS

In the group of GSM blocks; There are blocks for receiving SMS, sending SMS, starting DTMF call, receiving DTMF call and GSM signal quality.

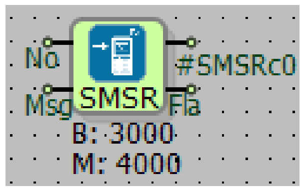


(1)

SMS contents and GSM numbers are written in the "String Table" in GSM blocks. Picture (1) SMS contents and phone numbers written in the text table are selected with the "String Reference Block".

7.1 SMS RECEIVER

7.1.1 Connections

No: Phone Number input		#SMSRc0: Block output
Msg: Message input		Fla: Flag output

7.1.2 Connection Explanations

<u>No: Phone Number input</u>

It is for SMS filtering by sender phone number. Only SMS messages send by this phone number is accepted. If it is empty or not connected, there will be no SMS filtering by sender phone number.

<u>Msg: Message input</u>

Reference message input for comparison

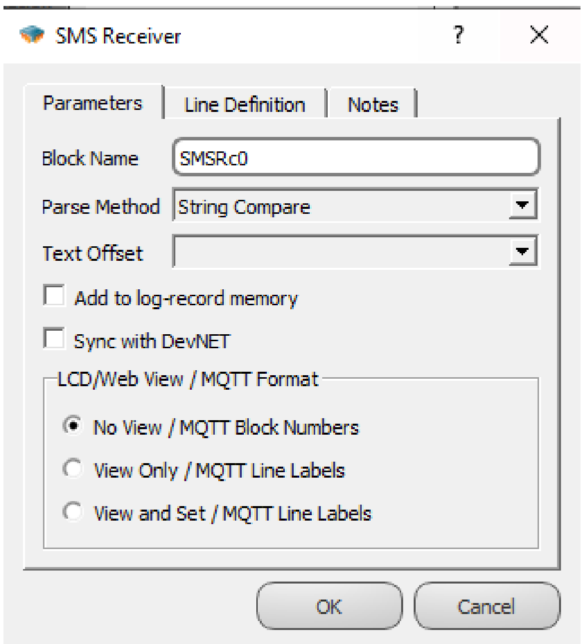
<u>#SMSRc0: Block output</u>

The received SMS message is processed according to the parse method. Result of SMS Text parsing is written block output.

<u>Fla: Flag output</u>

If a new SMS text message is received, the Fla output generates a single cycle pulse output.

7.1.3 Block Settings

	Parse Method: There are four methods; “String Compare”, “Ascii to Integer”, “Text=:Value” and ”Write Into Device”.
	Text Offset: In the string table, determines the offset which the received SMS will be saved into. **Note:** Text offset should be selected from unused string offset. Because, received text message will be written into it.

7.1.4 Block Explanation

SMS Receiver block is used in applications requiring SMS control. String reference blocks are connected to the No and Msg inputs.

“Text Offset” combobox, determines the offset which the received SMS will be saved into. This index value should be an appropriate value in the string table, care must be taken for not to affect the indices used by other blocks.

The incoming SMS text is written to the index determined from the options. Thus, this value can be used as desired with text reference.

Phone No to be accepted: You need to enter the telephone number into the "String Table" which will be used to accept SMS messages including the country code (i.e +44752…). If an SMS from any number will be accepted, this input is left blank or the phone number is entered as “0”.

Parse Method: If the “String Compare” option is selected in the “Parse Method” combobox in the SMS Receiver block settings, the text of the received SMS is compared with the text in the “Msg” input. If the text compared with the received SMS is same, the block output becomes high(1) and continuously remains in high(1) state.

If the "Ascii To Integer" option is selected in the "Parse Method" combobox, content of the received SMS is converted into integer and written to the block output.

If the “Text=:Value” option is selected in the “Parse Method” combobox, the “Value” value in the “Text=:Value:” format message saved in the text table is written to the block “Out” output as soon as the SMS is received. If the text reference connected to the msg pin is the same as the Text part of the message, the value is written to the output. If not the same, it preserves the value. For example, if ABC is written in the text reference, when ABC=123 is sent, 123 information is written to the output. If AB=12 is sent, the value does not change.

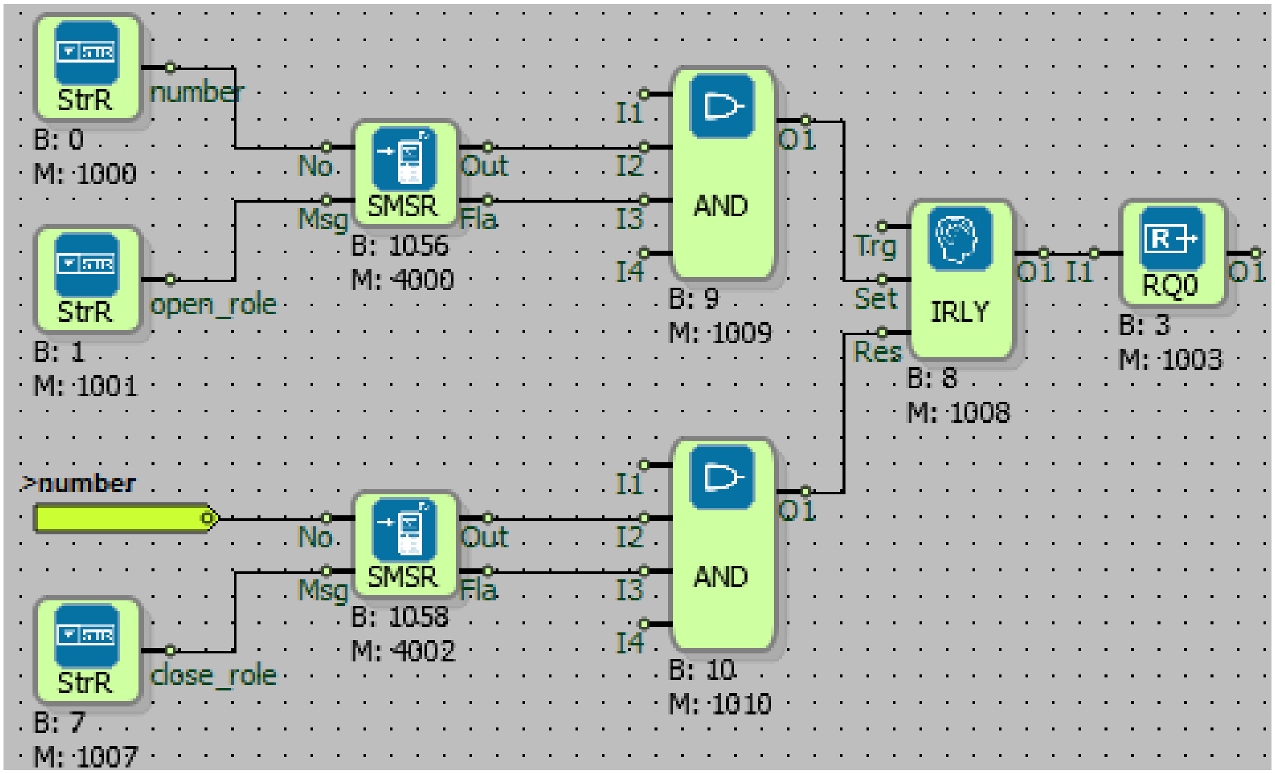
If the "Write Into Device" option is selected, it allows the received SMS to be saved to the string offset selected from the block special settings.

If a text reference block is connected to the No pin, only the SMS from the connected number will be received, otherwise it will receive the SMS from any number.

When each SMS is received, the Fla output generates a rising edge trigger.

The SMS Receiver block is available on non-PPP firmwares if only the device has GSM feature and SMS feature is turned on.

7.1.5 Sample Application

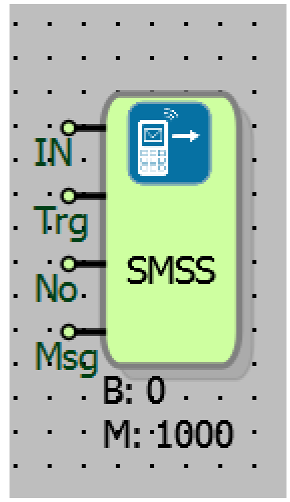


Id	String Text
000	+905321234567
001	openrole
002	closerole
003	

SMS Receiver blocks are used to turn the system on and off. The system works according to the information from the number indicated in the SMS Receiver block. When the "open_role" SMS is received from the number specified in the string table, the pulse relay output and the RQ0 will become logical high(1) and the system will start to operate. When we consider the system off blocks group; "close_role" is connected to the string reference, and when "close_role" SMS is received from the number specified in the string table, the pulse relay RQ0 will become logical low(0) and the system will stop. "Out" and "Fla" outputs are connected to AND gate, and each time the SMS arrives, the operations are performed in the same way.

7.2 SMS SEND

7.2.1 Connections

IN: Value input	
Trg: Block trigger input	
No: Number input	
Msg: Message input	

7.2.2 Connection Explanations

IN: Value input

Block input used in sending SMS Text

Trg: Block trigger input

Rising edge at this input sends the SMS.

No: Number input

Destination phone number of sending SMS

Msg: Message input

Text message body used in sending SMS Text

7.2.3 Block Settings

There are no block settings.

7.2.4 Block Explanation

This block is used to send SMS Texts from the device to any mobile phone. When a rising edge signal is applied to “Trg” input of the block, SMS text is build from the “Msg Input” - SMS text body and then it will be sent to mobile number defined in “Number Input”

No and Msg inputs must be connected to string type blocks. When the rising edge of the logical high(1) signal is input to the “Trg” input, the SMS is sent.

The string reference blocks are connected to the input “No” and the number to which the SMS will be sent is selected from string table.

The phone number must contain country code like "+901234567898".

If you need to send SMS to the last number which SMS is received from, the symbol "<" defined in the string table should be entered in the string reference connected to the No input.

In the Msg input, the SMS content to be sent is entered. This content also needs to be connected through a string reference block.

If you want to send a block value connected to the block’s “IN” input as SMS, “%s” should be written into the SMS content to be sent in the string table. For example; "Room temperature is %s". (“% s” is replaced with the block value in the IN input is replaced.)

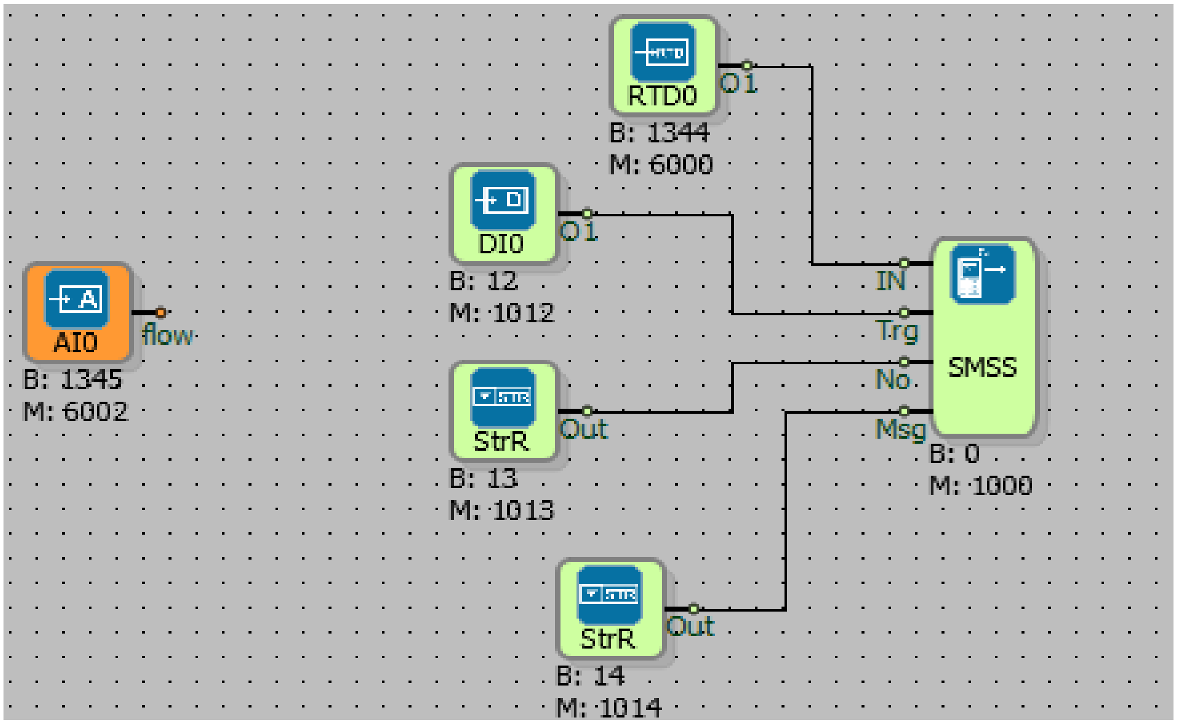
In order to be able to send more than one block values by SMS, '$' is added to the beginning of the block addresses and added to the string table. For example, if the description in the string table is "Measured values are line1: $3000, line2: $3004" is sent, values of the blocks 3000 and 3004 are sent.

Usage	Example	Text Result
$<Block Numner>	Temp: $5000 , Hum: $5001	Temp: 23.45, Hum: 88.02
$TIME	Value: $3000 at $TIME	Value: 2341 at 18.06.2018 09:55
$SRNO	Value $3008 from $SRNO	Value 324 from 1000213

Note: A maximum of 63 characters can be entered into the text field in the String Table.

Note: The SMS functions are only available on PPP disable firmware.

7.2.5 Sample Applications



Id	String Text
000	+905321234567
001	tempeture=%s_flow=$1345
002	

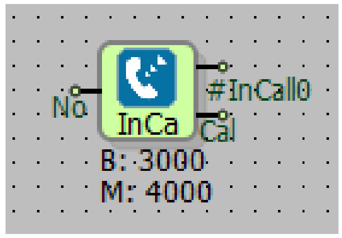
In the example; The SMS trigger is provided on rising edge trigger from DI0 input.

The string table contains the number and SMS content to be sent.

The SMS content is "temperature =%s,_flow=$5001". Here, the RTD temperature value of the SMS Send block’s “IN” input is sent with the command "%s" and AI0 (line 5001) is sent with "$ 5001" command as SMS with the rising edge trigger coming to the block value “Trg” input.

7.3 INCOMING DTMF CALL

7.3.1 Connections

No: Incoming call number input		#InCall0: DTMF code output Cal: Call accepted output

7.3.2 Connection Explanations

No: Incoming call number input

It is for filtering by caller phone number. Only incoming calls from this phone number is accepted. If it is empty or not connected, there will be no incoming call filtering.

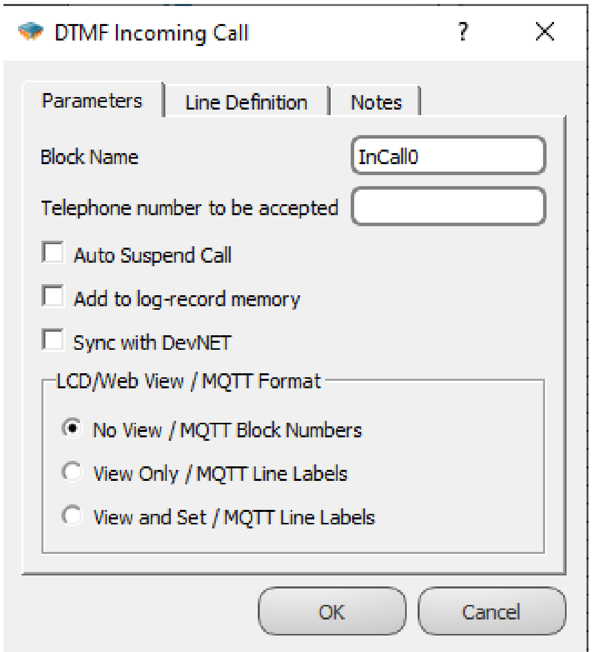
DTM: DTMF code output

DTMF code output.

Cal: Call accepted output

If an incoming call is accepted and haven't yet hang up, this output goes to Logic (1)

7.3.3 Block Settings

<table>
<tr>
<td rowspan="2">
</td>
<td>Telephone number to be accepted: The phone number to be accepted can be entered inside the block.</td>
</tr>
<tr>
<td>Auto Suspend Call: This option can be clicked if the incoming call is requested to be busy.</td>
</tr>
</table>

7.3.4 Block Explanation

Thanks to the DTMF blocks, Remote projects via Phone DTMF codes can be easily done. If incoming call is generated from the specified number or there is no phone number filter, then, call is accepted by the device and the DTMF codes entered from the remote phone is reflected on the block output.

The string reference blocks are connected to the input “No” and the number to which the call will be done is selected from string table.

Phone number should include country code like "+901234567898".

Call output generates a logical high(1) signal at its output as long as a call continues.

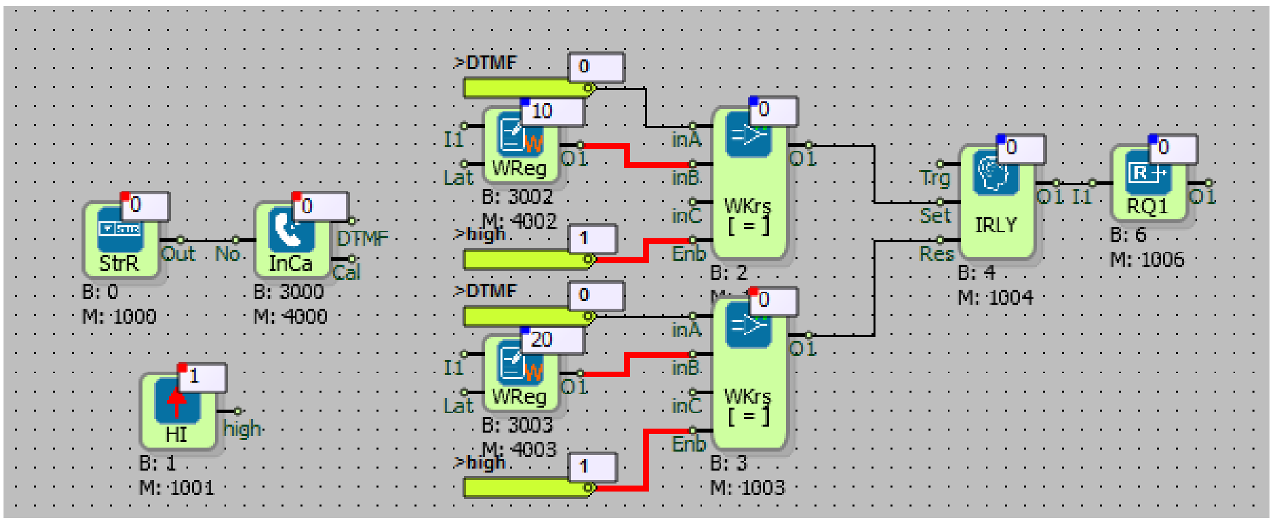
After the call is accepted, the "*" key is pressed first in the telephone in order to operate with the DTMF code. Enter the desired DTMF code and press the "#" key. Here, the value entered between * and # is transferred to the DTM output as a “word integer” output.

As an example, when "* 1234 #" is entered, the value of "1234" is read out from DTM output. This value can be used as a word value as desired.

The same operation is repeated to transfer the DTMF code again. That is, DTMF code input is started with "*" key. The DTMF code entered with the "#" key is transmitted to the output.

Note: The DTMF Incoming Call block is available on non-PPP firmware.

7.3.5 Sample Applications

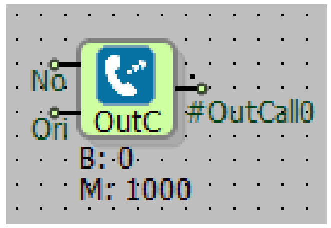


In the example; The telephone number whose call will be accepted is entered with the text reference. If "equal" is selected in the comparators, if the inB value of the comparison type is equal to the inA value, then the outputs are logic high(1).

When the DTMF code * 10 # is sent after the call is accepted, the pulse relay is set and RQ0 will be logic high(1). When the DTMF code * 20 # is sent, the pulse relay output will be reset and RQ0 will be logic (0). In this way, any equipment with DTMF codes can be subjected to remote control operations such as turn off/on etc.

7.4 OUTGOING DTMF CALL

7.4.1 Connections

No: Dialing Number Ori: Start to dialing		#OutCall0: Block output

7.4.2 Connection Explanations

No: Dialing Number

Phone number to dial

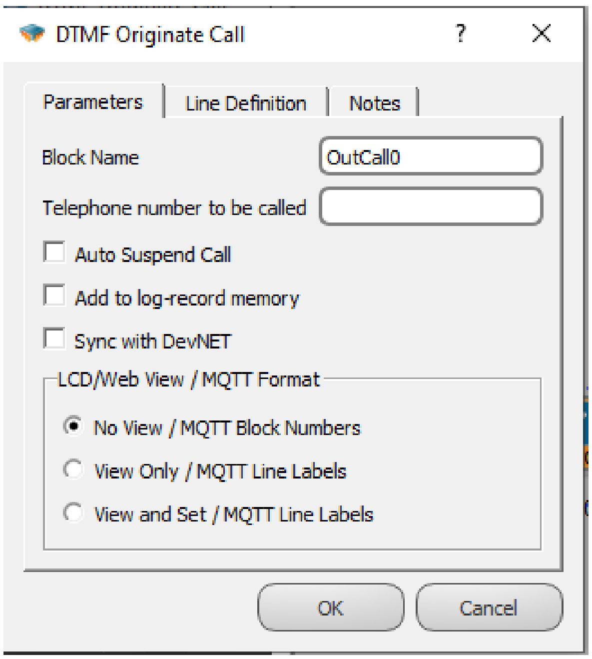
Ori: Start to Dial

The block input that must be changed to logical high(1) to start a call.

#OutCall0: Block output

It is the block output that indicates whether the call has been accepted or not.

7.4.3 Block Settings

<table>
<tr>
<td rowspan="2">
</td>
<td>Telephone number to be called: The phone number to be called can be entered in the block.</td>
</tr>
<tr>
<td>Auto Suspend Call: This option can be clicked if the incoming call is requested to be busy.</td>
</tr>
</table>

7.4.4 Block Explanation

Applying the logical high (1) signal to the “Ori” input of the DTMF Originate Call block makes a call to the defined number.

The DTMF code cannot be sent even if the incoming call is answered by the user. In the case of a scenario in which a program is defined, a call is made with the rising edge trigger coming to the “Ori” input.

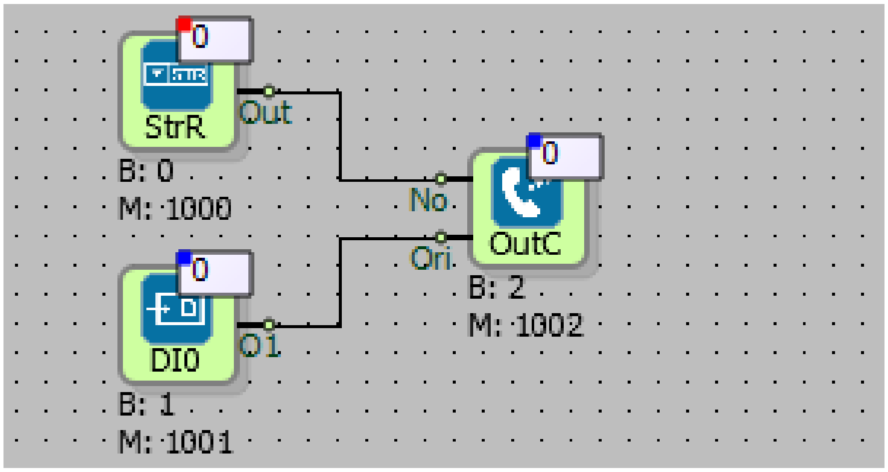
Enter the phone number to originate the call to input “No” with string reference blocks. You can also enter the number in the block options by leaving this input blank.

When a high-level signal arrives at the “Ori” input, the block will be activated and the specified number will be called.

Enter the phone number to originate the call will be done to Turkey in the text table "+90" adding "+901234567898" should be entered.

Note: The DTMF Originate Call block is available on devices with the GSM feature and the device is available on the SIM card when the call feature is turned on.

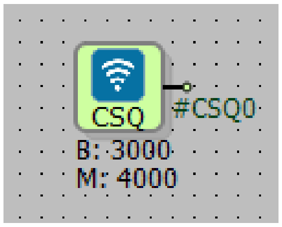
7.4.5 Sample Applications



The telephone number to call is determined by text reference. The number specified by the rising edge trigger signal coming to the input "Ori" will be dialed.

7.5 GSM SIGNAL QUALITY

7.5.1 Connections

	#CSQ0: Block output

7.5.2 Connection Explanations

O: Block output

It is a block output with a signal quality value between -1 and 31.

7.5.3 Block Settings

There are no block settings

7.5.4 Block Explanation

This is a block that can be used to monitor GSM signal quality. It gives a value between -1 and 31. Values -1 and 0 indicate that there is no GSM connection, and values 1 and 31 indicate the signal quality of the device.

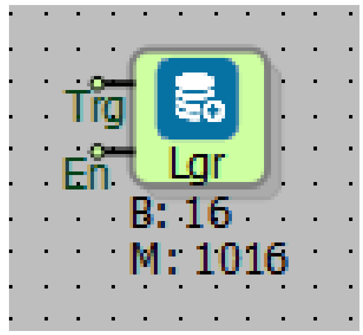
If block value is 1, the signal level is at the lowest level and 31 is at the highest level.

This feature is only available on non-PPP firmwares for devices with GSM capability.

8 DATA/EVENT RECORDING BLOCK

8.1 LOGGER

8.1.1 Connections

Trg: Block trigger input	
En: Block activation input	

8.1.2 Connection Explanations

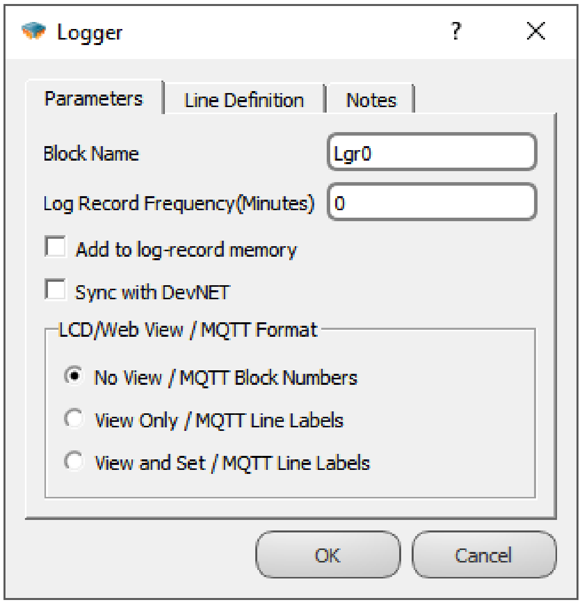
Trg: Block trigger input

Every rising edge triggers all the block data with the "Add to log-record memory" selected in log-memory.

En: Block activation input

When there is logic(1) signal in its input, the block is active.

8.1.3 Block Settings

	Log Record Frequency(Minutes): How often the data can be logged is set in minutes from within the block.

8.1.4 Block Explanation

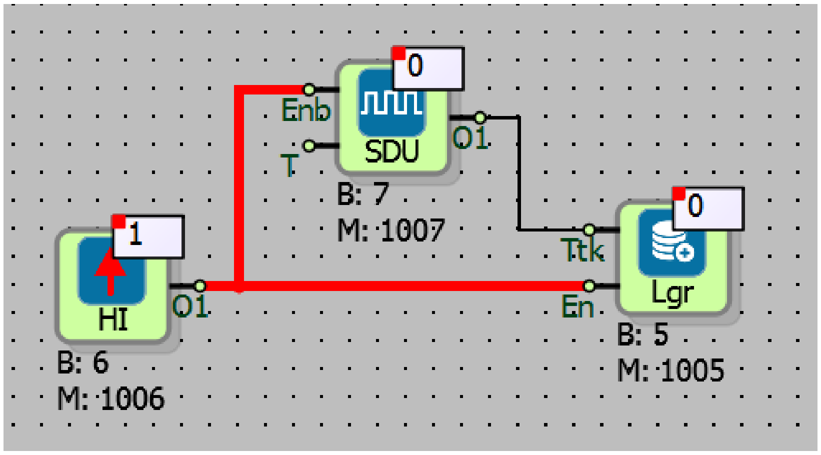
In control devices to do LOG record operation is used. LOG record operation on the devices which support the SD card is made on SD card, if there is no SD card in the device it is done on the flash memory.

With every high edge signal which is comes to Trg input, the LOG record is kept. Which block datas will write to the LOG memory in logger operation is determined with the choosing “Add to log-record memory". Block data and real time information are written together.

When is applied the high signal to the “En” input, The block will active.

“Add to log-record memory” choice must be choosen in block choices which is wanted recording for log record.

8.1.5 Sample Application

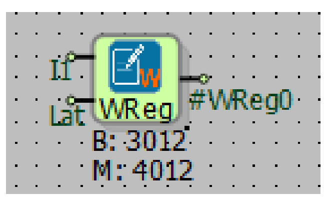


In the sample; A logging process is performed periodically using a symmetric pulse generator for 5 minutes. The values of all blocks with the add to log record option checked are added to the log record memory every 5 minutes.

9 REGISTER/VARIABLE BLOCKS

9.1 WORD REGISTER

9.1.1 Connections

I1: Data Input Lat: Latch Signal		#WReg0: Word Output

9.1.2 Connection Explanations

<u>I1: Data Input</u>

Data input which is latched into register.

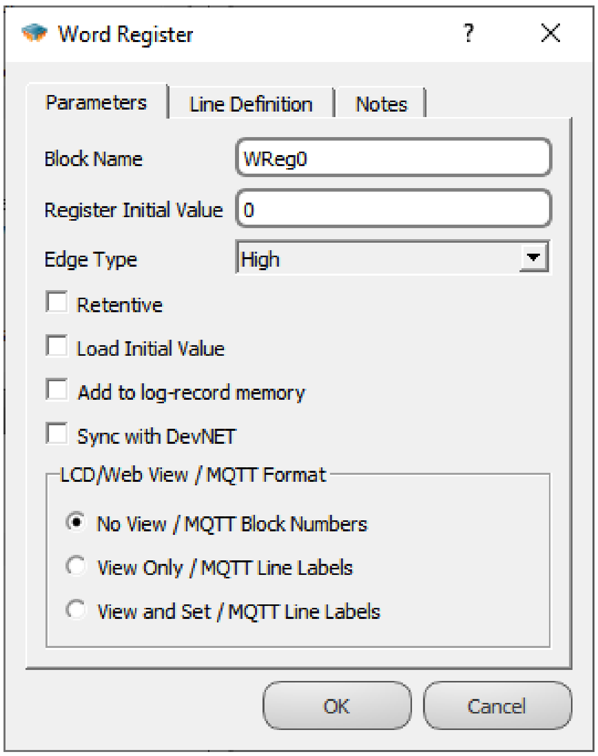
<u>Lat: Latch Signal</u>

Data is latched into the register memory within control of Lat signal.

<u>#WReg0: Word Output</u>

#WReg0 is Block output. It reflects the internal 16 bit Word Register value.

9.1.3 Block Settings



Register Initial Value: The initial value which will be written in the register memory at startup.
Edge Type: Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process. Edge Type Options: High, Low, Raise, Fall, Raise/Fall
Persistence: If it is selected, register value is non-volatile even if the device power is off. Last value of the register is reloaded automatically after power on.
Load Initial Value: Active only Persistence is selected. This is a selection between initial value coming from user project or last saved value coming from non-volatile momory as a initial value after new project is downloaded into device.

9.1.4 Block Explanation

Word Register Block is used as a 16 bit unsigned integer type value holder. It is used as variable in PLC projects.

Using the Lat Signal, the block can be used like a D-Type Latch.

Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process.

Possible “Edge type “ options and usage are given at following table:

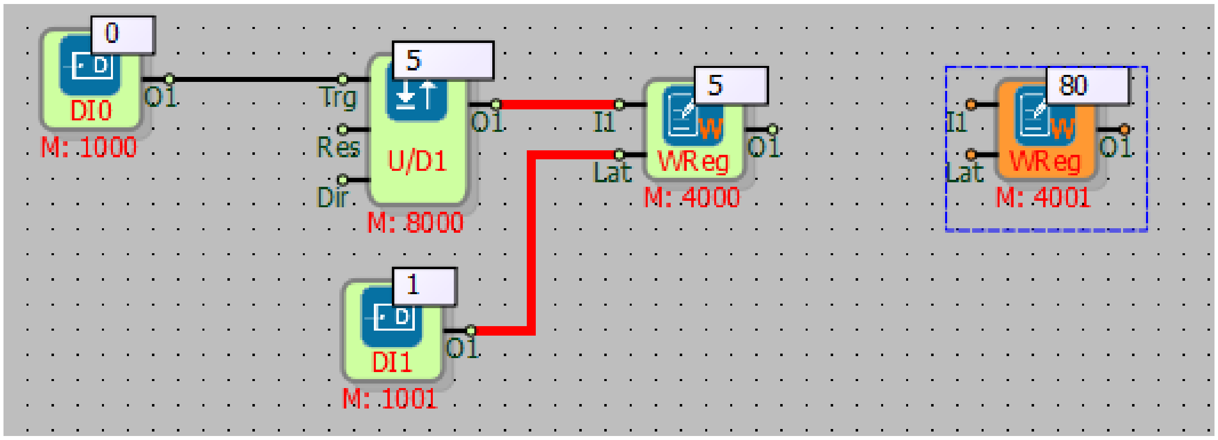
High	Only if the Lat Signal is Logic(1), Value at I1 input is saved into Register Memory
Low	Only if the Lat Signal is Logic(0), Value at I1 input is saved into Register Memory Note: if Lat signal is not connected, it means to Low – Logic(0)
Raise	Value at I1 input is saved into Register Memory when Raising edge of the Lat Signal
Fall	Value at I1 input is saved into Register Memory when Falling edge of the Lat Signal
Both	Value at I1 input is saved into Register Memory when Raising or Falling edge of the Lat Signal

I1 Data Input signal type may be different from register block type. For example, Analog signal can be applied to Word register block. In that case, Automatic variable casting occurs. Therefore, user must be pay attention to variable types.

Sample transformation table is given the below from different variable types for entiring value to the word register

It is the variable type in input	Sample Input Value	It is value which is will be loaded to the word register
Binary	0	0
Binary	1	1
Analog	12.34	12
Analog	98.9	98
Long	65000	65000
Long	80000 (0x00013880)	14464 (0x3880)

9.1.5 Sample Application

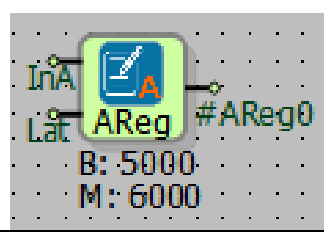


In samples

1- Word register which is 4000 block round, counter value which is in the I1 input to “Lat” input with the logic(1) signal which is comes from DI1 is taken to in the 4000 round block. (Edge type is selected as “High”)
2- The value is written as offline and online to in the 4001 block number word register.

9.2 ANALOG REGISTER

9.2.1 Connections

I1: Data Input		#AReg0: Analog Output
Lat: Latch Signal		

9.2.2 Connection Explanations

<u>I1: Data Input</u>

Data input which is latched into register.

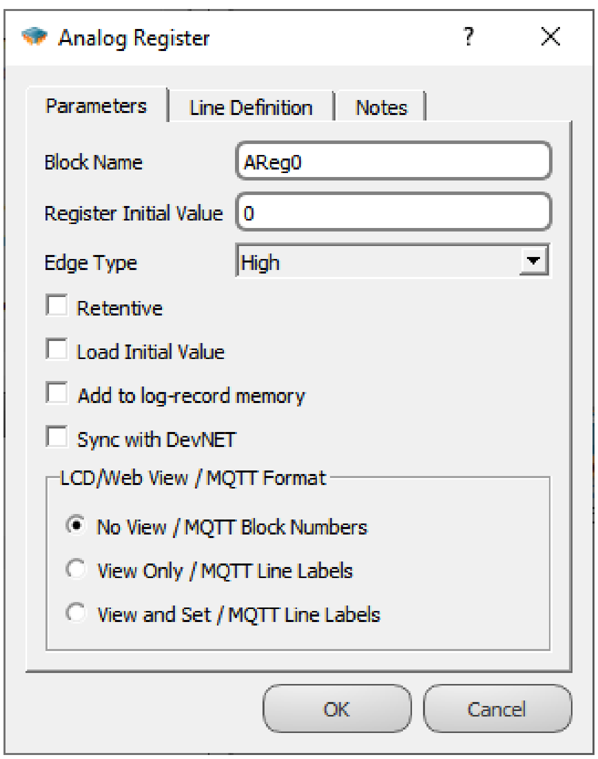
<u>Lat: Latch Signal</u>

Data is latched into the register memory within control of Lat signal..

<u>#AReg0: Analog output</u>

#AReg0 is Block output. It reflects the internal 32 bit Floating Point Analog Register value.

9.2.3 Block Settings

	Register Initial Value: The initial value which will be written in the register memory at startup.
	Edge Type: Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process. Edge Type Options: High, Low, Raise, Fall, Raise/Fall
	Persistence: If it is selected, register value is non-volatile even if the device power is off. Last value of the register is reloaded automatically after power on.
	Load Initial Value: Active only Persistence is selected. This is a selection between initial value coming from user project or last saved value coming from non-volatile momory as a initial value after new project is downloaded into device.

9.2.4 Block Explanations

Analog Register Block is used as a 32 bit Floating Point type value holder. It is used as variable in PLC projects.

Using the Lat Signal, the block can be used like a D-Type Latch.

Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process.

Possible "Edge type " options and usage are given at following table:

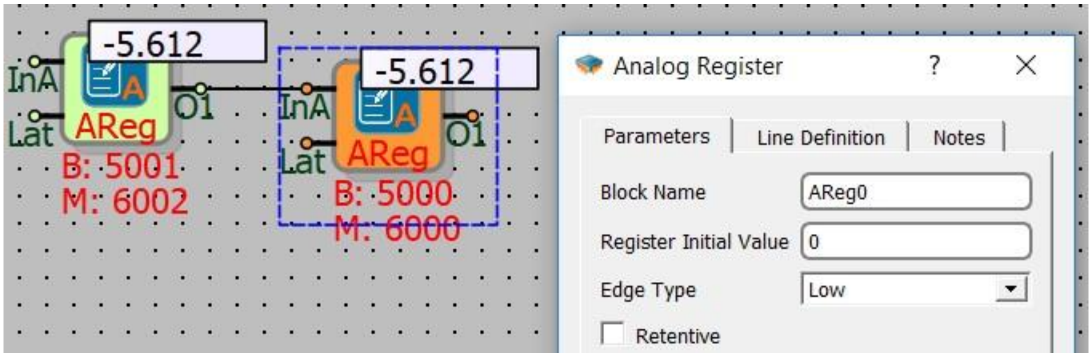
High	Only if the Lat Signal is Logic(1), Value at I1 input is saved into Register Memory
Low	Only if the Lat Signal is Logic(0), Value at I1 input is saved into Register Memory Note: if Lat signal is not connected, it means to Low – Logic(0)
Raise	Value at I1 input is saved into Register Memory when Raising edge of the Lat Signal
Fall	Value at I1 input is saved into Register Memory when Falling edge of the Lat Signal
Both	Value at I1 input is saved into Register Memory when Raising or Falling edge of the Lat Signal

I1 Data Input signal type may be different from register block type. For example, Word signal can be applied to Analog register block. In that case, Automatic variable casting occurs. Therefore, user must be pay attention to variable types.

Sample transformation table is given the below from different variable types for entiring value to the word register

It is the variable type in input	Sample Input Value	It is value which is will be loaded to the analog register
Binary	0	0.0
Binary	1.12	1.12
Word	12	12.0
Word	98.45	98.45
Long	65000	65000.0
Long	80000	80000.0

9.2.5 Sample Application

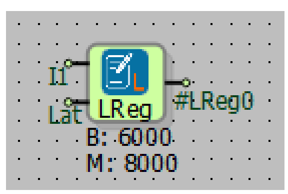


In the sample;

“-5.612 “ value was written as offline or online in to the analog register which is 5001 block number by the user. The output of block which is 6002 due to connected the 6000 block number “–5.612” value was written in to the analog register which is 6000 block number. (“Lat” input is given the blank because of “Edge Type is selected as “low”.)

9.3 LONG REGISTER

9.3.1 Connections

I1: Data Input		#LReg0: Long Output
Lat: Latch Signal		

9.3.2 Connection Explanations

<u>I1: Data Input</u>

Data input which is latched into register.

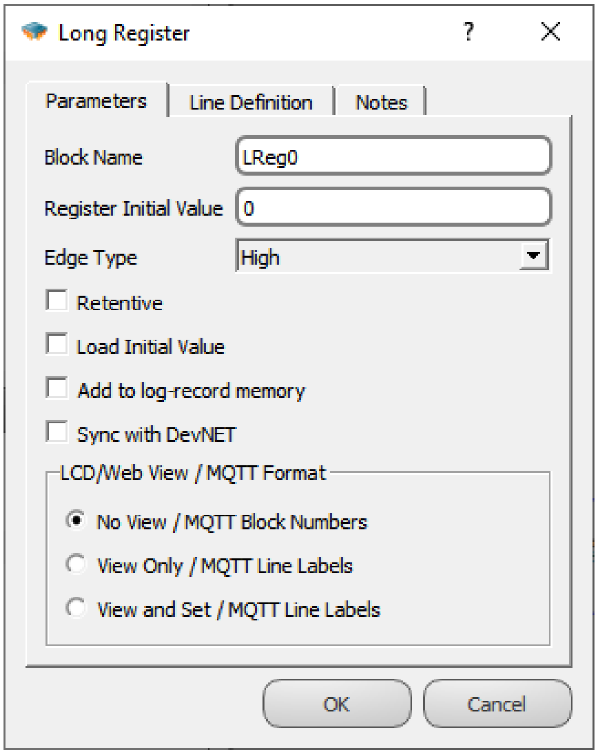
<u>Lat: Latch Signal</u>

Data is latched into the register memory within control of Lat signal.

<u>#LReg0: Long Output</u>

#LReg0 is Block output. It reflects the internal 32 bit signed Long Register value.

9.3.3 Block Settings

	Register Initial Value: The initial value which will be written in the register memory at startup.
	Edge Type: Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process. Edge Type Options: High, Low, Raise, Fall, Raise/Fall
	Persistence: If it is selected, register value is non-volatile even if the device power is off. Last value of the register is reloaded automatically after power on.
	Load Initial Value: Active only Persistence is selected. This is a selection between initial value coming from user project or last saved value coming from non-volatile momory as a initial value after new project is downloaded into device.

9.3.4 Block Explanation

Word Register Block is used as a 32 bit signed integer type value holder. It is used as variable in PLC projects.

Using the Lat Signal, the block can be used like a D-Type Latch.

Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process.

Possible “Edge type “ options and usage are given at following table:

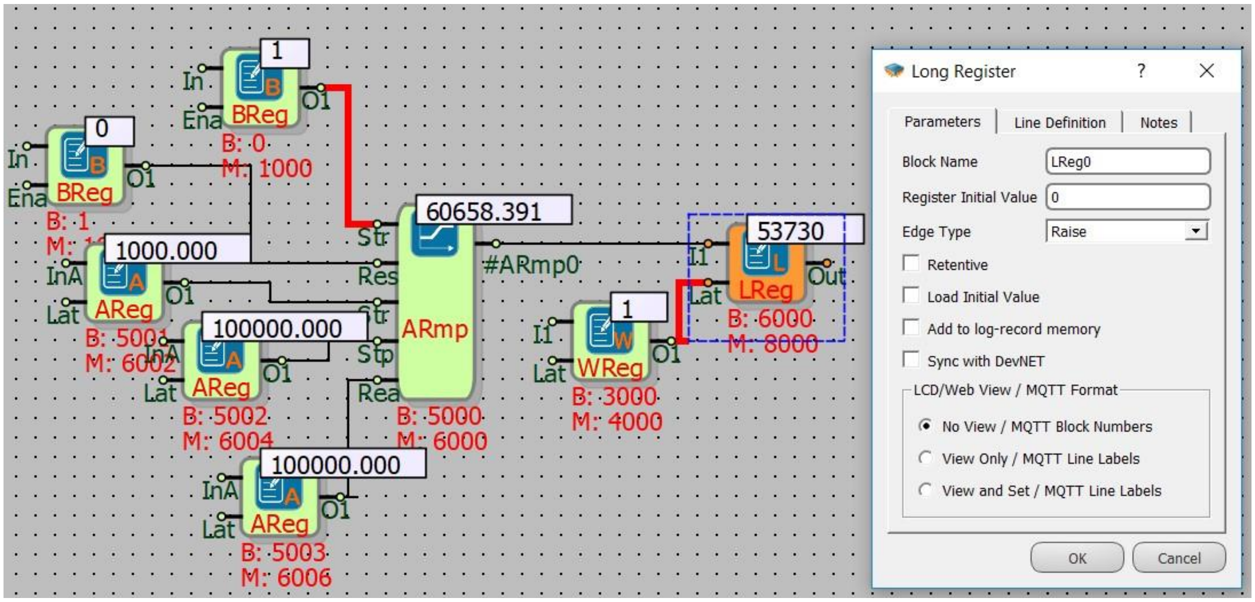
High	Only if the Lat Signal is Logic(1), Value at I1 input is saved into Register Memory
Low	Only if the Lat Signal is Logic(0), Value at I1 input is saved into Register Memory Note: if Lat signal is not connected, it means to Low – Logic(0)
Raise	Value at I1 input is saved into Register Memory when Raising edge of the Lat Signal
Fall	Value at I1 input is saved into Register Memory when Falling edge of the Lat Signal
Both	Value at I1 input is saved into Register Memory when Raising or Falling edge of the Lat Signal

I1 Data Input signal type may be different from register block type. For example, Analog signal can be applied to Long register block. In that case, Automatic variable casting occurs. Therefore, user must be pay attention to variable types.

Sample transformation table is given the below from different variable types for entiring value to the Long register.

It is the variable type in input	Sample Input Value	It is value which is will be loaded to the long register
Binary	0	0
Binary	1	1
Analog	12.34	12
Analog	98.9	98
Word	65000	65000

9.3.5 Sample Application

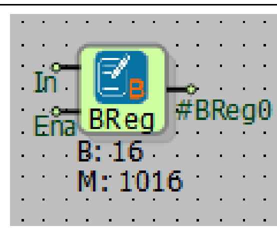


In the example:

Because of "Edge Type" of the "Long Register" is "Raise" selected , in each rising edge trigger to the Lat input, the value of the "Analog Ramp" is recorded in the "Long Register". (filtered after the comma)

9.4 BINARY REGISTERS

9.4.1 Connections

I1: Data Input		#BReg0: Binary output
Ena: Latch Signal		

9.4.2 Bağlantı Açıklamaları

<u>I1: Data Input</u>

Data input which is latched into register.

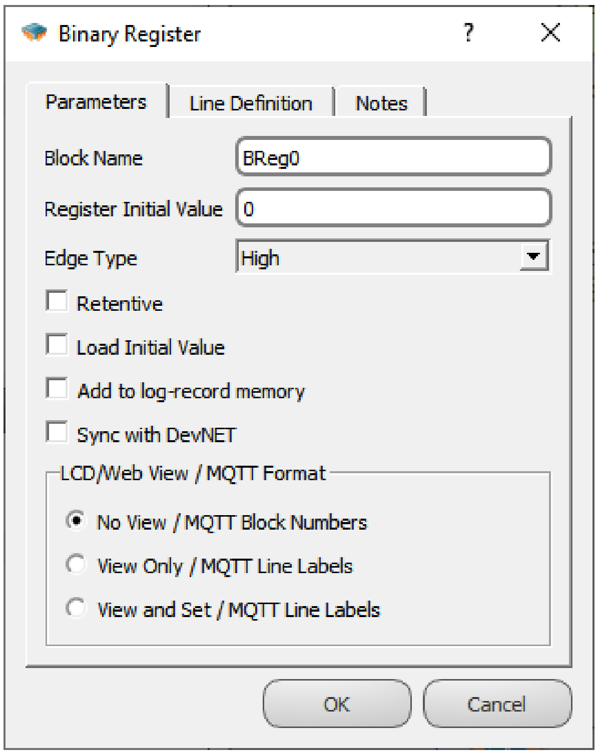
<u>Lat: Latch Signal</u>

Data is latched into the register memory within control of Lat signal.

<u>#BReg0: Binary output</u>

#BReg0 is block output. It reflects the internal 1 bit Boolean Register value.

9.4.3 Block Settings

	Register Initial Value: The initial value which will be written in the register memory at startup.
	Edge Type: Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process. Edge Type Options: High, Low, Raise, Fall, Raise/Fall
	Persistence: If it is selected, register value is non-volatile even if the device power is off. Last value of the register is reloaded automatically after power on.
	Load Initial Value: Active only Persistence is selected. This is a selection between initial value coming from user project or last saved value coming from non-volatile momory as a initial value after new project is downloaded into device.

9.4.4 Block Explanation

Binary Register Block is used as a 1 bit Boolean type value holder. It is used as variable in PLC projects.

Using the Lat Signal, the block can be used like a D-Type Latch.

Latching of I1 value into Register Memory is controlled by Lat Signal. The edge selection type determines how the Lat signal will control the Latching process.

Possible “Edge type “ options and usage are given at following table:

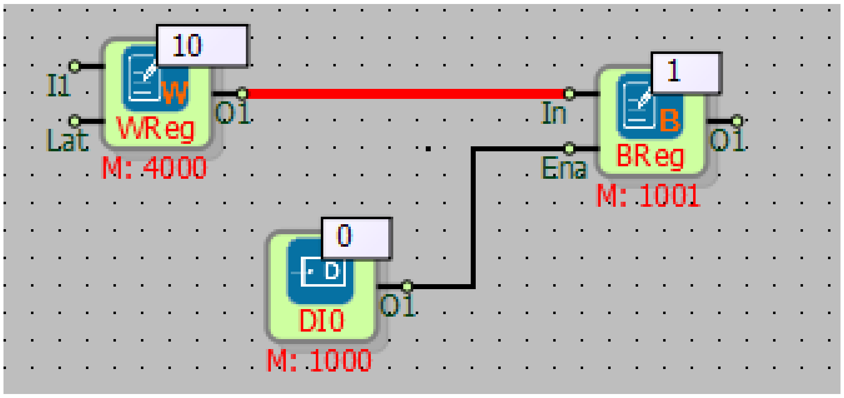
High	Only if the Lat Signal is Logic(1), Value at I1 input is saved into Register Memory
Low	Only if the Lat Signal is Logic(0), Value at I1 input is saved into Register Memory Note: if Lat signal is not connected, it means to Low – Logic(0)
Raise	Value at I1 input is saved into Register Memory when Raising edge of the Lat Signal
Fall	Value at I1 input is saved into Register Memory when Falling edge of the Lat Signal
Both	Value at I1 input is saved into Register Memory when Raising or Falling edge of the Lat Signal

I1 Data Input signal type may be different from register block type. For example, Analog signal can be applied to Binary register block. In that case, Automatic variable casting occurs. Therefore, user must be pay attention to variable types.

Sample transformation table is given the below from different variable types for entiring value to the word register

It is the variable type in input	Sample Input Value	It is value which is will be loaded to the binary register
Word	0	0
Word	234	1
Analog	0.001	1
Analog	-98.9	1
Long	0	0
Long	80000	1

9.4.5 Sample Application

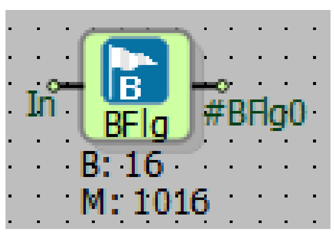


In the example;

Because of the name of binary register as “fall” is selected, every low edge trigger comes to Ena input, word register which ıts value is 10 was written to binary register as 1.

9.5 BINARY FLAG

9.5.1 Connections

In: Block input		#BFlg0: Block output

9.5.2 Connecrtion Explanation

In: Block input

It is block input.

#BFlg0: Block output

It is block output.

9.5.3 Block Settings

There are no block settings.

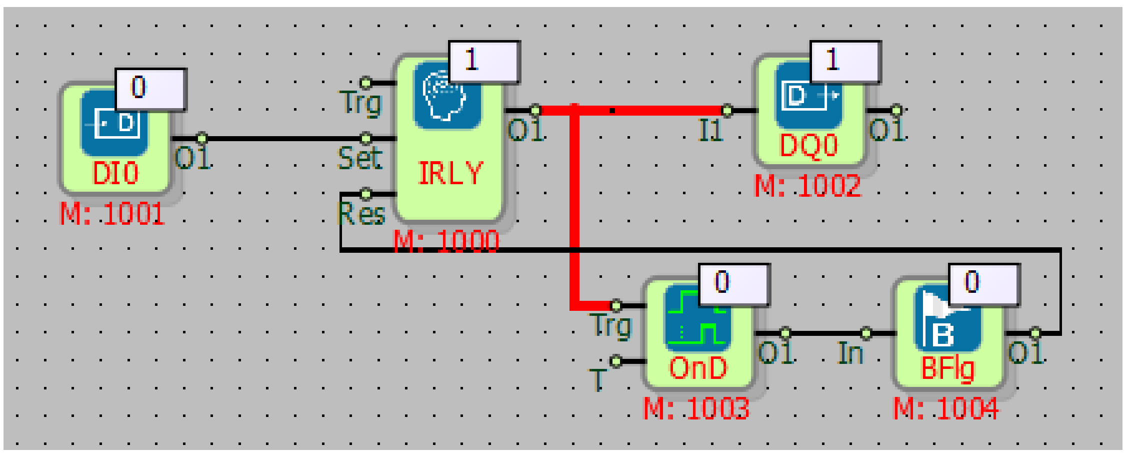
9.5.4 Block Explanation

The value in input signal is transmitted to the block output with one PLC cycle delay.

Flag register may be used to prevent logic operations from infinite logic loops when feedback is applied.

Binary Flags operate with 1 bit binary values.

9.5.5 Sample Application



In the example:

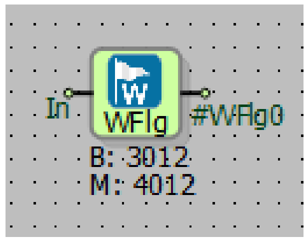
DI0 triggers the "Set" input of the "Pulse Relay" block and sets DQ0 to the logic (1) position, at the same time the pull delay is also triggered.

After delaying 3 second the draw, the binary flag has become logical (1), resetting the "Pulse Relay", DQ0 has taken to logical (0) position.

The binary flag is used to prevent "feedback error".

9.6 WORD FLAG

9.6.1 Connections

In: Block input		#WFlg0: Block output

9.6.2 Connection Explanations

In: Block input

It is the block input.

#WFlg0: Block output

It is block output.

9.6.3 Block Settings

There are no block settings.

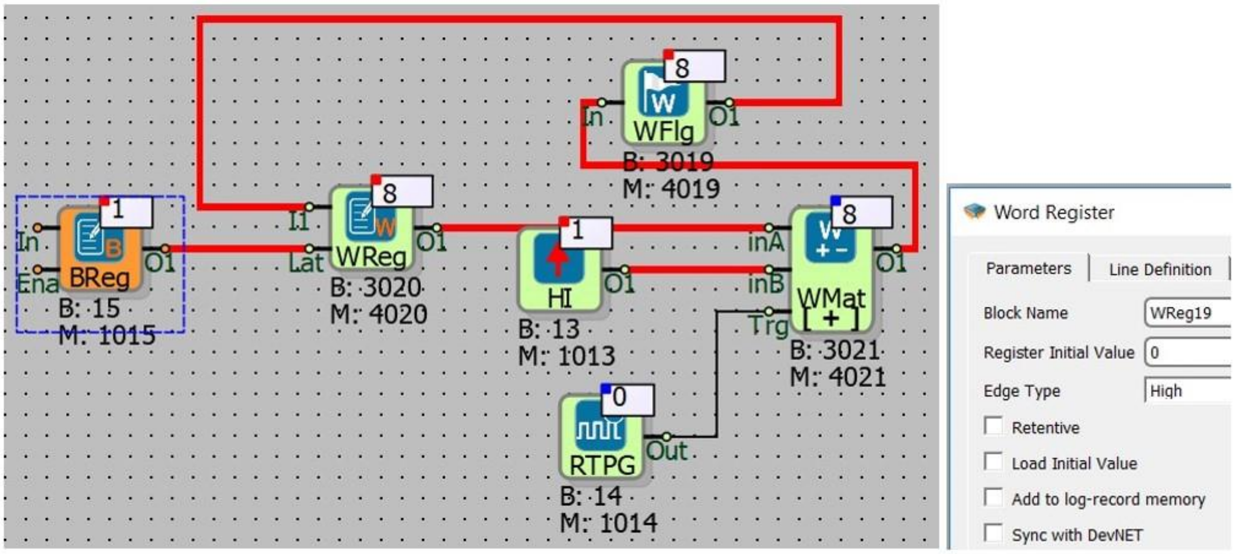
9.6.4 Block Explanation

The value in input signal is transmitted to the block output with one PLC cycle delay.

Flag register may be used to prevent logic operations from infinite logic loops when feedback is applied. This is not permitted, as this will cause an infinite loop in the PLC logic loop. In the logic where feedback is required, flag blocks are added to the feedback line to prevent an infinite loop error.

Word Flags operate with 16 bit unsigned values.

9.6.5 Sample Application



In the example; A 16-bit counter is designed.

As soon as the binary register has value 1, the counter starts to increase.

The GZDU block is programmed to produce 1 trigger per second. The output of the Word Math block is linked back to the Word Math block I1 entry with Word Flag.

Because of the "Word Register" "Edge Type" is "High", Binary Register has to be 1, for increasing the counter.

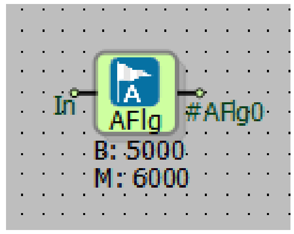
As soon as the Binary Register has a value of 1, the value of the "High Gate" is transferred to the "Word Flag" and then to the Word Register.

"Word Math" block, INB input and the INA input is added in each trigger of the GZDU block. Then new value is written in "Word Register".

So 16-bit counter has been designed.

9.7 ANALOG FLAG

9.7.1 Connections

In: Block input		#AFlg0: Block output

9.7.2 Connection Explanations

In: Block input

It is block input.

#AFlg0: Block output

It is block output.

9.7.3 Block Settings

There are no block settings.

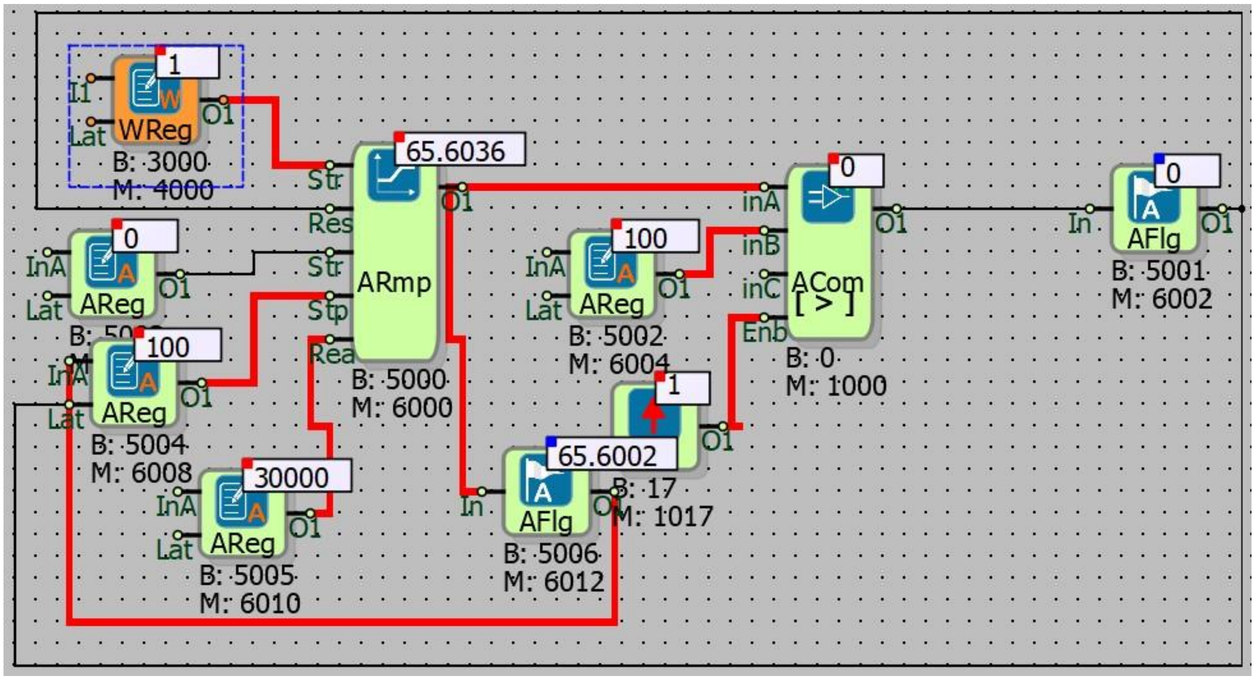
9.7.4 Block Explanation

The value in input signal is transmitted to the block output with one PLC cycle delay.

Flag register may be used to prevent logic operations from infinite logic loops when feedback is applied.

Analog Flags operate with 32 bit floating point values.

9.7.5 Sample Application



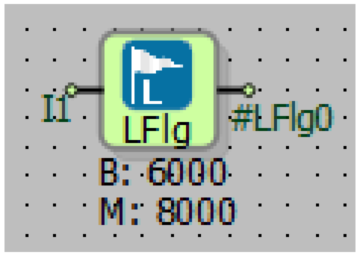
In the example;

“Stop Value” of the Analog Ramp is updated with Analog Flag 5006.

Analog Ramp is reset after the value of the output of the Analaog Comparator with Analog Flag 5000 has passed the threshold value. The ramping process has been restarted by the new Stop Value.

9.8 LONG FLAG

9.8.1 Connections

I1: Block input		#LFlg0: Block output

9.8.2 Connection Explanations

<u>I1: Block input</u>

It is block input

<u>#LFlg0: Block output</u>

It is block output

9.8.3 Block Settings

There are no block settings

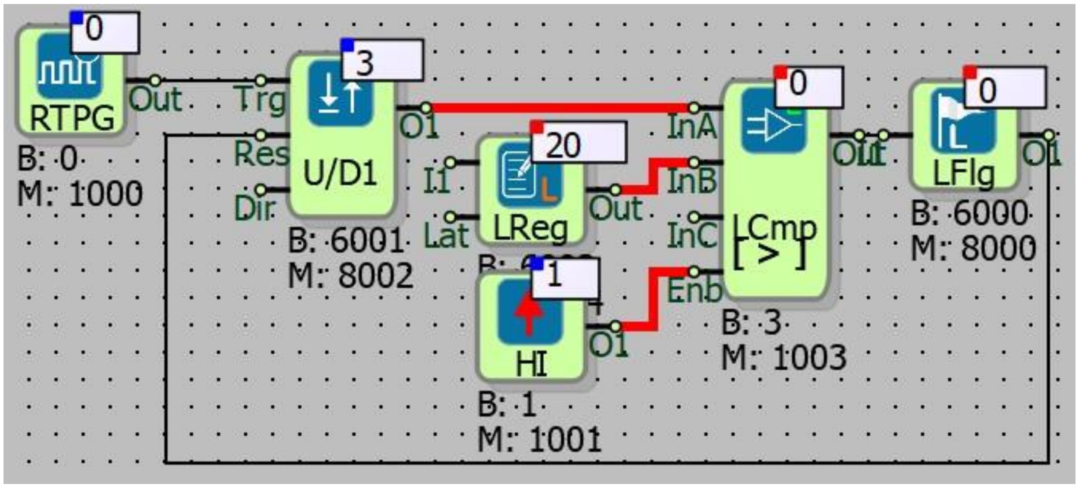
9.8.4 Block Explanation

The value in input signal is transmitted to the block output with one PLC cycle delay.

Flag register may be used to prevent logic operations from infinite logic loops when feedback is applied.

Word Flags operate with 32 bit signed integer values..

Sample Application



In the example;

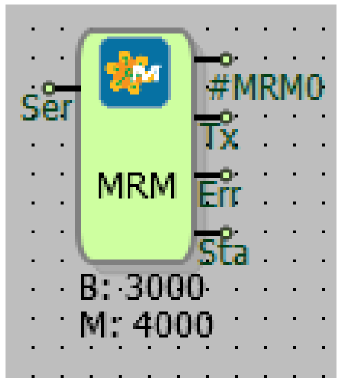
The value of Up / Down Counter is reset, when it reaches a certain value that is written on InB input of the Long Comparator. When the value of the "Long Comparator" InB input is exceeded, the block output is set to 1. Then "Long Flag" resets the Up / Down Counter after a PLC cycle time delay.

Note: In the example, because the output of the Long Comparator is binary, other flag types (word, analog, bit) can be used too.

10 MODBUS PROTOCOL BLOCKS

10.1 MODBUS RTU MASTER

10.1.1 Connections

Ser: Serial port block input		#MRM0: Block output
		Tx: Tx Value output
		Err: Number of errors in submitted requests
		Sta: Connection state output

10.1.2 Connection explantations

Ser: Serial port block input

It is the block input which will be connected to the communication port.

#MRM0: Blok output

Block’s output connection

Tx: Tx value output

It is the output connection where the number of requests sent is read

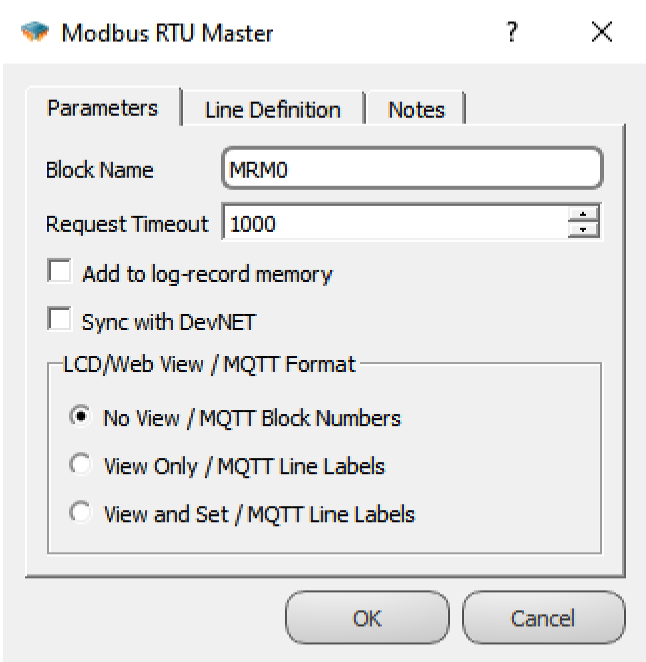
Err: Number of errors in submitted requests

It is the output connection where the error count of sent requests is read

Sta: Connection state output

State of the last executed request

10.1.3 Block Settings

<table>
<tr>
<td>
</td>
<td>Request Timeout: Determines the reply's timeout duration</td>
</tr>
</table>

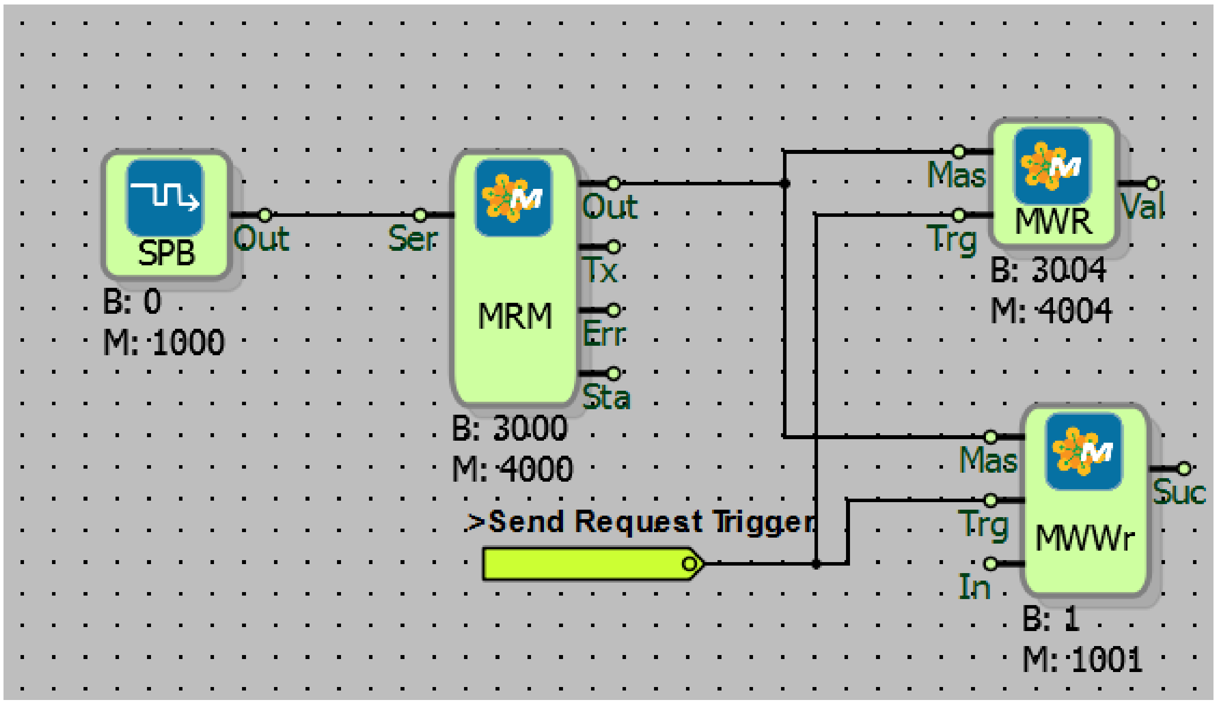
10.1.4 Block Explanation

Modbus RTU Master block activates the Modbus RTU Master protocol on physical interface connected over communication port input. Standart Modbus RTU Master block operates on RS485 or RS232 serial port. Since only one Modbus RTU Master block is possible on a RS485 bus, only one Modbus RTU Master block can be opened on each serial channel. A Modbus RTU Master block can be added per port to a device which have more than one RS485 ports.

After the protocol is actived with Modbus RTU Master block; as a final step you need to connect “request send blocks” to Master block. Generally, requests are grouped as reading and writing in the Modbus protocol. When Modbus request blocks which are used for reading and writing are triggered, the request is added to the queue on Master Block. If the RS485 line is idle, the requests in the queue on Master Block are sent one by one and response is waited. If a response is received before “timeout” duration, the reply is processed, if no reponse is received the request is canceled and error counter is increased by one. Here “timeout” duration is defined in master block’s settings section.

Modbus messages are instantenous reading/writing requests and they do not contain any time tag information. Therefore, request queue on master block has smart mechanisms that provides only keeping the latest request on queue regarding to a point.

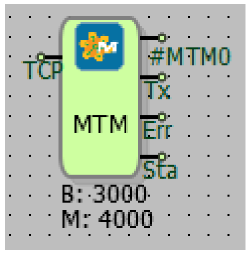
10.1.5 Sample Application



On the serial port Modbus RTU Master protocol is actived. The device as a Modbus RTU Master block sends reading and writing requests to slave devices.

10.2 MODBUS TCP MASTER

10.2.1 Connections

TCP: Block input		#MTM0: Block output Tx: Tx value output Err: Error value output Sta: Connection status output

10.2.2 Connection Explanations

TCP: Block input

The block input connection to which the communication port is connected.

#MTM0: Block output

The block output connection.

Tx: Tx value output

It is the output connection which indicates the number of requests sent

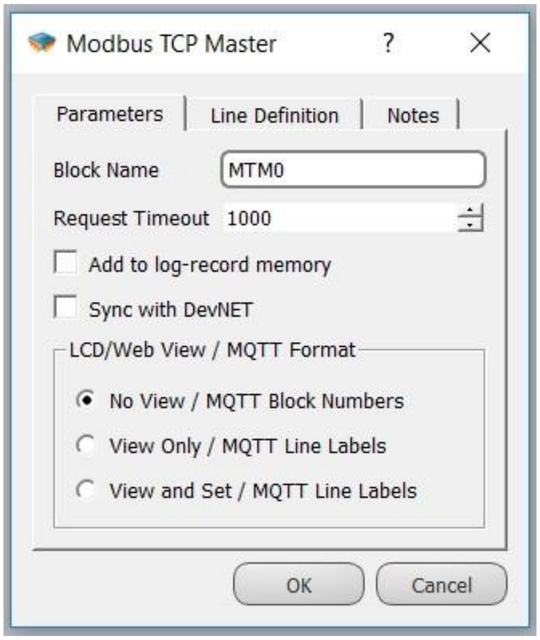
Err: Number of errors in submitted requests

It is the output connection which indicates the error count of the sent requests

Sta: Connection status output

Indicates if the last executed request is succesful or not.

10.2.3 Block Settings

	Request Timeout: This is the value which determines the response time.

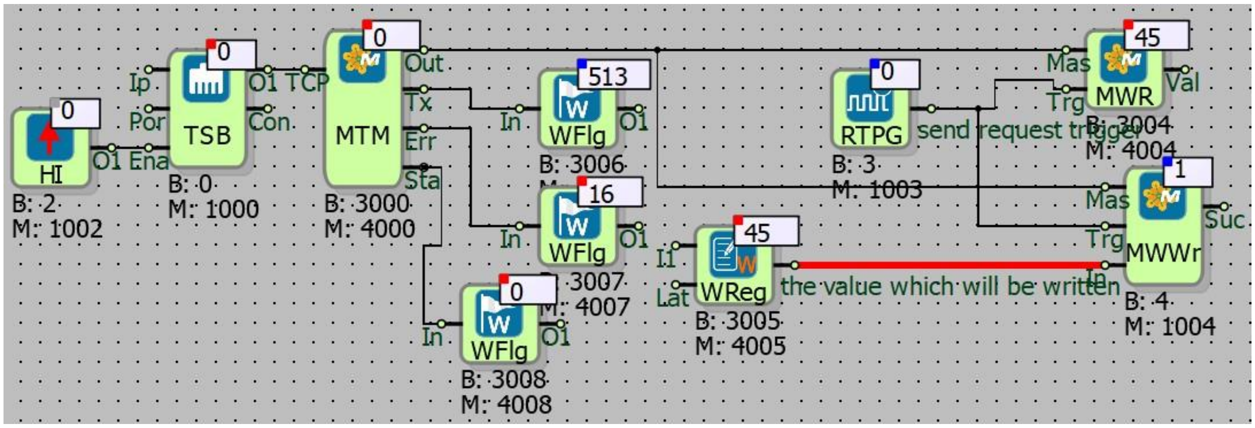
10.2.4 Block Explanation

The Modbus TCP Master block activates the Modbus TCP Master protocol on physical interface connected over communication port input.

After the protocol is actived with Modbus TCP Master block; as a final step you need to connect “request send blocks” to Master block. Generally, requests are grouped as reading and writing in the Modbus protocol. When Modbus request blocks which are used for reading and writing are triggered, the request is added to the queue on Modbus TCP Master Block. If the RS485 line is idle, the requests in the queue on Modbus TCP Master Block are sent one by one and response is waited. If a response is received before “timeout” duration, the reply is processed, if no reponse is received the request is canceled and error counter is increased by one. Here “timeout” duration is defined in master block’s settings section.

Modbus messages are instantenous reading/writing requests and they do not contain any time tag information. Therefore, request queue on master block has smart mechanisms that provides only keeping the latest request on queue regarding to a point.

10.2.5 Sample Application



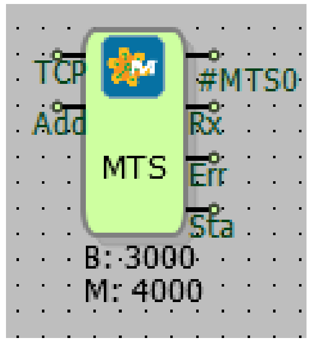
Modbus TCP Master protocol is actived on TCP socket. The device as an TCP Master sends reading and writing requests to slave devices.

It is necessary to connect the Modbus TCP Master block Out output to the corresponding “Mas” inputs of the Modbus Reader / Writer blocks.

If the data packet is transmitted / received successfully, the Sta output is 0 and if not, it is 1.

10.3 MODBUS TCP SLAVE

10.3.1 Connections

<table>
<tr><td rowspan="2">TCP: Block Input</td><td rowspan="4">
</td><td>Out: Block output</td></tr>
<tr><td>Rx: Rx value output</td></tr>
<tr><td rowspan="2">Add: Modbus ID input</td><td>Err: Error value output</td></tr>
<tr><td>Sta: Connection status output</td></tr>
</table>

10.3.2 Connection Explanations

<u>TCP: Block input</u>

The block input connection to which communication port is connected

<u>Add: Modbus ID input</u>

Used to identify the Modbus ID address externally

<u>Out: Block output</u>

The output connection of the block

<u>Rx: Rx value output</u>

It is the output link that shows the number of incoming requests.

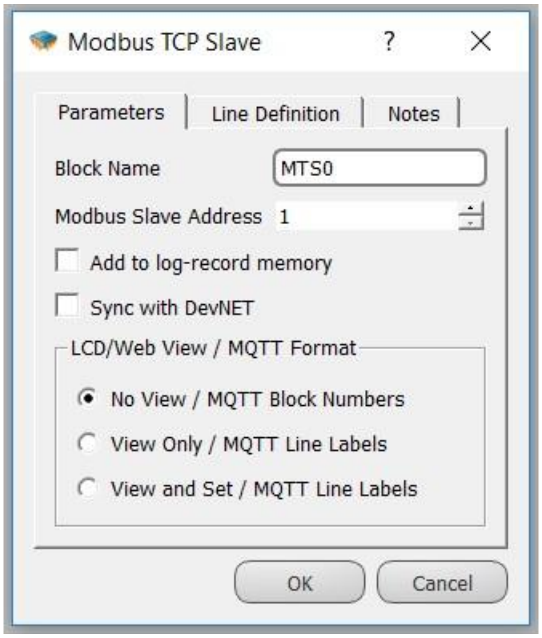
<u>Err: Error value output</u>

It is the output connection which indicates the error count of the requests sent.

<u>Sta: Connection status output</u>

Indicates the success state of the last executed request.

10.3.3 Block Settings

	Modbus RTU Slave: The ID of the slave device to be connected.

10.3.4 Block Explanation

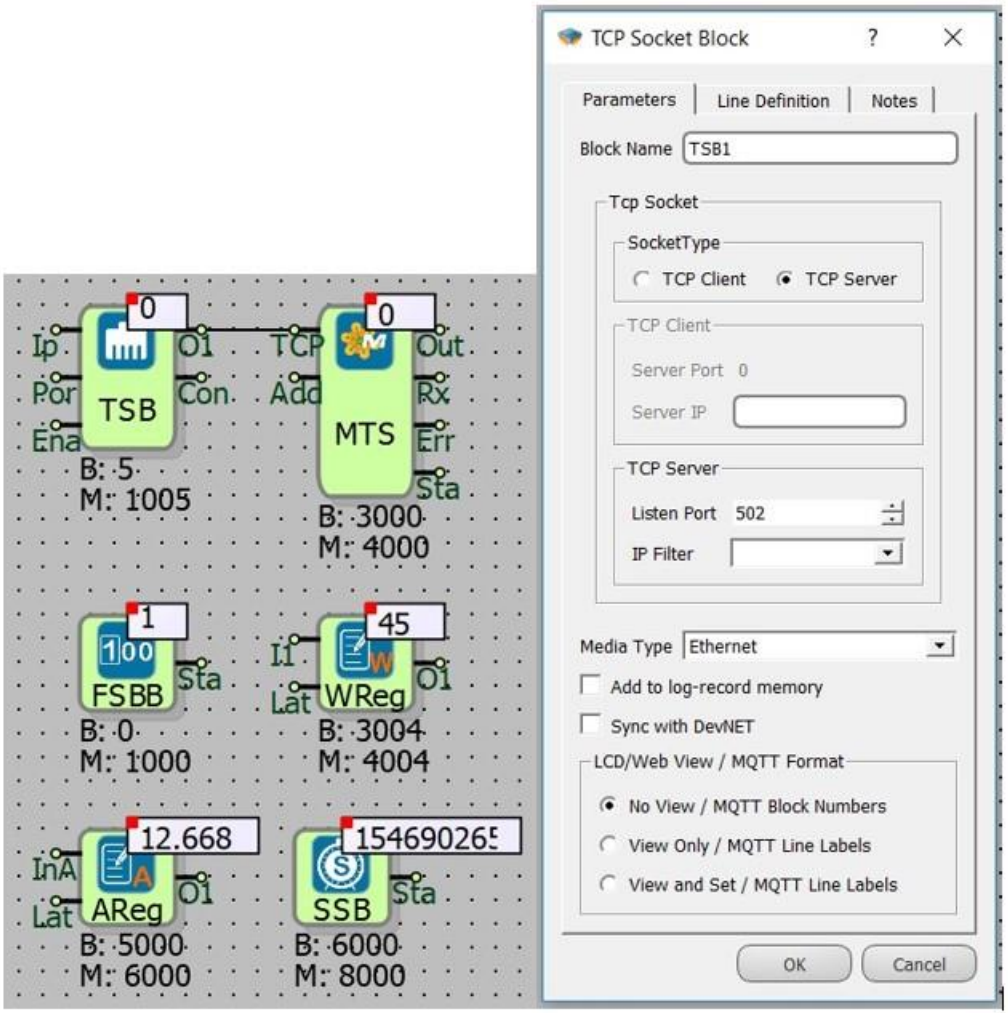
The Modbus TCP Slave block activates the Modbus TCP Slave protocol on physical interface connected over communication port input.

The device activated as a Modbus TCP Slave responds to requests with its own Modbus Id from the defined communication port.

All blocks in the logic project and the Modbus addresses defined in the variable address table will now be accessible with these channel and protocol settings

Block Name	Register Adress	Function Code
Binary - Binary Blocks	1000	(0x01) Read Coils (0x02) Read Discrete Inputs (0x05) Write Single Coil (0x0F) Write Multiple Coils
Word Blocks	4000	(0x03) Read Holding Registers (0x04) Read Input Registers (0x06) Write Single Register (0x10) Write Multiple registers
Analog Blocks	6000	(0x03) Read Holding Registers (0x04) Read Input Registers (0x06) Write Single Register (0x10) Write Multiple registers
Long Blocks	8000	(0x03) Read Holding Registers (0x04) Read Input Registers (0x06) Write Single Register (0x10) Write Multiple registers

10.3.5 Sample Application

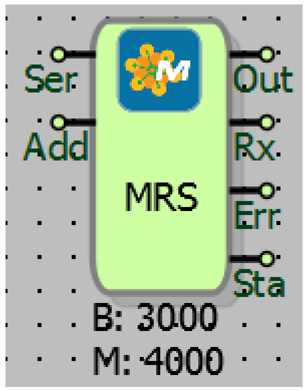


TCP Socket Block is selected as Server, Modbus TCP Slave block is connected to the block output and in this way the device is programmed in Server mode. (Connection type is selected as Ethernet.)

A device that is programmed in this way can be connected by another Modbus TCP Client.

10.4 MODBUS RTU SLAVE

10.4.1 Connections

<table>
<tr><td rowspan="2">Ser: Block input</td><td rowspan="4">
</td><td>Out: Block output</td></tr>
<tr><td>Rx: Rx value output</td></tr>
<tr><td rowspan="2">Add: Modbus ID input</td><td>Err: Error value output</td></tr>
<tr><td>Sta: Connection status output</td></tr>
</table>

10.4.2 Connection Explanations

Ser: Block input

The block input to which the communication port is connected.

Add: Modbus ID input

Used to identify the Modbus ID address externally

Out: Block output

Output connection of the block.

Rx: Rx value output

It is the output link that shows the number of incoming requests.

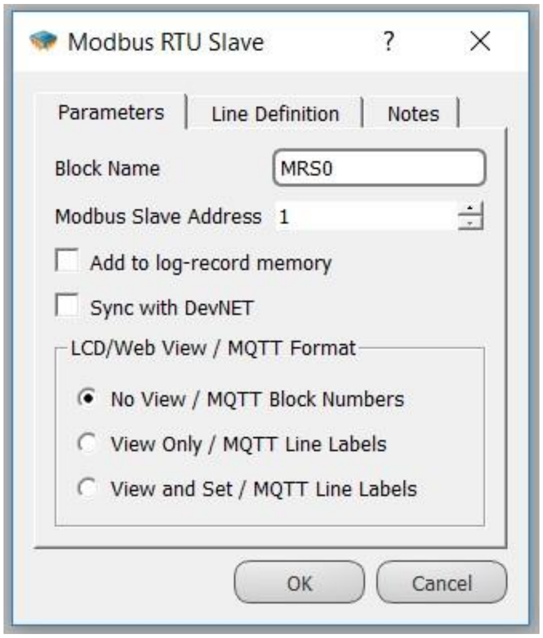
Err: Error value output

It is the output connection which indicates the error count of the submitted requests

Sta: Connection status output

Indicates the success state of the last executed request.

10.4.3 Block Settings

	Modbus Slave Adress: The ID of the slave device to be connected.

10.4.4 Block Explanation

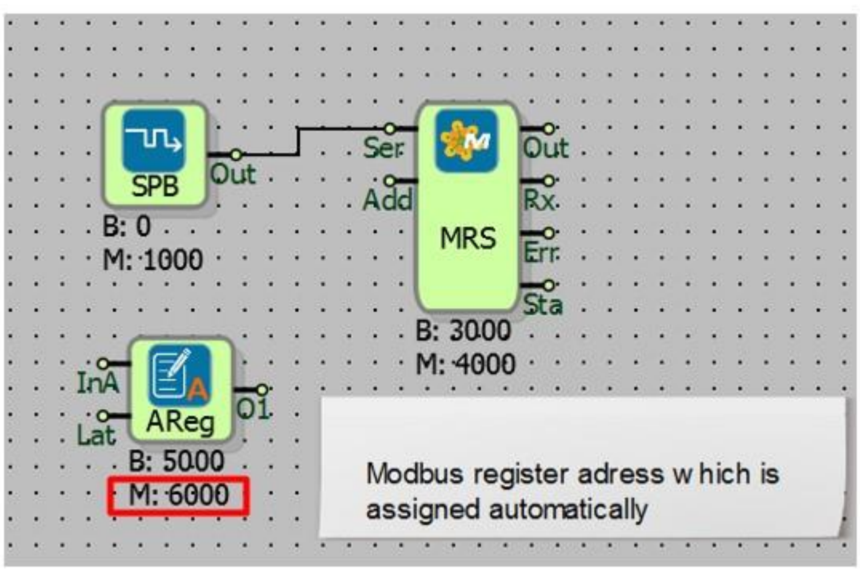
The MODBUS RTU Slave block activates the MODBUS RTU Slave protocol on physical interface connected over communication port input.

The device activated as a MODBUS RTU Slave responds to requests with its own MODBUS Id from the defined communication port.

All blocks in the logic project and the Modbus addresses defined in the variable address table will now be accessible with these channel and protocol settings

Blok Name	Modbus slave adress	Function Name
Two – Binary Blocks	1000	(0x01) Read Coils (0x02) Read Discrete Inputs (0x05) Write Single Coil (0x0F) Write Multiple Coils
Word Blocks	4000	(0x03) Read Holding Registers (0x04) Read Input Registers (0x06) Write Single Register (0x10) Write Multiple registers
Analog Blocks	6000	(0x03) Read Holding Registers (0x04) Read Input Registers (0x06) Write Single Register (0x10) Write Multiple registers
Long Blocks	8000	(0x03) Read Holding Registers (0x04) Read Input Registers (0x06) Write Single Register (0x10) Write Multiple registers

10.4.5 Sample Application



10.5 MODBUS GATEWAY BLOCK

10.5.1 Connections

Mas: Master input	v 
Sla: Slave input	

10.5.2 Connection Explanations

Mas: Master Input

Modbus TCP Master block reference input

Sla: Slave Input

Modbus RTU Slave block reference input

10.5.3 Block Settings

There is no block settings.

10.5.4 Block Explanations

Basically, MODBUS Gateway devices are used to create a gateway for master units in the MODBUS TCP network to access slave units in the MODBUS RTU network. Request packets coming from MODBUS TCP network are converted into MODBUS RTU packets and sent to RTU network. It also receives the response from the RTU network and sends it to the MODBUS TCP network. On the MODBUS TCP side, the number of requests and replies in the TRANSACTION must be the same. This is again the responsibility of the GATEWAY device.

Mikrodev Control Devices can be programmed as a GATEWAY between supported protocols. MODBUS GATEWAY block is one of the blocks used for this purpose.

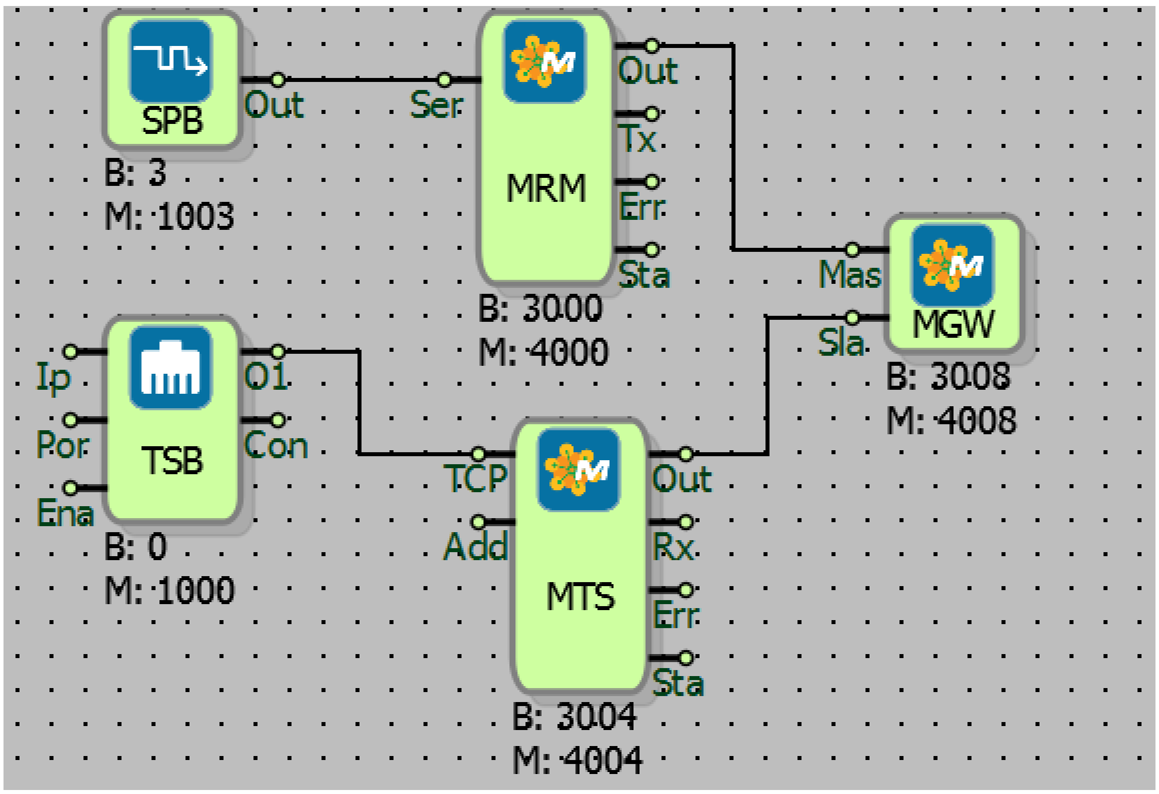
MODBUS GATEWAY block operates in both directions as below.

1-MODBUS TCP Master device to MODBUS RTU Slave device

2-MODBUS RTU Master device to MODBUS TCP Slave device.

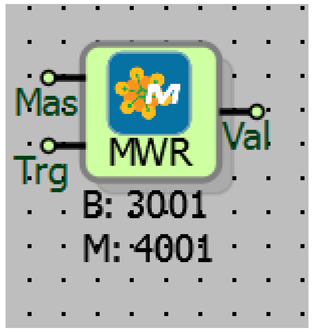
Connecting Master and Slave blocks is enough to operate as GATEWAY. If a request for a different ID is received from the slave block, the corresponding request will be read via the master block.

10.5.5 Sample Application



10.6 MODBUS WORD READER

10.6.1 Connections

Mas: Master input		Val: Block output
Trg: Trigger input		

10.6.2 Connection Explanations

Mas: Master input

It is master input connection.

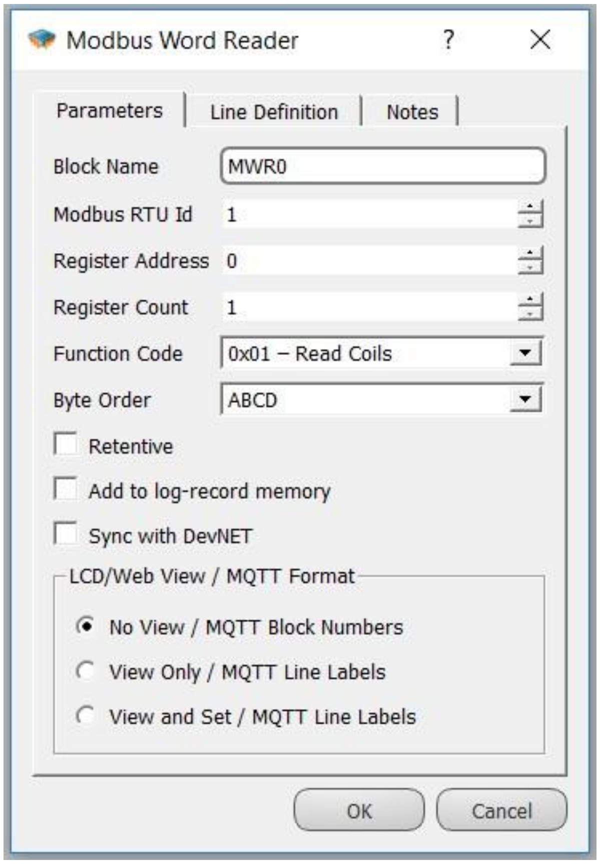
Trg: Trigger input

Trigger input connection.

Val: Block output

It is block output.

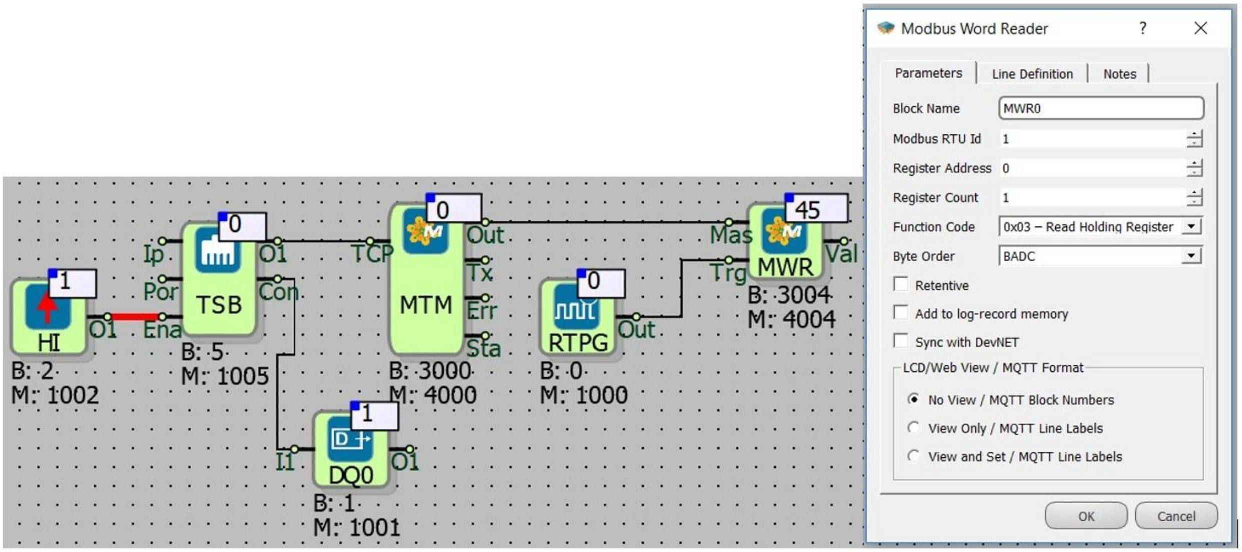
10.6.3 Block Settings

	Modbus RTU ID: Determines the ID, the data to be retrieved.
	Register Address: Register addresses to be read from slave IDs.
	Register Count: The number of registers to be read after the entered register address
	Function Code: The function code which will be selected to read the data.
	Byte Order: Determines in which byte order the data will be read.

10.6.4 Blok Explanation

It is used to read a single 16-bit length MODBUS register adress. Reading request is created on Trg signal's high edge, is added to request queue in MASTER block.

10.6.5 Sample Application

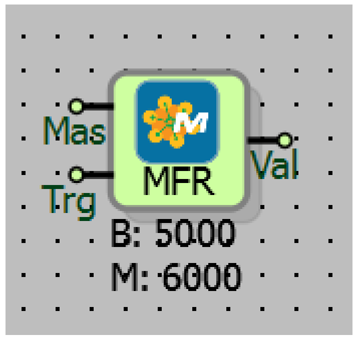


The MODBUS TCP Master protocol is used to read data from a MODBUS slave device. The MODBUS master protocol is activated on the device by connecting TCP socket block to the Modbus Master.

The reference connection from the MODBUS master block is connected to the reader blocks, and so the MODBUS master channel is selected to direct the reading requests. With every rising edge trigger signal coming to the Trg input of the MODBUS reader, the read request is added to the request queue of the master block. In cases where the master block communication channel is available and is not in a waiting state for the previous request, the requests in the queue will run sequentially.

10.7 MODBUS FLOAT READER

10.7.1 Connections

Mas: Master input		Val: Block output
Trg: Trigger input		

10.7.2 Connection Explantation

Mas: Master input

Master input connection.

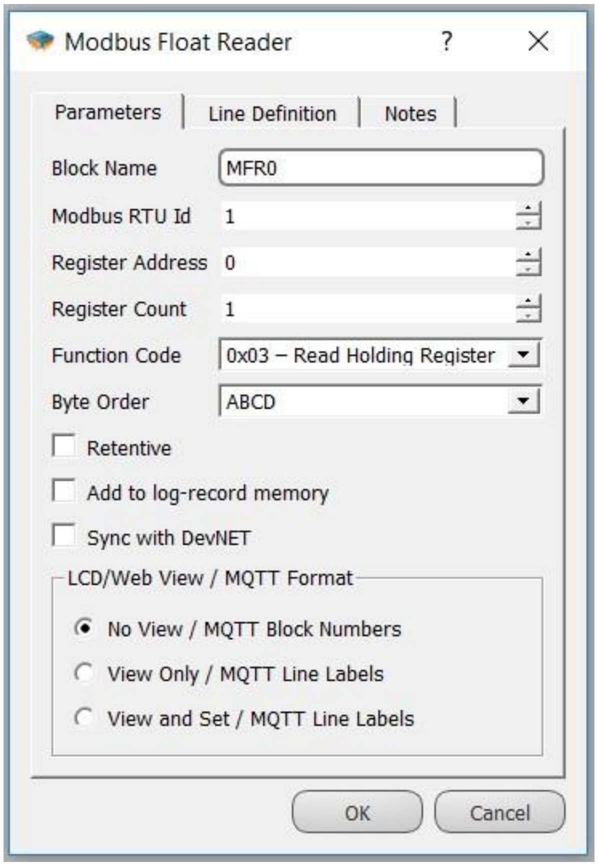
Trg: Trigger input

Trigger input connection.

Val: Block output

Block output connection.

10.7.3 Block Settings

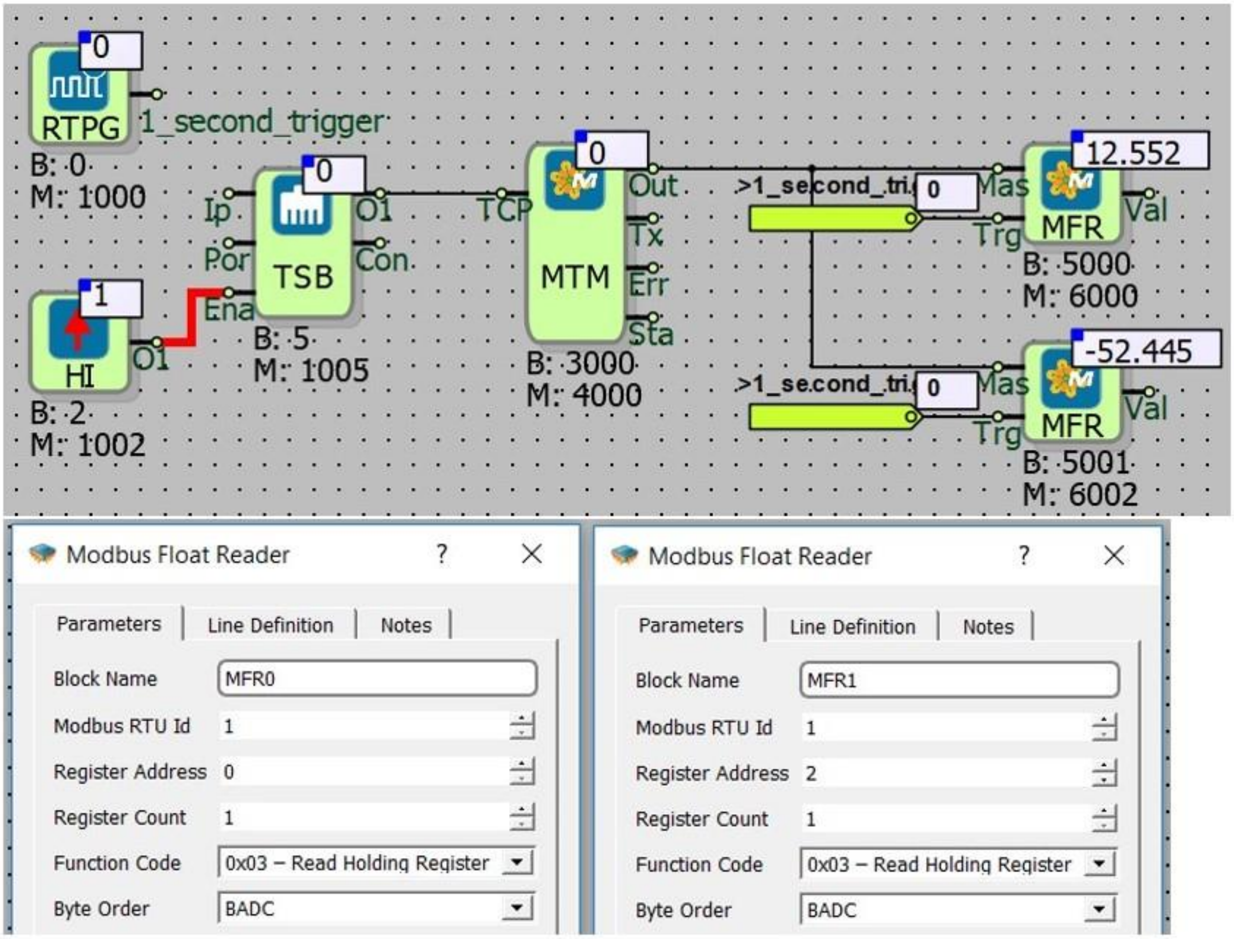


Modbus RTU Id: The value from which the data is to be retrieved
Register Adress: Register addresses to be read from slave Ids
Register Count: The number of registers to be read after the entered register address
Function Code: The function code which will be selected to read the data
Byte Order: The byte order of the data

10.7.4 Block Explanation

It is used for reading from 2 MODBUS registers which is storing 32 bits long IEEE 754 float number. Reading request is created at high edge on Trg input and is added to Master block's request queue. In cases where the Master block communication channel is available and in the case of no response waiting for the previous request, the requests in the request queue will run in order.

10.7.5 Sample Application



In the sample;

The values of 2 Float variables on another Modbus Server were read. Float Reader block Object Addresses are 0 and 2.

Two byte data is kept at 1 address. Since the float addresses are 2 bytes, 1 float data is read from 2 addresses (1 float data is read from the address 0 and 1, and 1 float is read from the 2nd and 3rd address.)

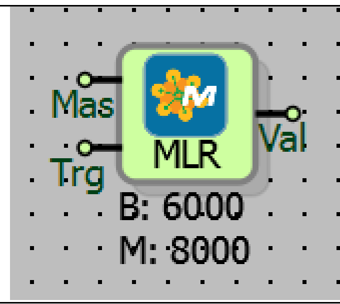
Because of the float variables can carry signed and decimal numbers, negative decimal (-x, yz) and positive decimal (+ x, yz) 32 bit values can be read.

In order to make the reading process;

1- TCP socket block client must be selected.
2- The Server IP and Port to be connected in the TCP socket block must be the same as the server.
3- TCP socket block “Ena” input must be set to logic1.
4- Float Reader block Trigger input signal must be given to the trailing edge trigger signal. (It should be noted that every rising edge trigger is a reading.)
5- Float reader block Object Properties, Modbus ID of the server to be connected must be entered.
6- The desired variable to be read, the function code and byte order of the variable must not be selected incorrectly.

10.8 MODBUS LONG READER

10.8.1 Connections

Mas: Master input		Val: Block output
Trg: Trigger input		

10.8.2 Connection Explanations

Mas: Master input

Master input connection.

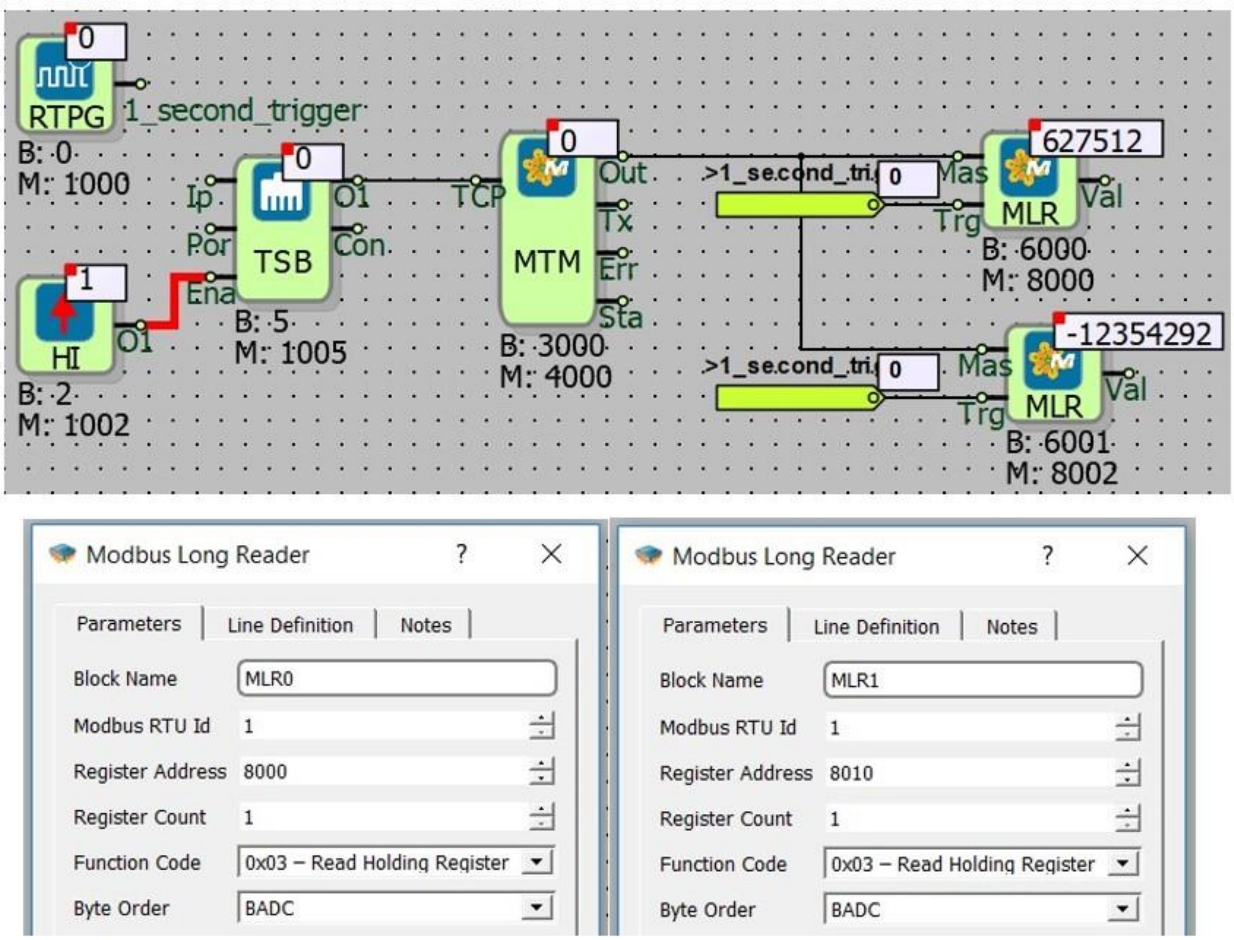
Trg: Trigger input

The trigger input connection.

Val: Block output

Block output connection.

10.8.3 Sample Application



In the sample;

The values of 2 Long variables on another Modbus Server were read. Long Reader block Object Addresses are 8000 and 8010.

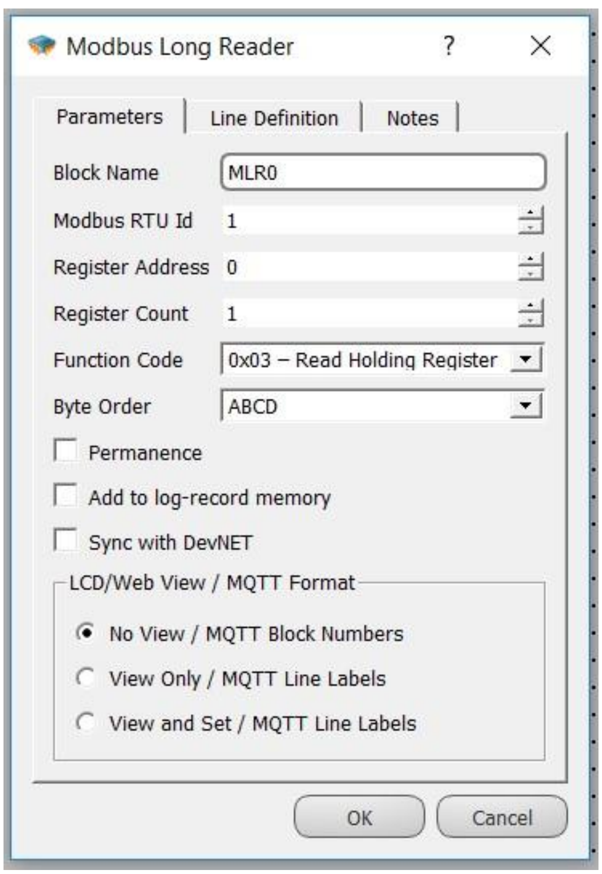
1 byte data is kept at 1 address. Since the Long addresses are 2 bytes, 1 Long data is read from 2 addresses (1 Long data is read from the address 8000 and 8001, and 1 Long data is read from the 8010. and 8011. addresses.)

Because of the Long variables can carry signed numbers, negative (-) and positive (+) 32 bit values can be read.

In order to make the reading process;

1- TCP socket block client must be selected.
2- The Server IP and Port to be connected in the TCP socket block must be the same as the server.
3- TCP socket block “Ena” input must be set to logic1.
4- Long Reader block Trigger input signal must be given to the trailing edge trigger signal. (It should be noted that every rising edge trigger is a reading.)
5- Long reader block Object Properties, Modbus ID of the server to be connected must be entered.
6- The desired variable to be read, the function code and byte order of the variable must not be selected incorrectly.

10.8.4 Block Settings



(screenshot above)	Modbus RTU Id: The id of the device from which the data is to be retrieved
	Register Adress: Register address to be read from slave Ids
	Register Count: The number of registers to be read after the entered register address
	Function Code: Function code which will be selected to read the data.
	Byte Order: The byte order of the data

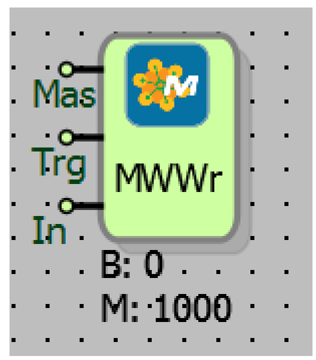
10.8.5 Block Explanation

The long of 32 byte which keep two numbers from long type fort to read the register adress.

Reading request is created on Trg signal’s high edge and added to Master block’s request queue. In cases where the master block communication channel is available and is not in a waiting state for the previous request, the requests in the queue will run sequentially.

10.9 MODBUS WORD WRITER

10.9.1 Connections

Mas: Master input	
Trg: Trigger input	
In: Block input	

10.9.2 Connection Explanations

Mas: Master input

Master input connection.

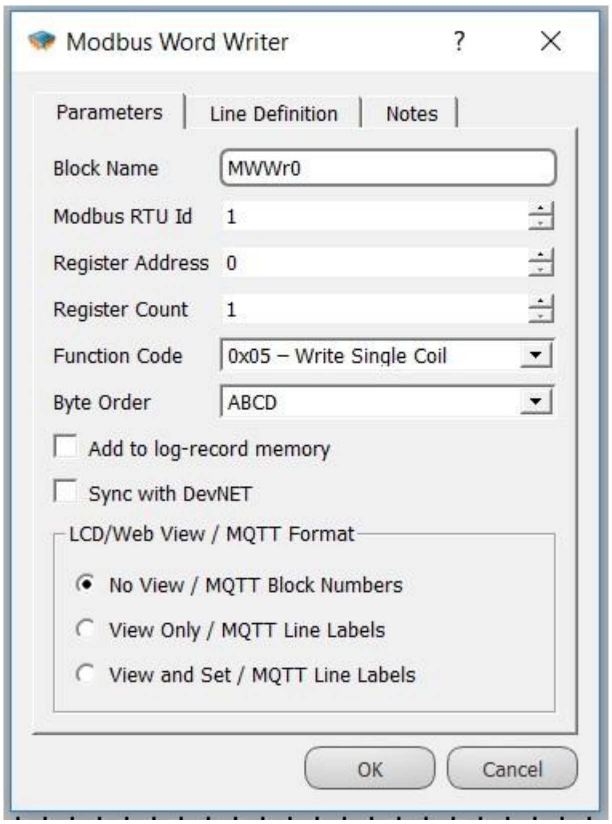
Trg: Trigger input

The trigger input connection.

In: Block input

Block input connection.

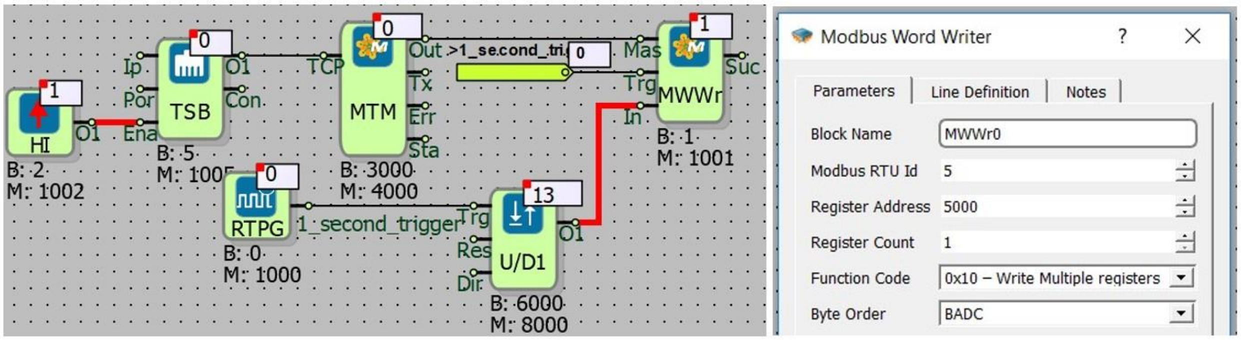
10.9.3 Block Settings

<table>
<tr><td rowspan="5">
</td><td>Modbus RTU ID: The ID of the device from which the data is to be retrieved.</td></tr>
<tr><td>Register Adress: Register address to be read from slave IDs.</td></tr>
<tr><td>Register Count: The number of registers to be read after the entered register address</td></tr>
<tr><td>Function Code: Function code which will be selected to write the data.</td></tr>
<tr><td>Byte Order: The byte order in which the data is written is determined</td></tr>
</table>

10.9.4 Block Explanation

It is used for writing on a single 16 bits long MODBUS register address. Writing request is created on Trg signal’s high edge and added to Master block’s request queue.

10.9.5 Sample Application



The MODBUS TCP Master protocol is used to read data from a MODBUS slave device. MODBUS master protocol is activated on the device by connecting the TCP socket block to Modbus Master block.

The reference connection from the MODBUS master block is connected to the reader blocks, and so the MODBUS master channel is selected to direct the reading requests. With every rising edge trigger signal coming into the “Trg” input of the MODBUS writer, the value in “In” input is added to the request queue of the master block as a read request. In cases where the master block’s communication channel is available and is not in a waiting state for the previous request, the requests in the queue will run sequentially.

10.10 MODBUS FLOAT WRITER

10.10.1 Connections

Mst: Master input	
Trg: Trigger input	
In: Block input	

10.10.2 Connection Explanations

<u>Mas: Master input</u>

Master input connection

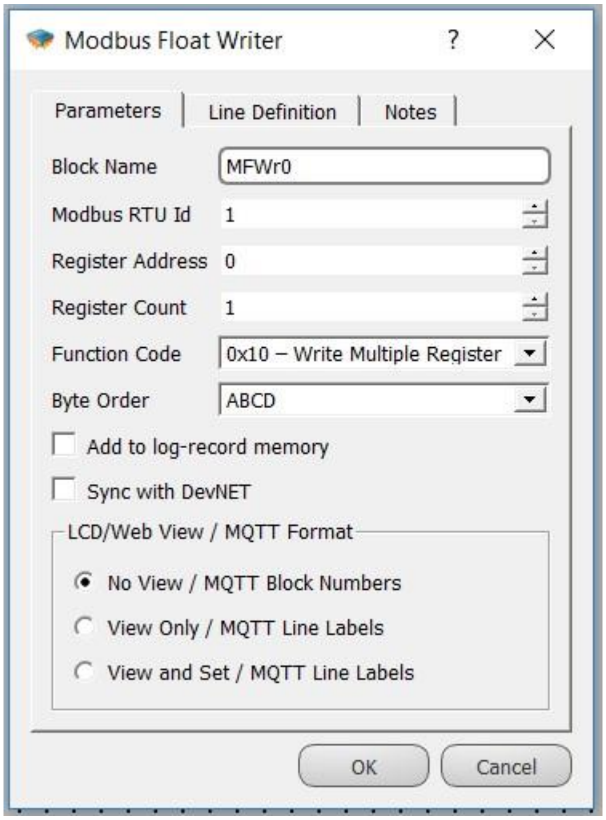
<u>Trg: Trigger input</u>

The trigger input connection

<u>In: Block input</u>

Block input connection

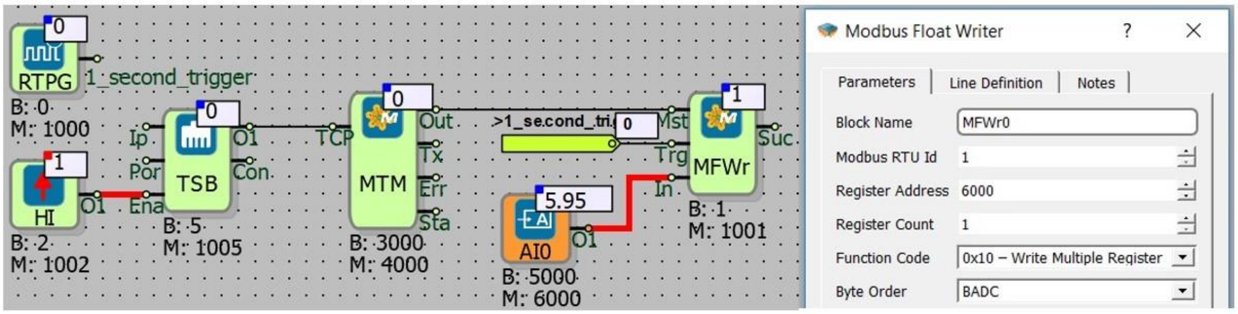
10.10.3 Block Settings

	Modbus RTU ID: The ID of the device from which the data is to be retrieved
	Register Adress: Register address to be read from slave IDs.
	Register Count: The number of registers to be read after the entered register address
	Function Code: Function code which will be selected to write the data.
	Byte Order: The byte order of the data

10.10.4 Block Explanation

It is used for writing into 2 MODBUS registers which is storing 32 bits long IEEE 754 float number. The writing request is created on the rising edge of the Trg input, and is added to the MASTER block’s request queue.

10.10.5 Sample Application

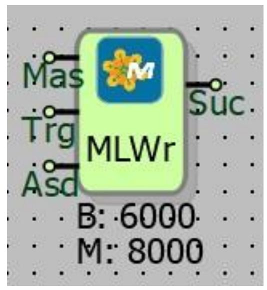


The MODBUS TCP Master protocol is used to read data from a MODBUS slave device. MODBUS master protocol is activated on the device by connecting the TCP socket block to Modbus Master block.

The reference connection from the MODBUS master block is connected to the reader blocks, and so the MODBUS master channel is selected to direct the reading requests. With every rising edge trigger signal coming into the “Trg” input of the MODBUS writer, the value in “In” input is added to the request queue of the master block as a read request. In cases where the master block’s communication channel is available and is not in a waiting state for the previous request, the requests in the queue will run sequentially.

10.11 MODBUS LONG WRITER

10.11.1 Connections

Mas: Master input	
Ttk: Trigger input	
Asd: Asdu address input	

10.11.2 Connection Explanations

Mas: Master input

Master is the entrance.

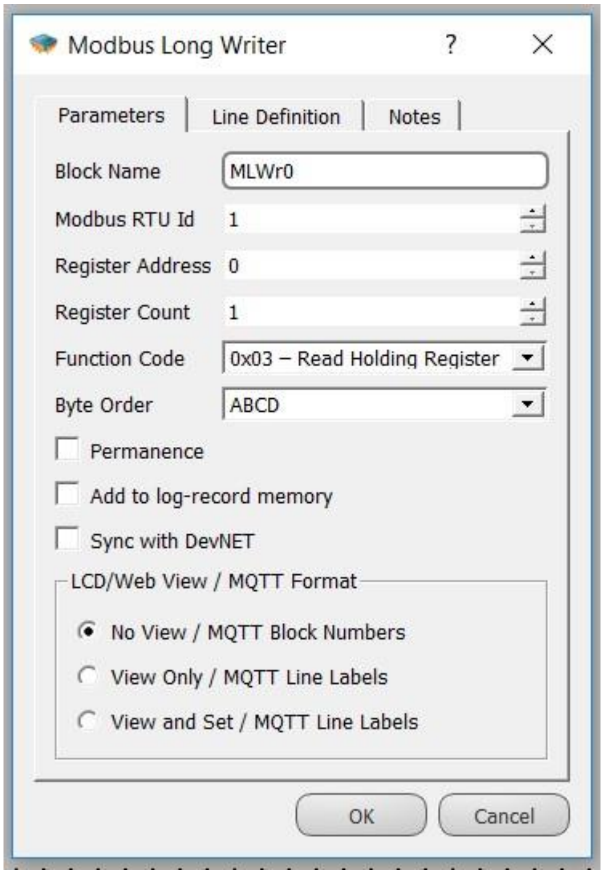
Trg: Trigger input

The trigger is the input connection.

Asd: Asdu address input

Asdu address entry for connection.

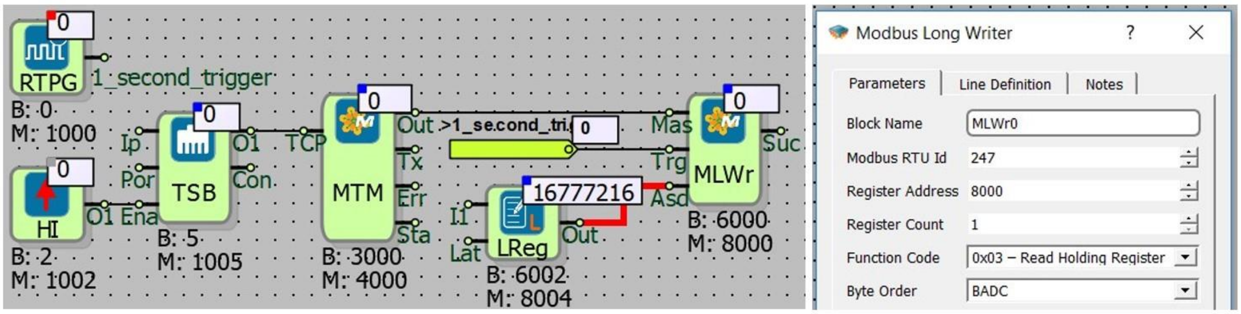
10.11.3 Block Settings

	Modbus RTU ID: The value from which the data is to be retrieved
	Register Adress: Register addresses to be read from slave IDs
	Register Count: The number of registers to be read after the entered register address
	Function Code: The function code which will be selected to read the data
	Byte Order: The byte order of the data

10.11.4 Block Explanation

It is used to write into 2 MODBUS registers that hold a 32 bits length long number. The writing request is created on the rising edge of the Trg signal, and is added to the MASTER block’s request queue.

10.11.5 Sample Application



The MODBUS TCP Master protocol is used to read data from a MODBUS slave device. MODBUS master protocol is activated on the device by connecting the TCP socket block to Modbus Master block.

The reference connection from the MODBUS master block is connected to the reader blocks, and so the MODBUS master channel is selected to direct the reading requests. With every rising edge trigger signal coming into the “Trg” input of the MODBUS writer, the value in “In” input is added to the request queue of the master block as a read request. In cases where the master block’s communication channel is available and is not in a waiting state for the previous request, the requests in the queue will run sequentially.

10.12 MODBUS READ/WRITE TABLE

10.12.1 Connections

Mas: Master input	
Tab: Table input	
Trg: Trigger input	

10.12.2 Connection Explanations

<u>Mas: Master input</u>

Master input connection

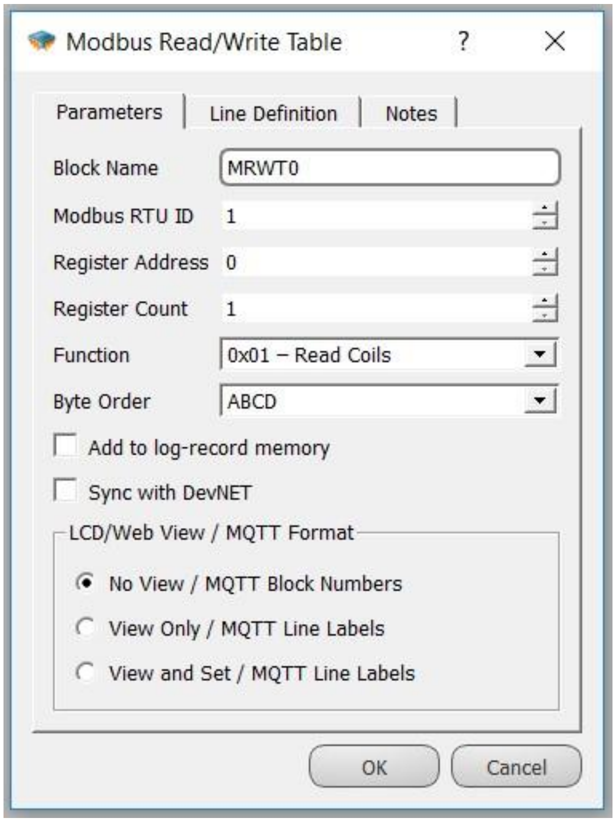
<u>Tab: Table input</u>

It is the reference input connection for the table or target/source block's start

<u>Trg: Trigger input</u>

The trigger input connection

10.12.3 Block Settings

	Modbus RTU Id: The value from which the data is to be retrieved
	Register Adress: Register addresses to be read from slave Ids
	Register Counter: The number of registers to be read after the entered register address
	Function Code: The function code which will be selected to read the data
	Byte Order: The byte order of the data

10.12.4 Block Explanation

It is used for reading/writing one or more registers starting from a specific register address.

The "register address" specifies from which register to start reading/writing.

"Number of registers" specifies the number of registers to read/write after the register specified by the register address. The maximum number of registers can be 120.

For multi-line reading, the source of the data to be read is determined by the Tab input on the block. The data source can be;

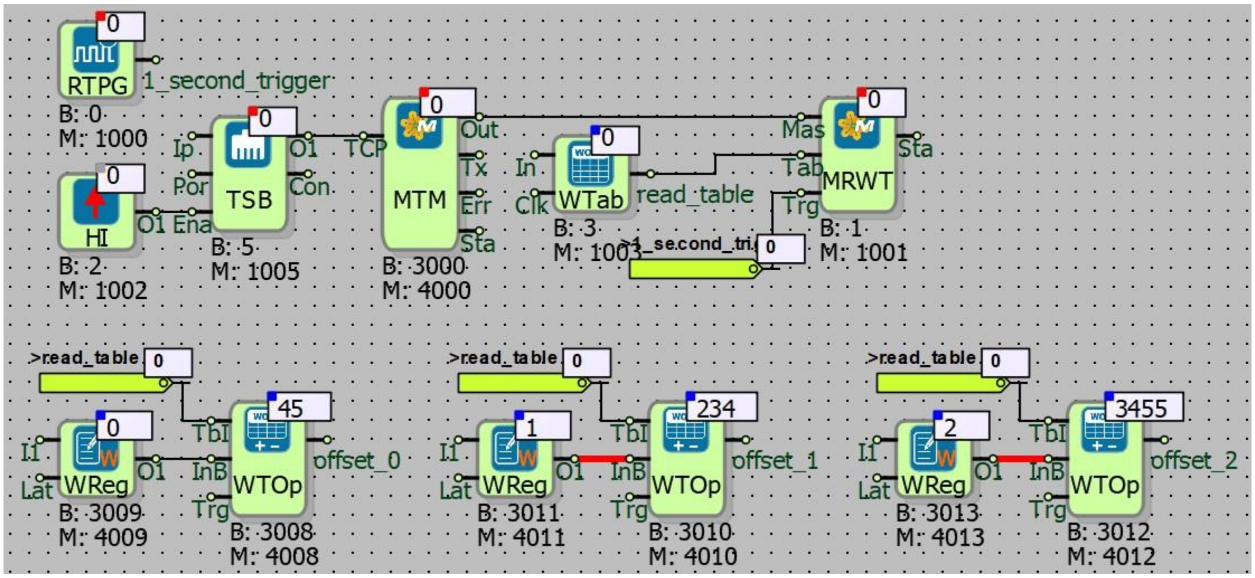
1- Table,

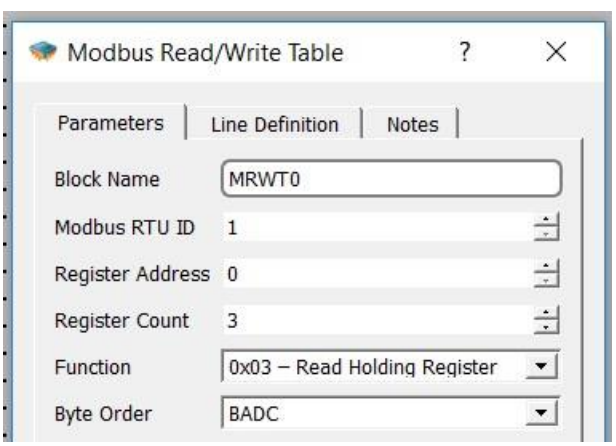
2- Normal Block Reference.

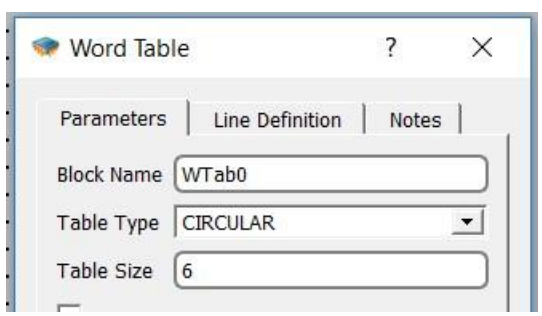
If the table is used as a data source; the memory area occupied by the table block is used as the source. The table size must be 2 times the number of registers defined by the block as BYTE, because each MODBUS writer is 2 bytes in size.

10.12.5 Sample Applications

10.12.5.1 Reading Word Table



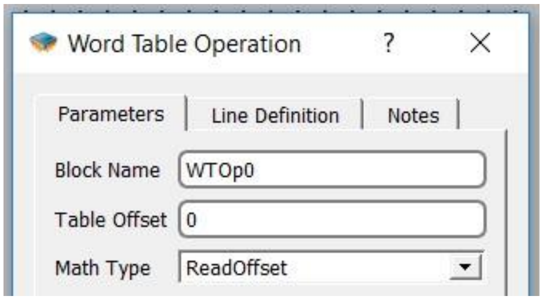




In the sample;

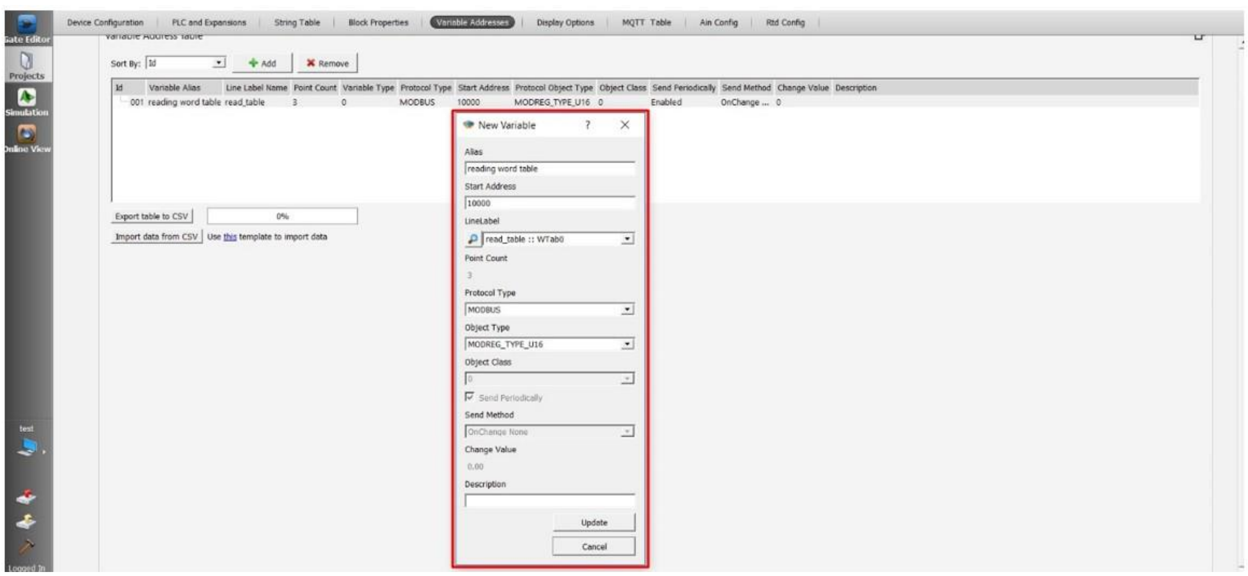
It is aimed to read 3 Word variables starting from the first Modbus address with the Modbus Reading / Writing Table (MRWT) block. In the MRWT block is defined the starting address (0. Address) and the number of registers (3) to be read.

The data read with the MRWT block is written to the Modbus Table block. To do this, open the 6 Byte area in the Modbus table block. (Each Word variable is 2 Bytes.)



The datas saved on Word Table Block, is written on Word Table Operation (WTop) block with the property of Read Offset on WTop block.

Another practical method for transferring data through a gateway via PLC:



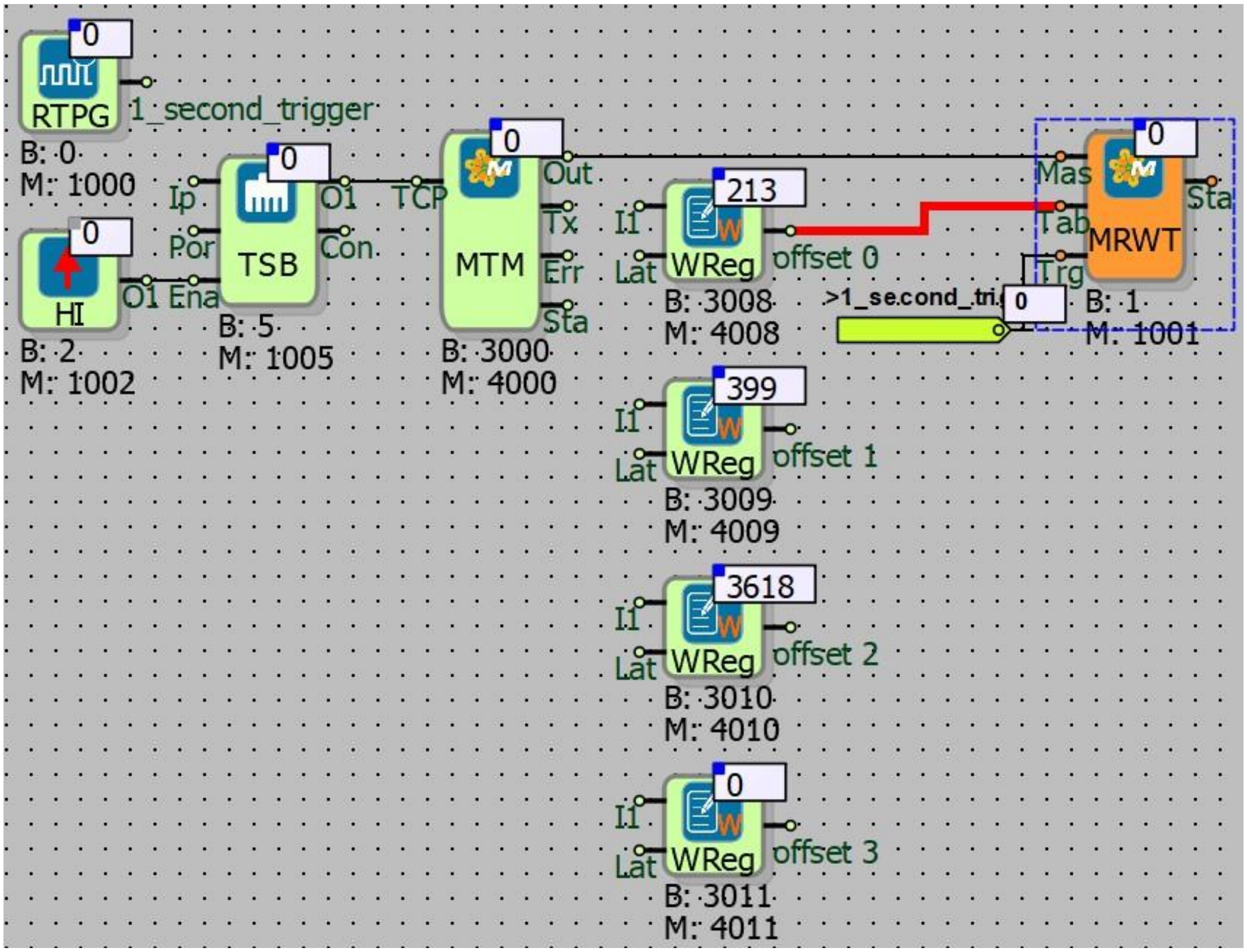
If you want to read the data carried on the Modbus Word Table block by another Modbus TCP client via this PLC, the Projects / Variable Address Table can be used.

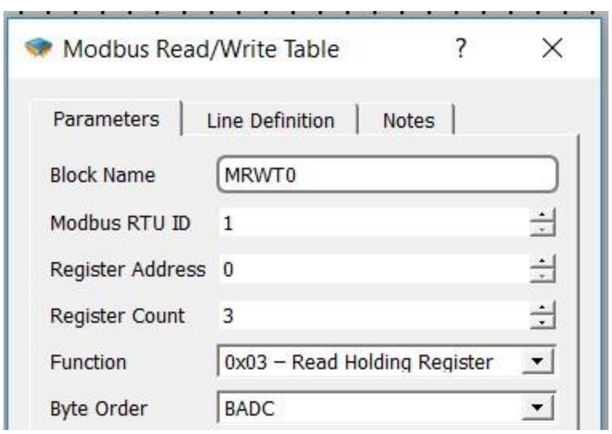
The Line Label is defined in the Word Table block. This defined Line Label is selected from the Line Label section of the Variable Address Table. The Modbus Word Addresses is automatically defined as the size of the Word Table block from the start address.

Thus, other Modbus TCP Clients can read the addresses of these PLC defined on the Variable Address Table.

Note: The Modbus addresses defined in the variable address table should be selected differently from the Modbus variable address ranges defined automatically in the Mikrodiagram. (Modbus addresses starting from 1000, 4000, 6000, 8000 should not be used.)

10.12.5.2 Reading to Consecutive Addresses





Another method of reading the variables of another Modbus RTU / TCP Slave with Modbus Read / Write Table (MRWT); read the values on sequential address registers.

In the above example, it is aimed to read 3 Word variables with MRWT block. For this reason, 3 registers have been opened in the MRWT block.

The address from which the reading is to be made is selected by the Register Address in MRWT block.

With Register Count of MRWT block is defined that how many addresses from Modbus RTU / TCP Slave's address is selected in the Register Address are selected. (In the above example, it is selected to read 3 addresses as from the 0th address.)

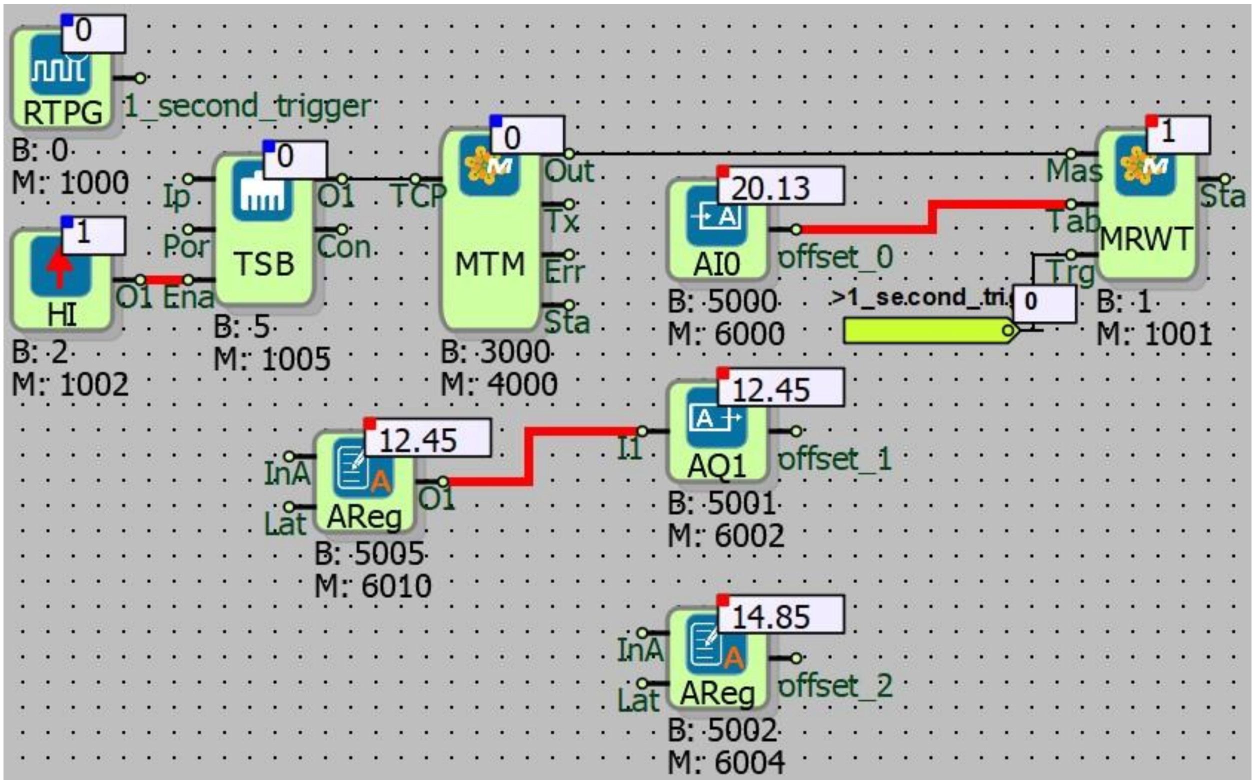
From the Word Register connected to the Tab input of the MRWT block, the data in the Modbus RTU / TCP Slave will be read on 3 Word Registers (4008, 4009., and 4010. Modbus addresses) with sequential address.

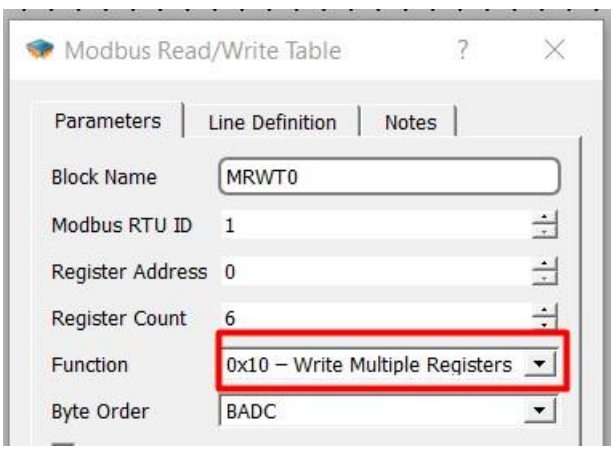
Reading operation MRWT block is repeated at each rising edge triggering to the Trg input.

Modbus RTU/TCP Slave Addresses	**Modbus RTU/TCP Master Addresses**
0. Address	4008. Address
1. Address	4009. Address
2. Address	4010. Address

Note: In this example, for MRWT block only 3 addresses have been opened, that's why 4011 Modbus Addressed block has not read any address.

10.12.5.3 Writing to Consecutive Addresses





With Modbus Read / Write Table (MRWT) can be written to another Modbus RTU / TCP Slave's consecutive sequential writeable (W or R/W) variables.

In the above example, the value of 3 analog variables with MRWT block is written to Modbus RTU / TCP Slave. For this reason, 3 field have been opened in the MRWT block for Analog variables. (Each Analog variable is equal to 2 Word variables.)

The registrar address on the MRWT blog is the starting address for writing on the Modbus RTU/TCP Slave device.

Register Count in the MRWT block is defined for how many addresses would be written by MRWT block to Slave. (In the above example, it is selected to write 3 Analog addresses from the 0. address. 0., 2. and 4. Addresses)

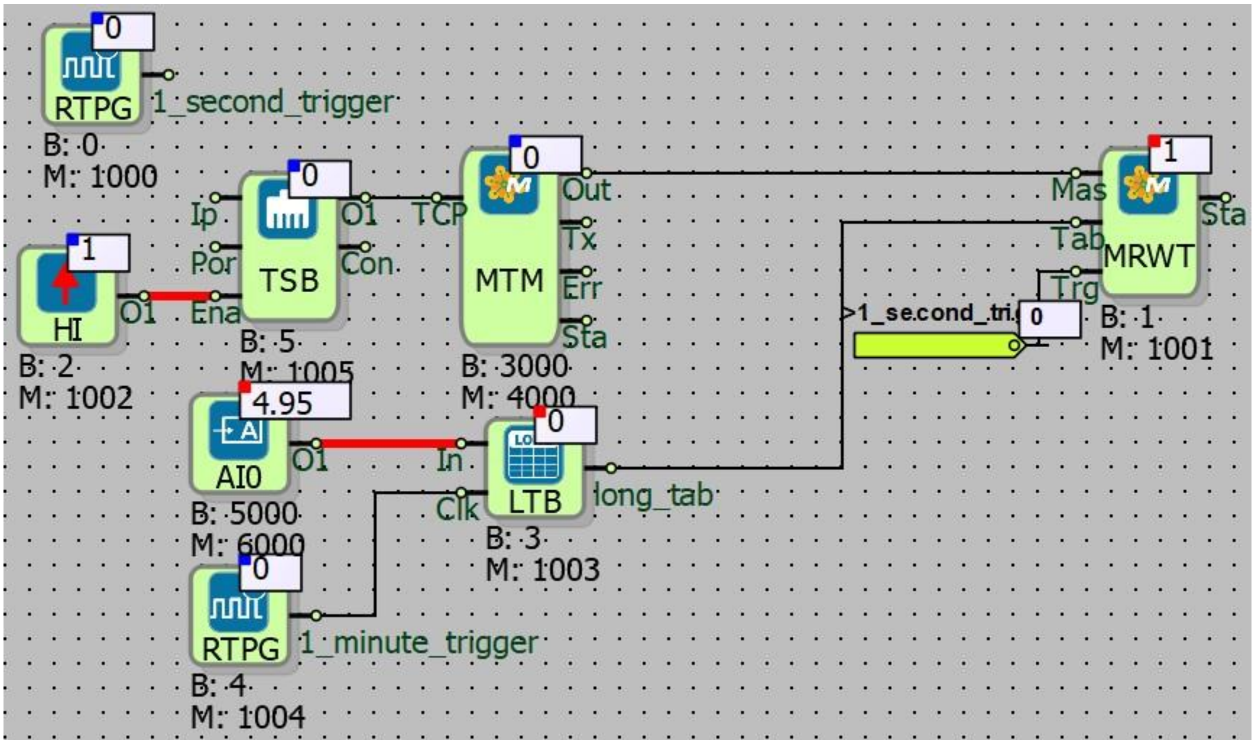
From the analog input block connected to the tab input of the MRWT block, data on 3 analog blocks with sequential sequential address will be written to Modbus RTU / TCP Slave. (6000. Modbus Addressable AI0 block, 6002. Modbus Addressed AQ0 block, 6004. Modbus Addressable Analog Register block.)

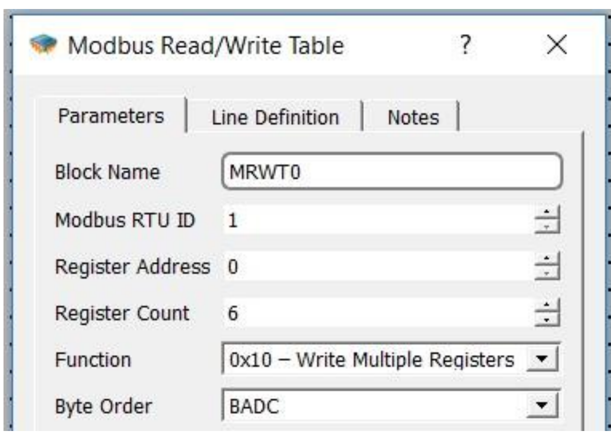
In the MRWT block, the function type should be selected according to the type of writing function and the variable typ3. (Write Multiple Registers, Write Multiple Coils.)

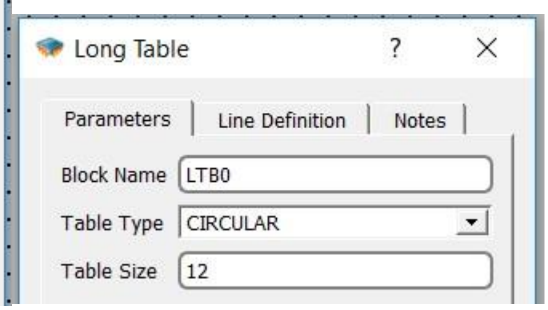
Write operation The MRWT block is repeated at each rising edge trigger that is input to input Trg.

Modbus RTU/TCP Slave Addresses	**Modbus RTU/TCP Master Addresses**
Analog Address 0	Analog Address 6000
Analog Address 2	Analog Address 6002
Analog Adrdess 4	Analog Address 6004

10.12.5.4 Writing to Successive Addresses from a Table







With Modbus Read / Write Table (MRWT) can be written to another Modbus RTU / TCP Slave's consecutive sequential writeable (W or R/W) variables.

In the above example, the area for 3 long variables is opened in the Long Table block that connects to the Tab input of the MRWT block. (Each Long variable is equal to 2 Word variables.)

The register address on the MRWT blog is the starting address for writing on the Modbus RTU/TCP Slave device.

Register Count in the MRWT block is defined for how many addresses would be written by MRWT block to Slave. (In the above example, it can be written from 0. to 5. Addresses. 0, 1, 2, 3, 4, 5. Addresses)

In every minute a sample is taken from the Analog Input (AI0) block connected to the In input of the Long Table block connected to the “Tab” input of the MRWT block. This samples are written in 3 long fields in the long table. The values in the table are written to a Modbus RTU / TCP Slave per second.

In the MRWT block, the function type should be selected according to the type of writing function and the variable typ3. (Write Multiple Registers, Write Multiple Coils.)

Write operation The MRWT block is repeated at each rising edge trigger that is input to input Trg.

10.13MODBUS STATUS BLOK

10.13.1 Connections

Efe: Efendi girişi Rtu: Slave ID Girişi		Sta: Connection Status

10.13.2 Connection Explanations

Mas: Master input

Master input connection.

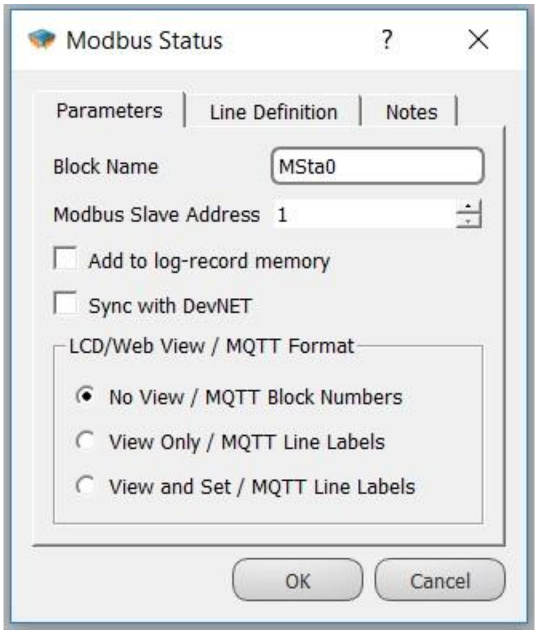
Rtu: Slave ID input

The ID of the device to which the connection status information will be read is entered.

Sta: Connection Status

This output is for connection status information.

10.13.3 Block Settings

<table>
<tr>
<td>
</td>
<td>Modbus Slave Address: The ID of the device in which status information is to be received can also be selected from the block.</td>
</tr>
</table>

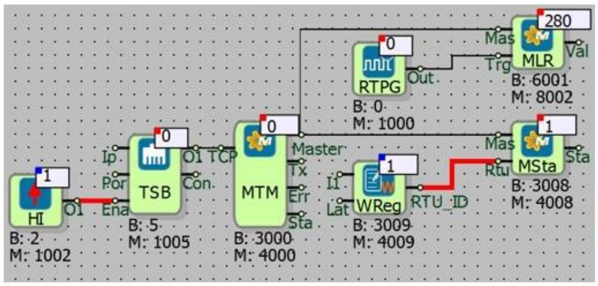
10.13.4 Block Explanation

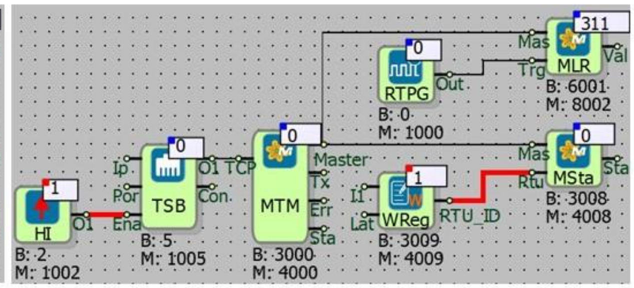
The status information of slave devices that read and write via Modbus Master blocks is read by this block. The Modbus Status block reads the status information via the Modbus Master block to which it is connected. The ID of the device to read status information can be defined from Block Object Properties or Block second input (RTU ID input).

If the block output is 1, communication with the Modbus device at the entered slave address is exist and is successful. If the block output is 0, there is no communication or response packets with the Modbus device at the entered slave address.

The block output is updated when the corresponding slave sends a request to the device. If the expected response from the slave device cannot be received during the defined timeout, the status information is updated to 0 at the end of this timeout time.

10.13.5 Sample Application





(1) (2)

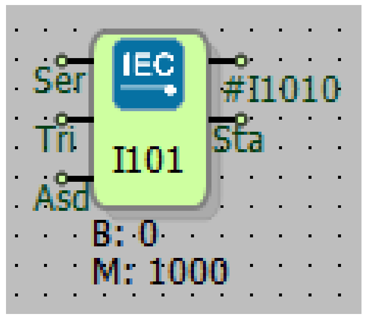
In the sample; communication connection is inquired by Modbus Status block to Modbus TCP/RTU Slaves'.

When there is a communication connection, the block output is set to 1. The block output is 0 when there is no communication connection.

11 IEC DNP3 PROTOCOL BLOCKS

11.1 IEC101 SLAVE

11.1.1 Connections

<table>
<tr><td>Ser: Communication input</td><td rowspan="3">
</td><td>#I1010: Block number output</td></tr>
<tr><td>Tri: Block trigger input</td><td rowspan="2">Sta: Status Information output</td></tr>
<tr><td>Asd: Asdu address input</td></tr>
</table>

11.1.2 Connection Explanations

Ser: Communication input

Communication input.

Tri: Block trigger input

Block trigger input.

Asd: Asdu address input

The input that defines the asdu address.

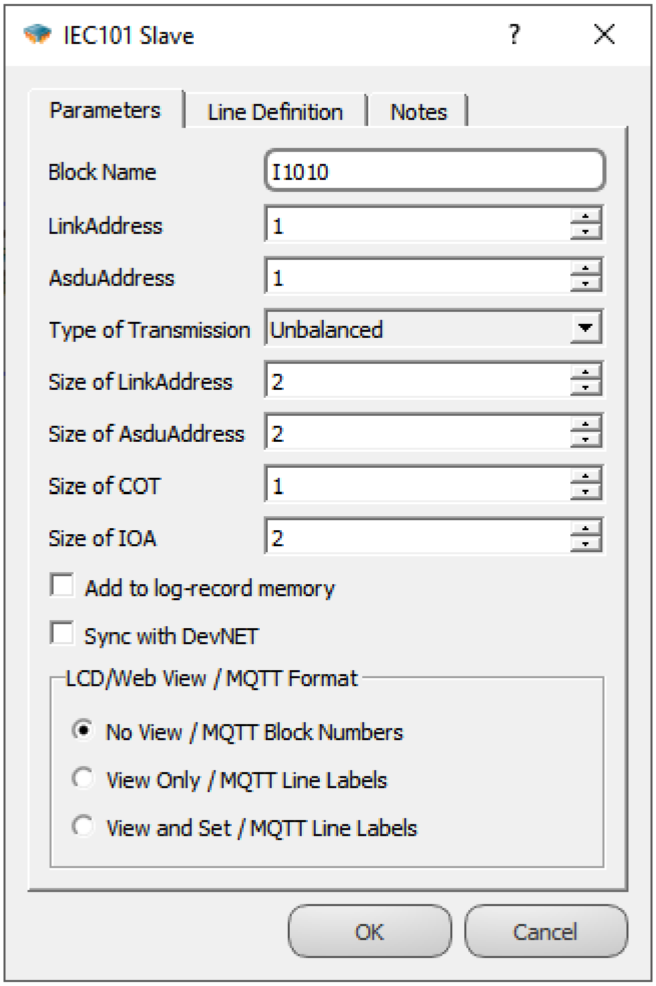
#I1010: Block number output

The output of the block number

Sta: Status information output

The status information output

11.1.3 Block Settings



Connection address: Link state address.
Asdu Address: ASDU state address.
Transmission type: Balanced or Unbalanced protocol is selected.
Link address size: Select the number of the bytes of link address.
Asdu Address size: How many bytes of the asdu address will be selected.
COT size: The number of bytes of the Cause of Transmission field is selected.
IOA Byte: The number of bytes of Information Object Addresses is selected.

11.1.4 Block Explanation

By adding IEC101 block, The IEC 60870-5-101 slave is activated on the RTU.

TCP or Serialport block is added to IEC101block “ser” input.

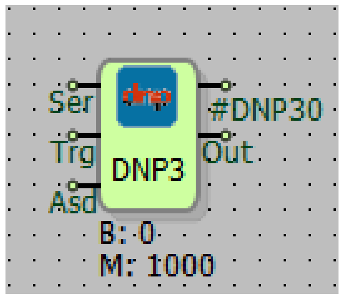
To serve more than one server, IEC101 block must be added for each server.

.If The IEC101 Asdu address is set from outside not from the inside the block, The Asd enter is used

On the rising edge of the trigger, the periodically sending objects between IEC101 objects are actively transmitted to the server by the periodic COT. Trigger input can be left blank.

11.2 DNP3 Slave

11.2.1 Connection

Ser: TCP Socket Input		#DNP30: Not used
Trg: Trigger Input		Out: Not used
Asd: Asdu Address Input		

11.2.2 Connection Explanations

<u>Ser: TCP Socket Input</u>

The TCP server socket block from which the DNP3 protocol will run is connected to this input.

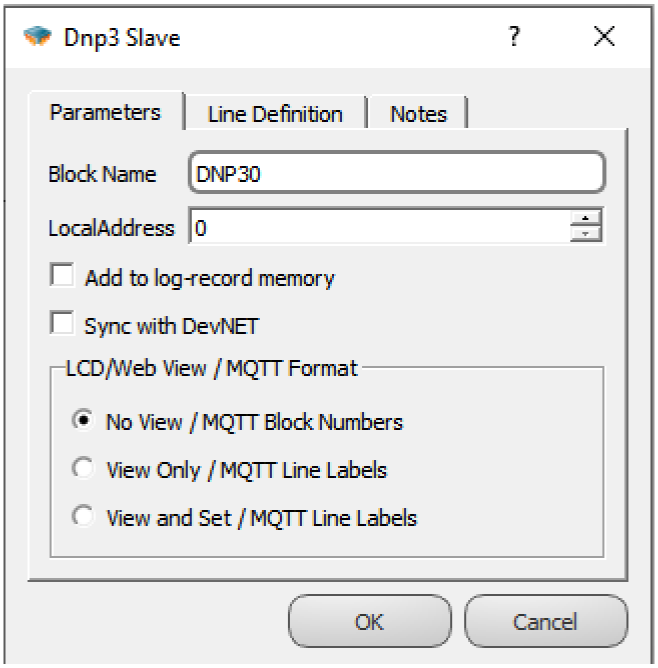
<u>Trg: Trigger Input</u>

Trigger input for periodic sending. It works as a rising edge.

<u>Asd: Asdu Address Input</u>

The ASDU address is used as input.

11.2.3 Block Settings

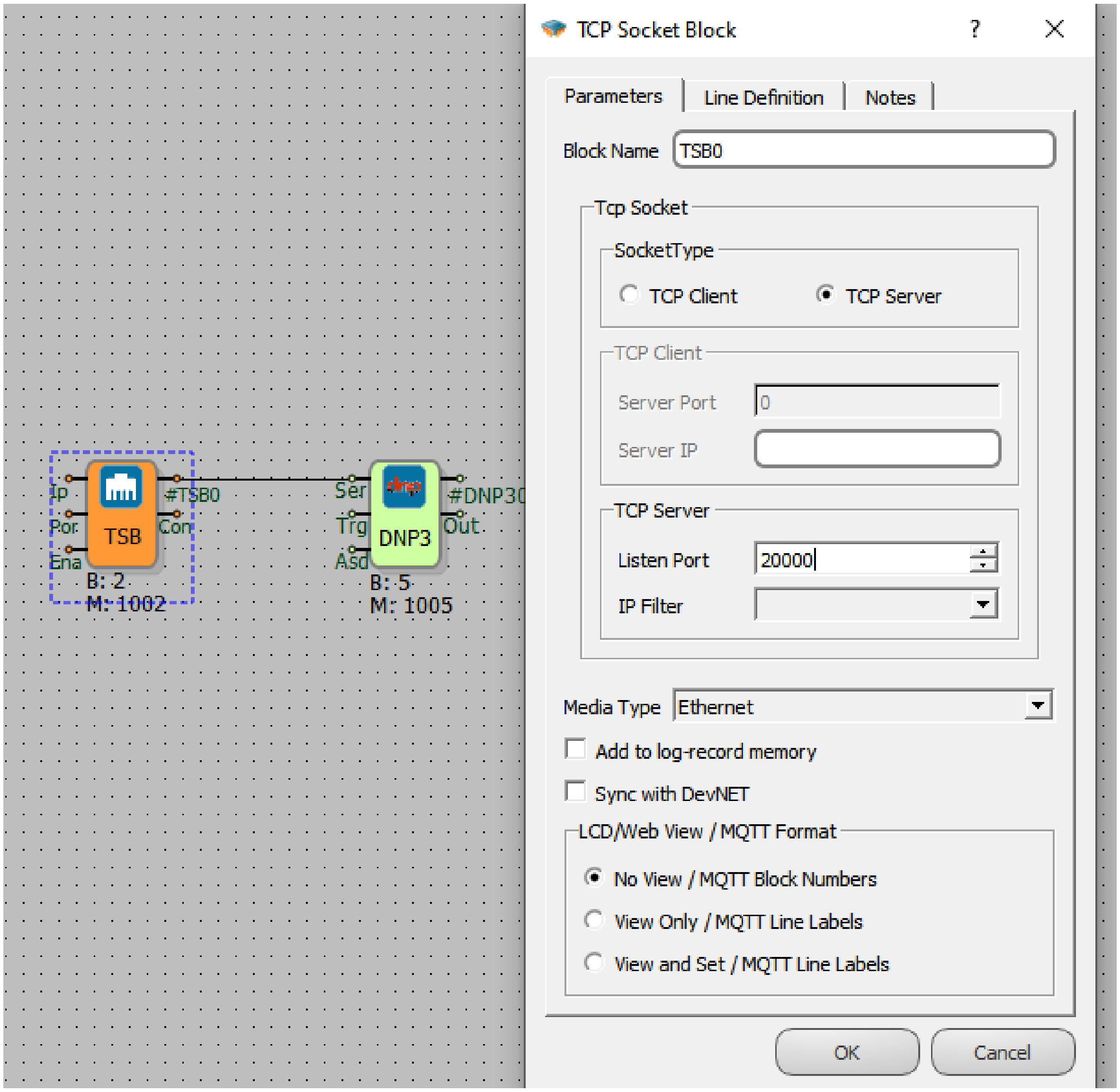
	Local Address: Define DNP3 slave address

11.2.4 Block Explanation

By adding the DNP3 slave block, DNP3 is activated on the RTU.
TCP or Seriport block is added to DNP3 block ser input.
To serve more than one server, a DNP3 block must be added for each server.
Asp input is used if the DNP3 Asdu address is to be set from outside, not from within the block.

On the rising edge of the trigger, DNP3 objects are periodically sent active selected objects,
It is transmitted to the server via periodical COT. Trigger input can be left blank.

11.2.5 Sample Application



With the addition of DNP3 Slave Block to the RTU logic project, the DNP3 protocol becomes active in the RTU.

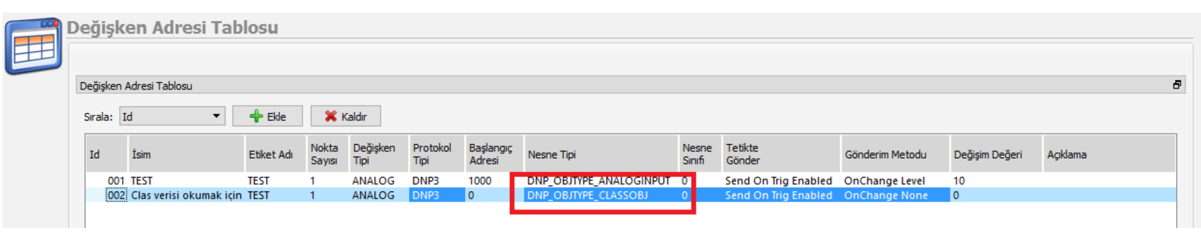
Variables in the RTU logic project, DNP3 association is provided in the variable address table.

11.2.6 DNP3 Object Types

11.2.6.1 DNP3 Object Types in Reading Direction

DNP3 Object Type	Possible Attribution Object Type
Single Bit Binary Input Data Object 01 - Variation 01	Binary, Word, Analog, Long
Binary Input With Status Data Object 01 - Variation 02	Binary, Word, Analog, Long
Binary Input Change Without Time Data Object 02 - Variation 01	Binary, Word, Analog, Long
Binary Output Data object 10 - Variation 01	Binary, Word, Analog, Long
Binary Output Status Data object 10 - Variation 02	Binary, Word, Analog, Long
32 BIT Analog Input Data Object 30 - Variation 01	Long
16 BIT Analog Input Data Object 30 - Variation 02	Binary, Word
32 BIT Analog Input Without Flag Data Object 30 - Variation 03	Long
16 BIT Analog Input Without Flag Data Object 30 - Variation 04	Binary, Word
Short Float Analog Input Without Flag Data Object 30 - Variation 05	Analog

NOTE 1: A variable of type DNP_OBJTYPE_CLASSOBJ must be added from the variable Adress table to draw Class 0, Class 1, and Class 2 data. Other settings of this variable, such as address, line tag, can be selected at random.



11.2.6.2 DNP3 Object Types in Control Direction

The write variable is also automatically generated for each block mapped to the read type. The types of variables that can be accessed as write to defined read objects are as follows:

DNP3 Nesne Tipi	Possible Attribution Block Types
Control Relay Output Block Data Object 12 - Variation 01	Binary, Word, Analog, Long
32 Bit Analog Output Block Data Object 41 - Variation 01	Long
16 Bit Analog Output Block Data Object 41 - Variation 02	Binary, Word
Short Float Analog Output Block Data Object 41 - Variation 03	Analog

11.2.7 DNP3 Event Mechanism

11.2.7.1 Event Definition for DNP3 Objects

In the variable address table, the send in change selection is available for DNP3 objects. When the value of the variable defined in this menu changes, the operation to be performed is selected.

None: Spinner does not trigger submission

- Level: when the amount defined in “Change Value” is changed, the sending is triggered.
- Percentage: Submission is triggered when there is a percentage change defined in " Change Value”.

The change face or level is set with the value” Change Value". Sets the percent or level change value with “Send method”.

11.2.7.2 DNP3 Instantaneous Transmission of Event Situations

The RTU device labels the states defined as send and change detected as events and assigns a time tag to the event. In case of a tagged event, if there is a connection to the server and the server is active in the device sending “UNSOLICED”, the relevant object is immediately forwarded as “UNSOLICED”.

If the connection exists with the server and the events detected with the “UNSOLICITED” sending active are sent with the DNP3 object types specified in the following table.

DNP3 Object Type	Possible Attribution Block Types
Binary Input Change Without Time Data Object 02 - Variation 01	Binary
32 Bit Analog Input Change Without Time Data Object 32 - Variation 01	Long
16 Bit Analog Input Change Without Time Data Object 32 - Variation 02	Word
Short Float Analog Input Change Without Time Data Object 32 - Variation 05	Analog

11.2.7.3 DNP3 Time-Tagged Submission Of Event States

Event Control controls continue passively sending “UNSOLICED” even if there is no connection to the server or even if there is no connection. In the event of an event under these circumstances, event information is recorded in the event memory with the time tag and this data is kept in the device as CLASS 1 data..

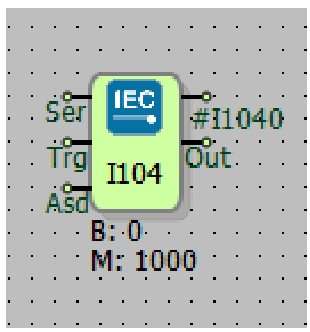
This event data stored in memory can be read by the server with Class 1 data read management. This CLASS 1 data is also automatically forwarded to the server by RTU if” UNSOLICED " sending is enabled.

Class 1 event data is dispatched with the DNP3 object types specified in the following table.

DNP3 Object Type	Possible Attribution Block Types
Binary Input Change With Time Data Object 02 - Variation 02	Binary
32 Bit Analog Input Change With Time Data Object 32 - Variation 03	Long
16 Bit Analog Input Change With Time Data Object 32 - Variation 04	Word
Short Float Analog Input Change With Time Data Object 32 - Variation 07	Analog

11.3 IEC104 Slave

11.3.1 Connections

Ser: TCP Socket Input		#I1040: Link information output
Trg: Trigger Input		Out:
Asd: Asdu Address Input		

11.3.2 Connection Explanations

<u>Ser: TCP Socket Input</u>

The TCP server socket block from which the IEC104 protocol will work is connected from this input

<u>Trg: Trigger Input</u>

Trigger input for periodic sending. It works as a rising edge.

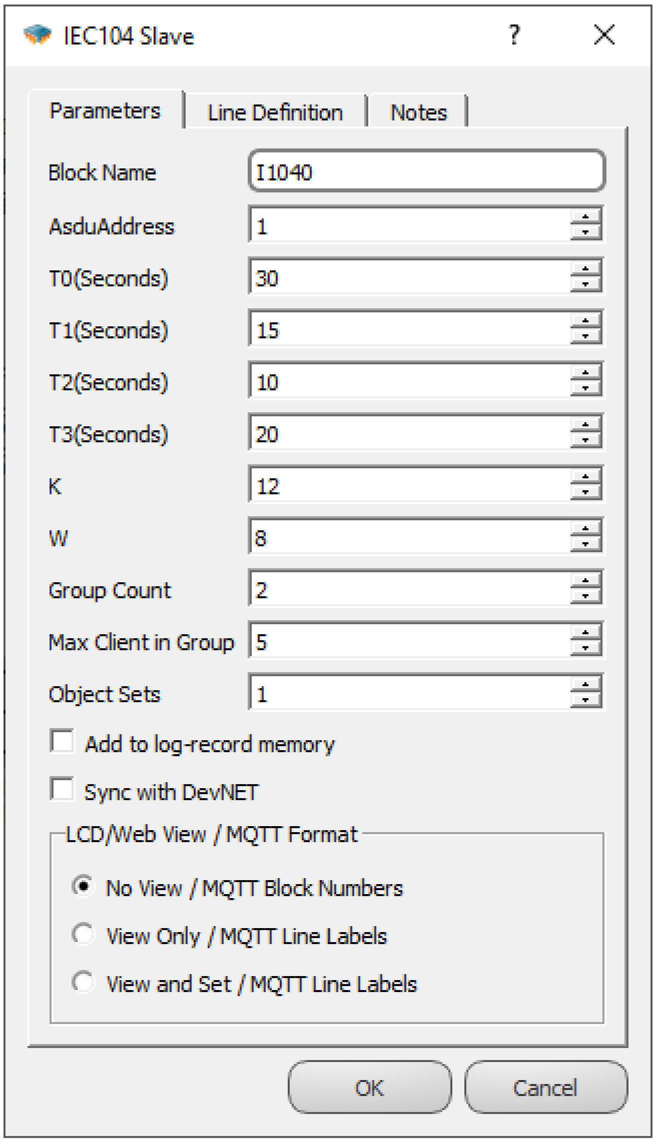
<u>Asd: Asdu Address Input</u>

The ASDU address is used as input.

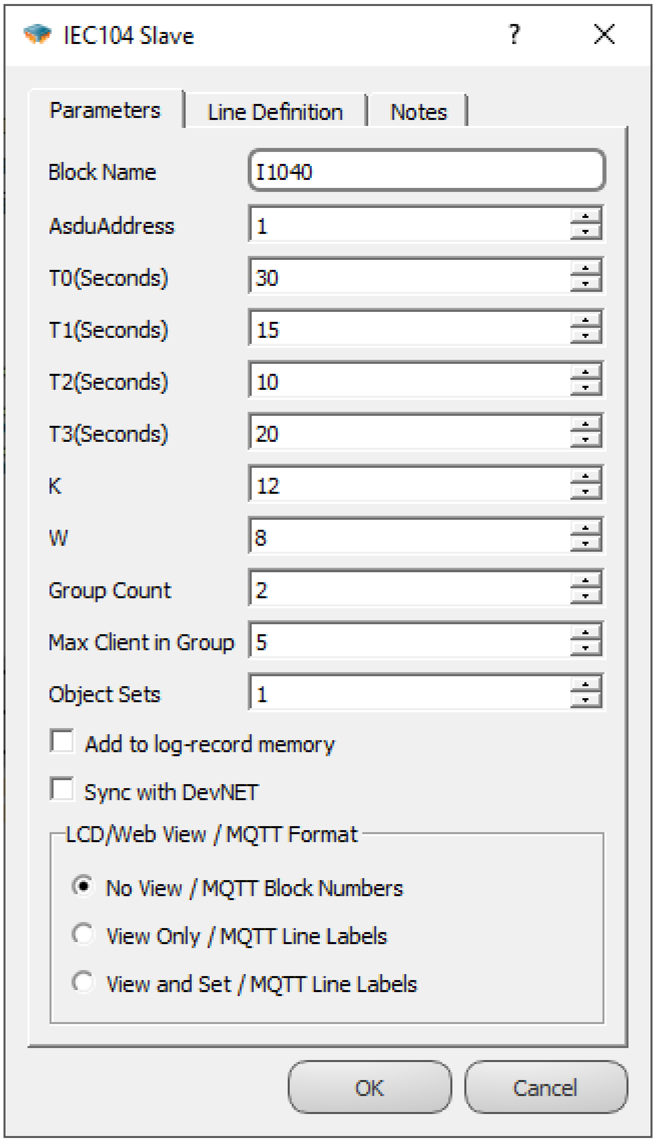
<u>#I1040: Connection Status Output</u>

If there is an IEC104 Master connected to the IEC104 Slave block, this output will be 1.

11.3.3 Block Settings



AsduAddress: IEC104 slave station ASDU address is defined.
T0: TCP IEC104 slave station ASDU address is defined.
T1: Test APDU timeout period.
T2: Timeout period for Ack.
T3: Test frame sending time
K: The maximum allowable difference between the sequence number in the received packet and the number in the send status variable
W: ACK is sent after receiving W up to I Format APDU
Group Count**: The number of Masters that the device can establish connections with as an IEC 104 Slave is specified here. This value can be a maximum of 2 for RTU devices and a maximum of 4 for DM devices.

	Max Client in Group**: The maximum number of Slave connections that can be established to an IEC 104 Master is specified here. (Currently set to 5.)
	Object Sets*: It is used to define multiple IEC 104 Slaves. Thanks to the value entered here, IEC 104 objects can be assigned to different Slave addresses. It is used in conjunction with the 'Object Set No' in the Variable Address Table. For more detailed information, please refer to the Block Descriptions.
	Add to log-record memory: If block values are desired to be added to the event log memory when there is no connection with the server, the "Add to Log Memory" option should be selected.
	Sync with DevNET: If it is desired to send the values of all blocks to the server when the connection is established, this option should be selected.

**This is valid for Telediagram version 18 and later.*
***In Telediagram versions earlier than 18, these features are provided by sending special commands through the Mikroterminal application.*

11.3.4 Block Explanation

To enable the IEC104 protocol over RTU, you need to add an IEC104 Slave block to the Telediagram project and connect the TCP Socket block to the "Ser" input of the IEC104 Slave block. In the TCP Socket block settings, the TCP Socket Type should be selected as "Server" and the listening port should be defined. To activate the TCP Socket block, the "Ena" input of the TCP Socket block should be connected to the High Gate block.

If you want to serve multiple servers, you need to add an IEC104 Slave block for each server in the Telediagram project.

The IEC104 ASDU address can be configured either from the block settings of the IEC104 Slave block or from the "Asd" input of the IEC104 Slave block.

The values of the IEC104 objects that are selected for periodic transmission will be sent to the server when a rising edge signal is received at the "Trg" input of the IEC104 Slave block. If there is no data transmission through periodic or trigger-based methods, the trigger input can be left unconnected.

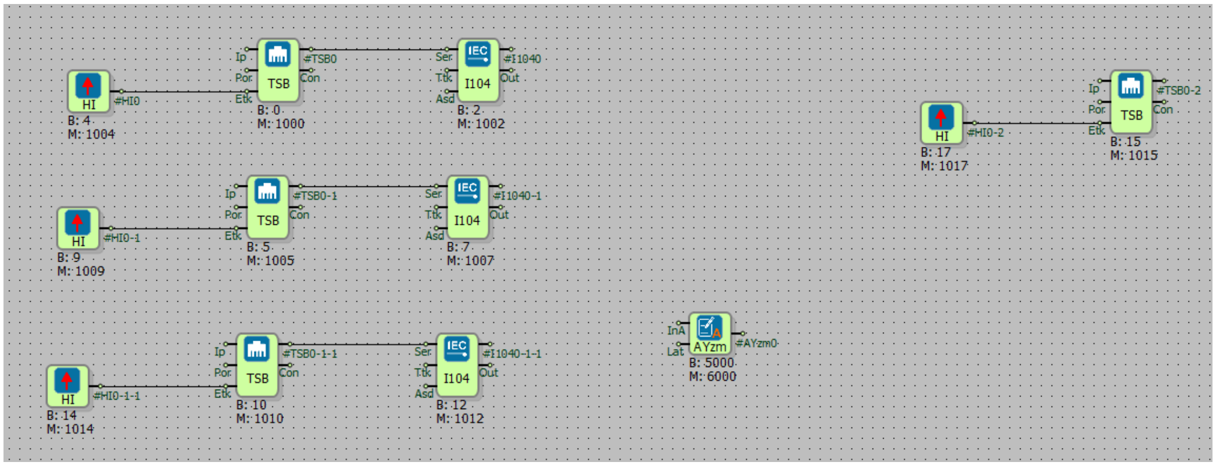
If you want to open multiple IEC104 Slaves on the device, you should make the configuration from the "object sets" section in the block settings of the IEC104 Slave block. This section is used in conjunction with the variable address table. When defining IEC104 objects in the variable address table, the "object set no" entered should correspond to the object sets value.

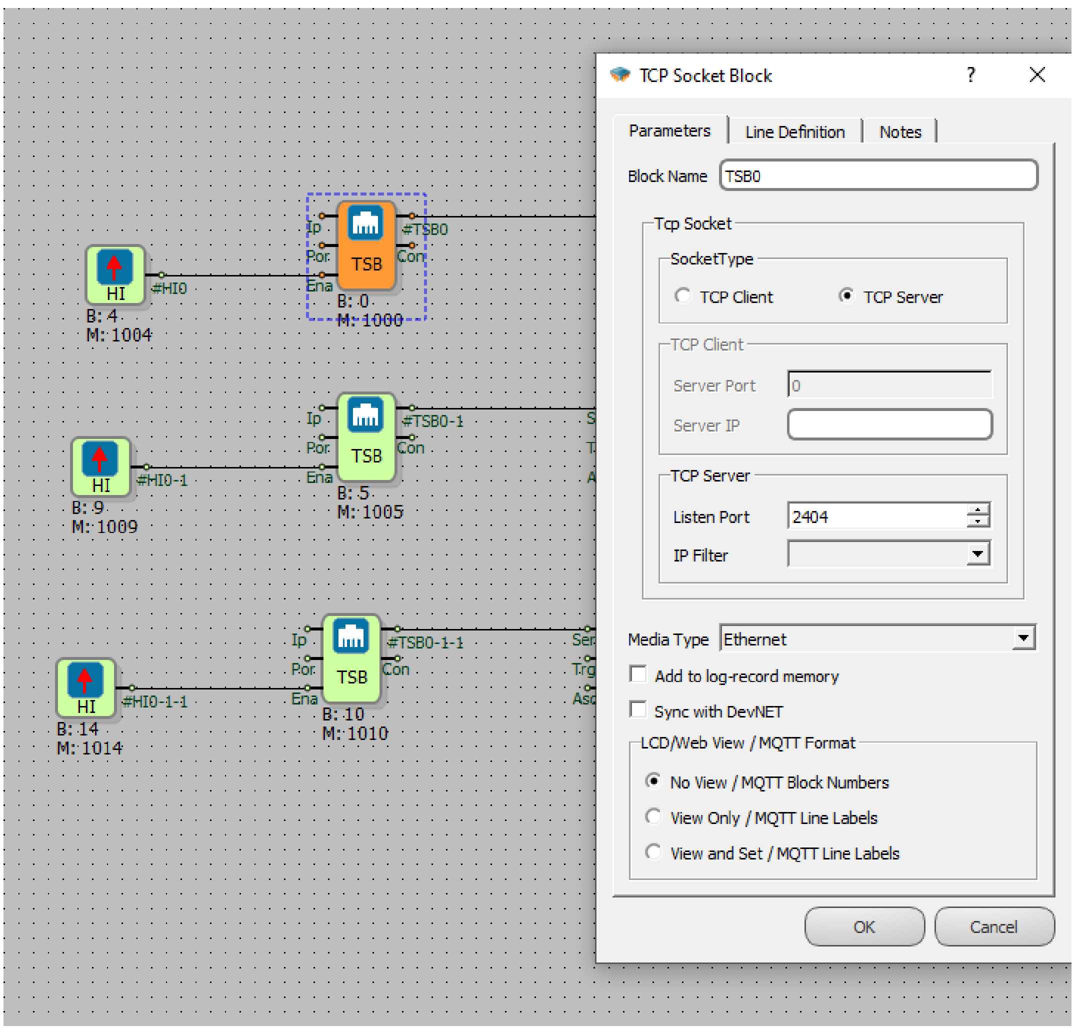
For example, if the "object sets" value in the block settings of the IEC104 Slave block is set to 1, the "object set no" in the variable address table should be 0. ($2^0=1$)

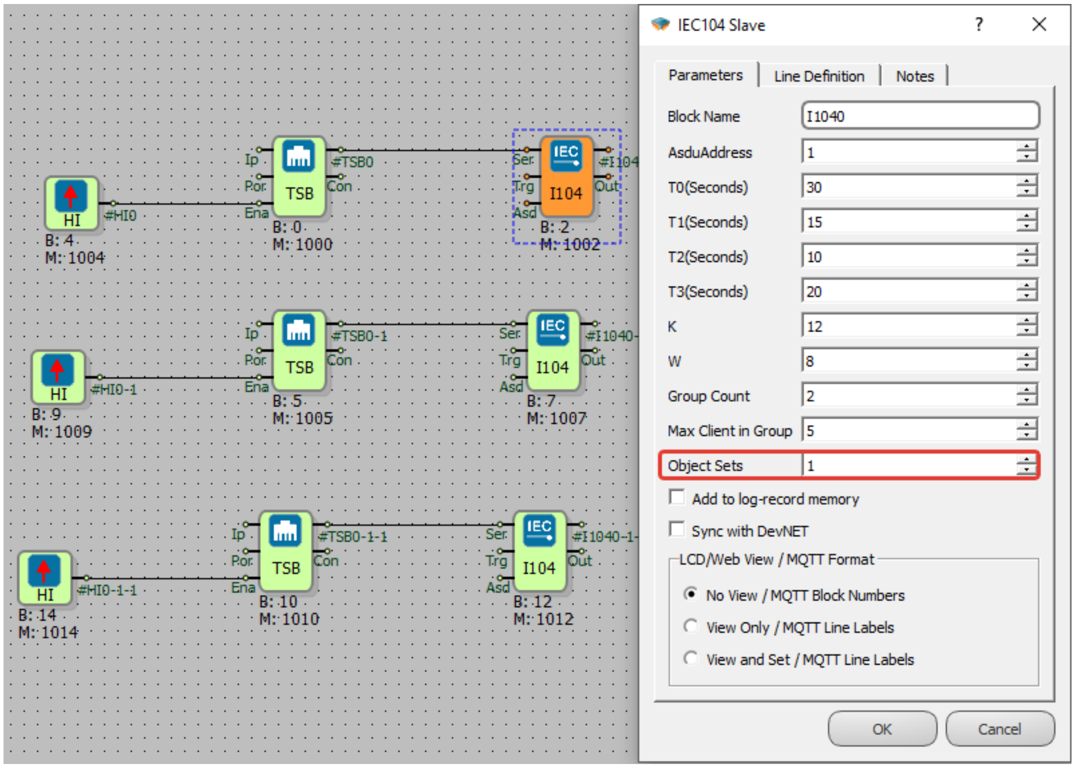
If the "object sets" value is set to 2, the "object set no" in the variable address table should be 1. ($2^1=1$)

And if the "object sets" value is set to 8, the "object set no" in the variable address table should be 3. ($2^3=8$)

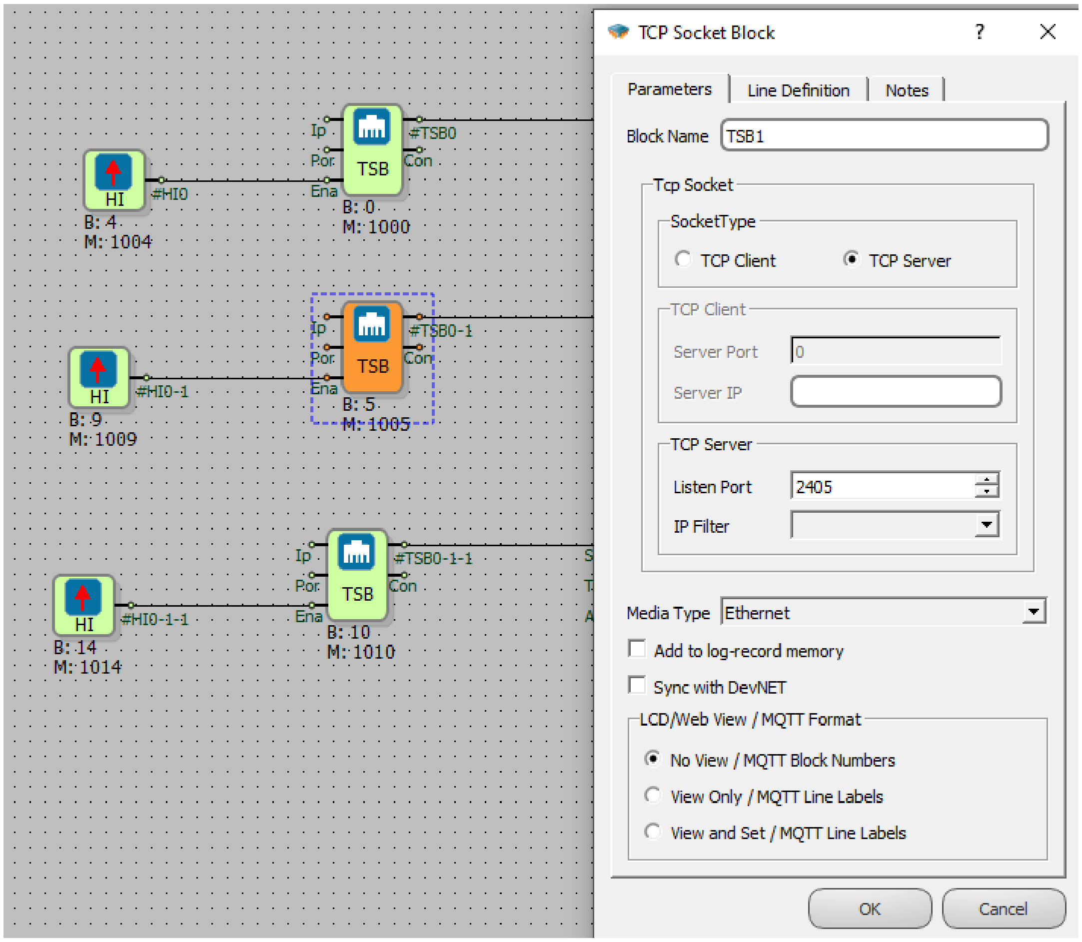
11.3.5 Sample Application



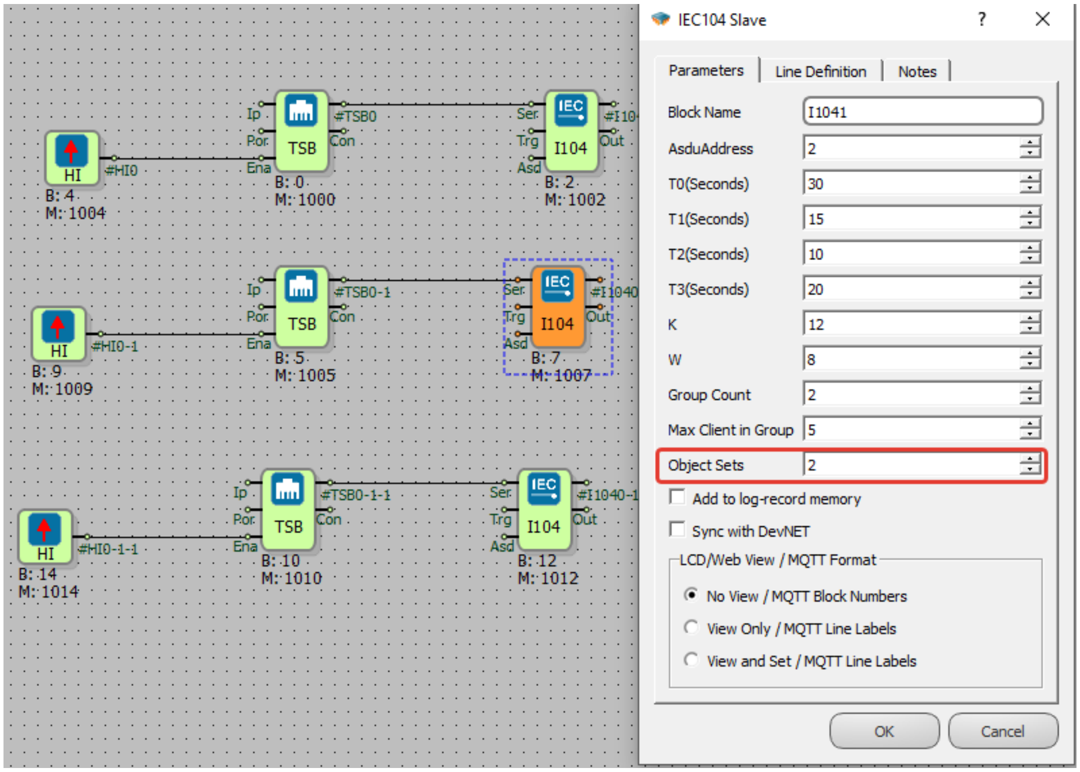


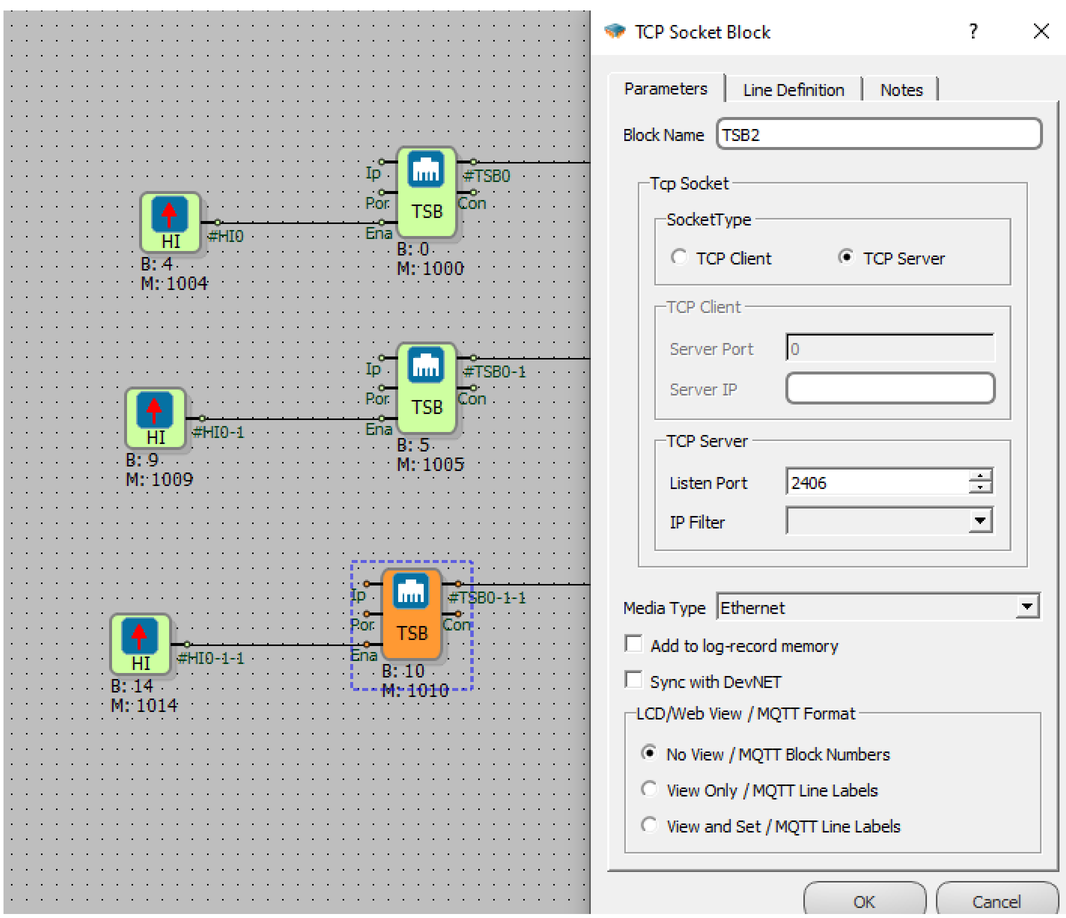


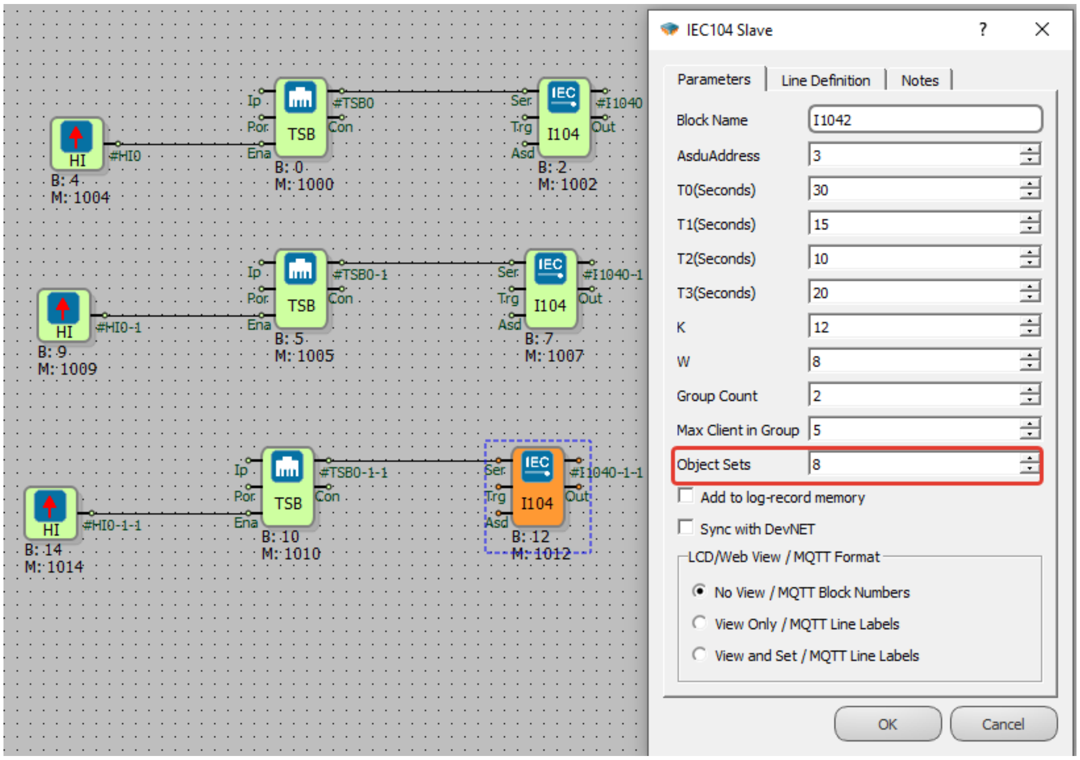
IEC104 Slave
Parameters
Line Definition
Notes
Block Name
I1040
AsduAddress
1
T0(Seconds)
30
T1(Seconds)
15
T2(Seconds)
10
T3(Seconds)
20
K
12
W
8
Group Count
2
Max Client in Group
5
Object Sets
1
Add to log-record memory
Sync with DevNET
LCD/Web View / MQTT Format
No View / MQTT Block Numbers
View Only / MQTT Line Labels
View and Set / MQTT Line Labels
OK
Cancel
HI
#HI0
B: 4
M: 1004
TSB
#TSB0
B: 0
M: 1000
I104
B: 2
M: 1002
HI
#HI0-1
B: 9
M: 1009
TSB
#TSB0-1
B: 5
M: 1005
I104
#I1040-
B: 7
M: 1007
HI
#HI0-1-1
B: 14
M: 1014
TSB
#TSB0-1-1
B: 10
M: 1010
I104
#I1040-1-
B: 12
M: 1012



TCP Socket Block
Parameters
Line Definition
Notes
Block Name
TSB1
Tcp Socket
SocketType
TCP Client
TCP Server
TCP Client
Server Port
0
Server IP
TCP Server
Listen Port
2405
IP Filter
Media Type
Ethernet
Add to log-record memory
Sync with DevNET
LCD/Web View / MQTT Format
No View / MQTT Block Numbers
View Only / MQTT Line Labels
View and Set / MQTT Line Labels
OK
Cancel
HI
#HI0
B: 4
M: 1004
TSB
#TSB0
B: 0
M: 1000
HI
#HI0-1
B: 9
M: 1009
TSB
#TSB0-1
B: 5
M: 1005
HI
#HI0-1-1
B: 14
M: 1014
TSB
#TSB0-1-1
B: 10
M: 1010

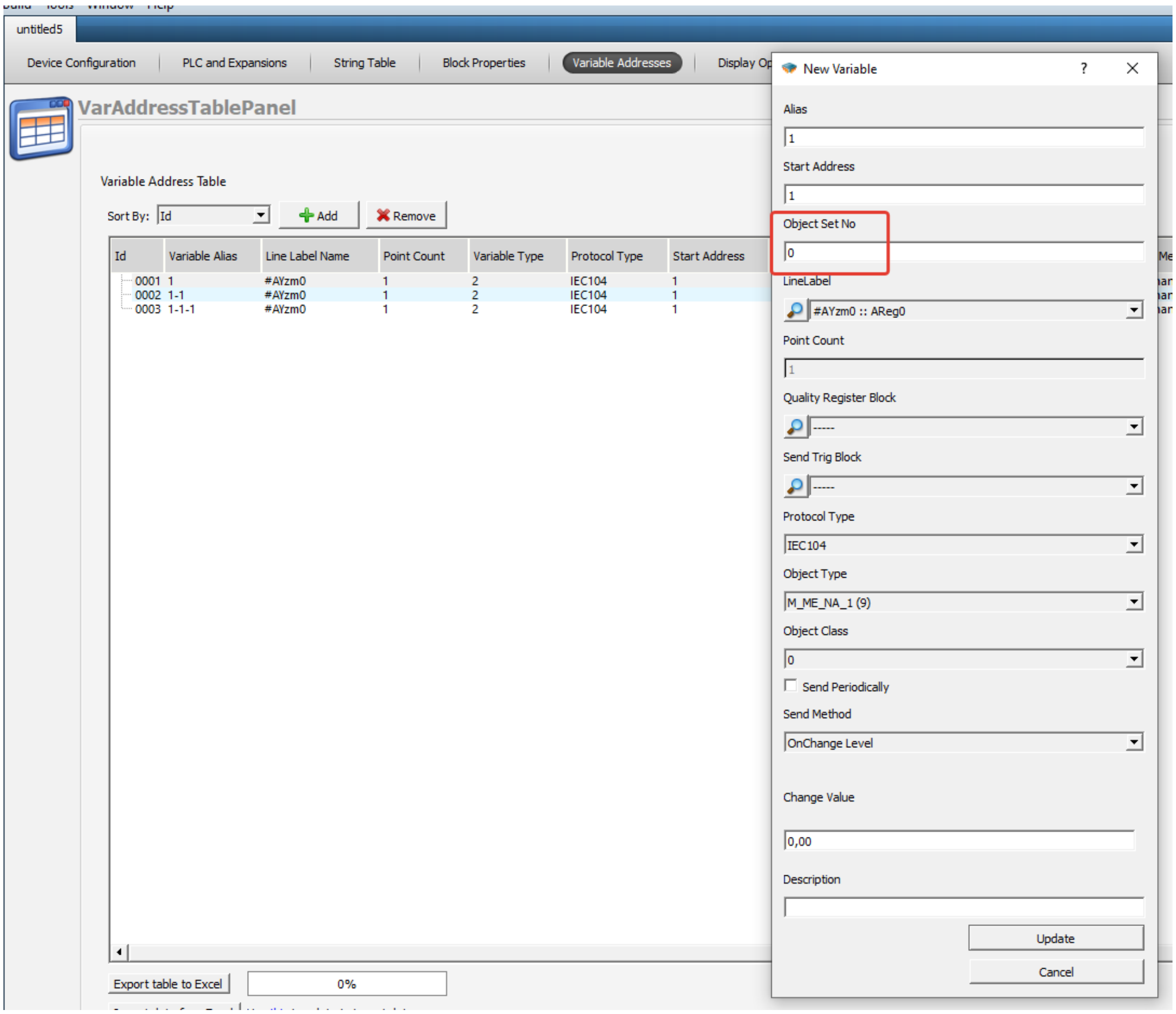

IEC104 Slave
Parameters
Line Definition
Notes
Block Name
I1041
AsduAddress
2
T0(Seconds)
30
T1(Seconds)
15
T2(Seconds)
10
T3(Seconds)
20
K
12
W
8
Group Count
2
Max Client in Group
5
Object Sets
2
Add to log-record memory
Sync with DevNET
LCD/Web View / MQTT Format
No View / MQTT Block Numbers
View Only / MQTT Line Labels
View and Set / MQTT Line Labels
OK
Cancel
HI
#HI0
B: 4
M: 1004
TSB
#TSB0
B: 0
M: 1000
I104
B: 2
M: 1002
#HI0-1
B: 9
M: 1009
#TSB0-1
B: 5
M: 1005
B: 7
M: 1007
#HI0-1-1
B: 14
M: 1014
#TSB0-1-1
B: 10
M: 1010
#I1040-1
B: 12
M: 1012


TCP Socket Block
Parameters
Line Definition
Notes
Block Name
TSB2
Tcp Socket
SocketType
TCP Client
TCP Server
TCP Client
Server Port
0
Server IP
TCP Server
Listen Port
2406
IP Filter
Media Type
Ethernet
Add to log-record memory
Sync with DevNET
LCD/Web View / MQTT Format
No View / MQTT Block Numbers
View Only / MQTT Line Labels
View and Set / MQTT Line Labels
OK
Cancel
HI
#HI0
B: 4
M: 1004
TSB
#TSB0
B: 0
M: 1000
#HI0-1
B: 9
M: 1009
#TSB0-1
B: 5
M: 1005
#HI0-1-1
B: 14
M: 1014
#TSB0-1-1
B: 10
M: 1010

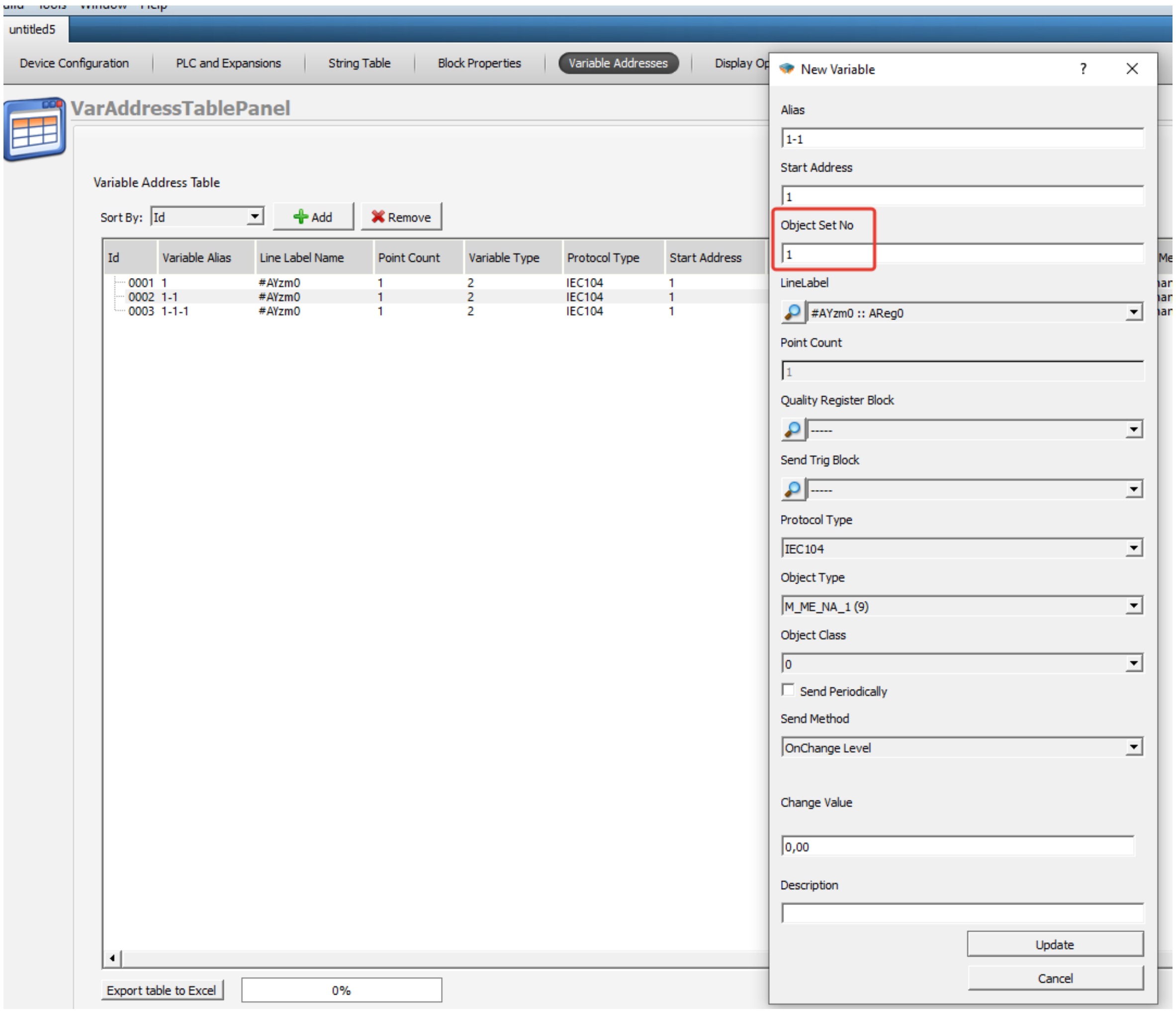


In the example application, three different IEC104 Slave blocks were defined for three different listening ports. During the configuration, each IEC104 Slave block was assigned a different "Object Sets" value.

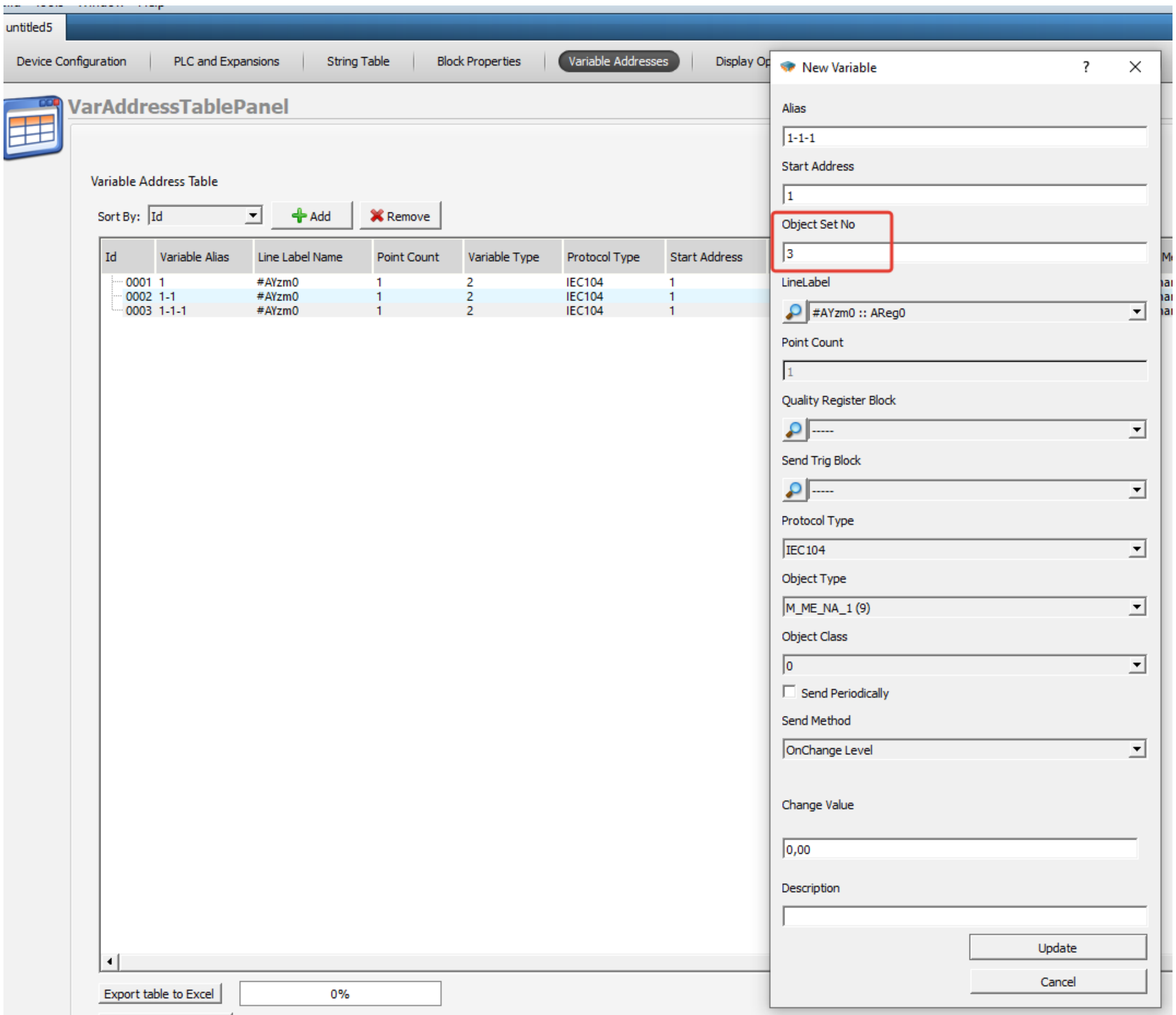
For listening port 2404, the "Object Sets" value of the IEC104 Slave block is specified as 1 in the block settings. Therefore, the corresponding "Object Set No" value in the variable address table is entered as 0. (2^0=1)



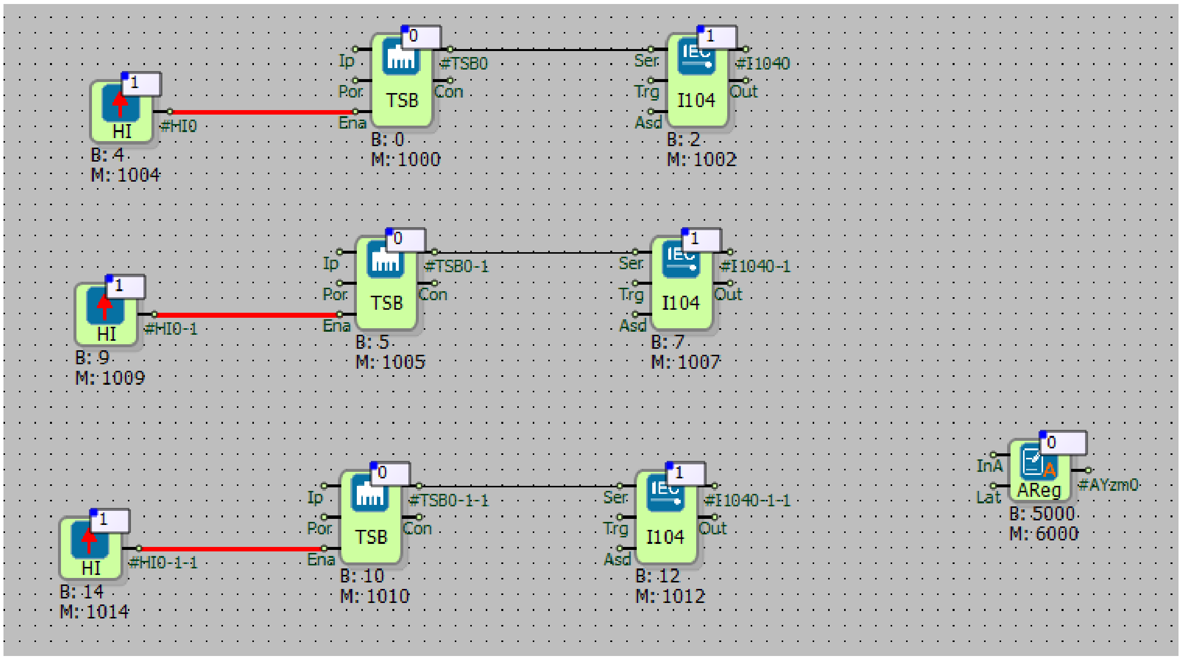
For listening port 2405, the "Object Sets" value of the IEC104 Slave block is specified as 2 in the block settings. Therefore, the corresponding "Object Set No" value in the variable address table is entered as 1. (2^1=2)



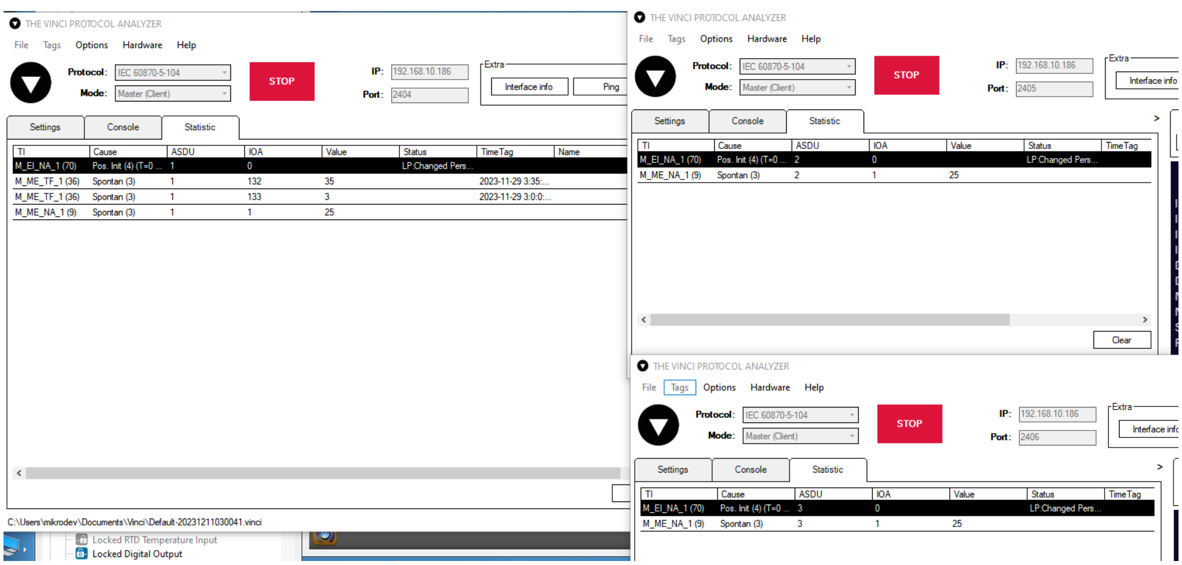
For listening port 2406, the "Object Sets" value of the IEC104 Slave block is specified as 8 in the block settings. Therefore, the corresponding "Object Set No" value in the variable address table is entered as 3. (2^3=8)



IEC 104 objects have been defined in the variable address table. A TCP connection has been established to the device, and online monitoring has been initiated.



IEC104 Masters were opened for different listening ports through the Vinci application, and the transmitted values were monitored.



11.3.6 IEC104 Object Types

11.3.6.1 IEC104 Read Direction Object Types

IEC 104 Object	Pe
1 (single-point)	Binary, Word, Analog, Long
3 (double-point)	Word, Analog, Long
5 (step position)	-
7 (bitstring)	-
9 (measured normalized value)	Binary, Word, Analog, Long
11 (measured scaled value)	-
13 (measured short floating point)	Binary, Word, Analog, Long
15 (integrated totals)	-
20 (packed single-point)	-
21 (normalized value without quality descriptor)	-
30 (single-point information with time tag)	Binary, Word, Analog, Long
31 (double-point information with time tag)	Word, Analog, Long
32 (step position information with time tag)	-
33 (bitstring of 32 bit with time tag)	-
34 (measured normalized value with time tag)	Binary, Word, Analog, Long
35 (measured scaled value with time tag)	-
36 (measured short floating point number with time tag)	Binary, Word, Analog, Long
37 (integrated totals with time tag)	-
38 (event of protection equipment with time tag)	-
39	-
40	-

11.3.6.2 IEC104 Object Types in Control Direction

The write variable is also automatically created for each block matched with the read type.

Variable types that can be accessed as writing to defined read objects are as follows:

Selected for reading IEC 104 Object Type	IEC 104 Object Type That can be accessed for writing to the same data point
1 (single-point)	45 (single command) 58 (single command with time tag)
3 (double-point)	46 (double command) 59 (double command with time tag)
13 (measured short floating point)	50 (set point command, short floatingpoint) 63 (set point command, short floating-point number with time tag)
30 (single-point information with time tag)	45 (single command) 58 (single command with time tag)
31 (double-point information with time tag)	46 (double command) 59 (double command with time tag)
36 (measured short floating point numberwith time tag)	50 (set point command, short floatingpoint) 63 (set point command, short floating-point number with time tag)

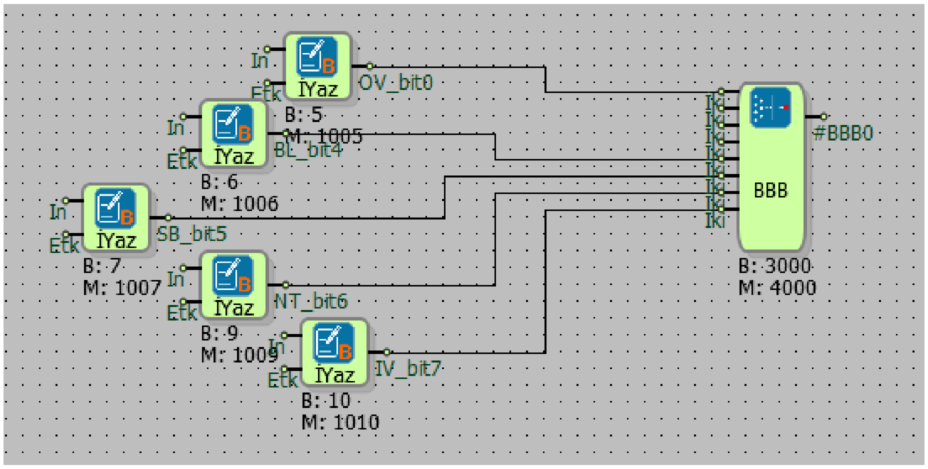
11.3.7 Quality Register Block Settings

Quality Descriptor (QDS) bits and accordingly Quality Register Block Settings are supported in our devices. Quality Descriptor bit definitions; OV, BL, SB, NT, IV, CY, CA, EI. As it is known, the use of QDS varies according to the defined object types. The Quality Descriptor (QDS) identification table is shown below.

Status / QDS	OV	CY	CA	EI	BL	SB	NT	IV
	overflow quality flag	carry flag	adjusted flag	elapsed flag	blocked quality flag	substituted quality flag	topical quality flag	invalid quality flag
1	overflow	carry	counter was adjusted	elapsed time not valid	blocked	substituted	not topical	invalid
0	no overflow	no carry	counter was not adjusted	elapsed time valid	not blocked	not substituted	topical	valid

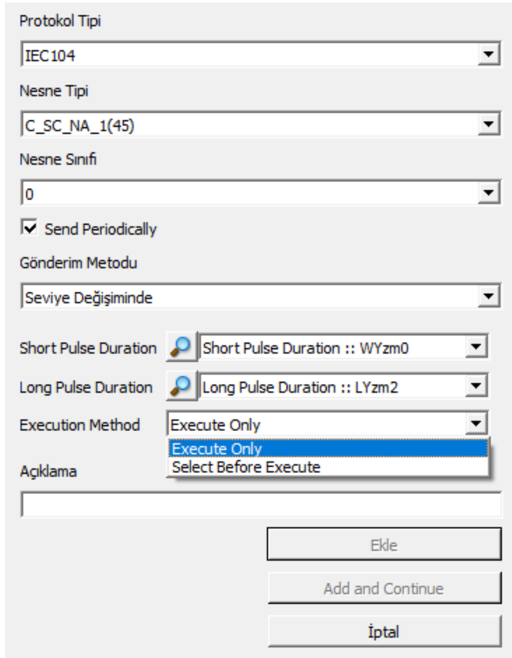
The QDS values to be used are created with the Bit Combination Block and defined by the Quality Register Block setting during the IEC 104 association in the variable addresses section.

For example, we will define the reading value with the IEC 104 protocol. We select 36 – measured short floating point number with time tag, as the reading object type. We will define QDS values for Quality Register Block definition. For this, the QDS bit definition is as follows; It should contain 0.bit OV, 4.bit BL, 5.bit SB, 6.bit NT, 7.bit IV. We can define Bit Combining Block as Quality Register Block.



11.3.8 Command Send Settings

It supports Single Command, Double Command and Set Point Command for appropriate object types in IEC 104 protocol. Object types command types mapping is shown in the Object Types Table. The settings are as follows; Depending on the object type, the options appear automatically in the selected IEC 104 protocol settings during line label association. For example, when Object type 45 (Single Command) is selected, options for parameter settings become active as seen in Figure 6. A register is selected for either Short Pulse Duration or Long Pulse Duration values. It should be noted that the entered value will be treated as ms. The Execution Method is also selected from the list. The Execution Method is of 2 types. Execute Only is selected if the operation is desired to be performed with a single command. If 2 different confirmation states are desired, Select Before Execute is selected. For example, the Select Before Execute option can be used for transactions that require confirmation with 2 different commands. For this, the Select command must be sent first and then the Execute command.



11.3.9 IEC 104 Event Mechanism

The variable address table has a send on exchange selection for IEC 104 objects. When the value of the variable defined in this menu changes, the action to be taken is selected.

On No Exchange: The spin submission is not triggered.

In Level Change: When the amount defined in “Change Value” changes, the sending is triggered.

In Percentage Change: Sending is triggered when there is a change in the percentage defined in “Change Value”.

The percentage or level of change is also set with the "Change Value" option. Sets the percentage or level change value along with the “Submission Method”.

The RTU device tags the statuses that are send on change and change detected as events and assigns a time tag to the event. In case of a tagged event, if there is a connection with the server, the relevant object is transmitted immediately as COT 0x03 Spontaneous.

If there is no connection with the server, the device is added to the event log memory and stored for sending when the server connection is established again. For storage, the option "Add to log-record memory" must be selected in the IEC104 Slave block.

Note: If all tags are to be sent to the server when the connection is established, the Sync with DevNET option must be selected in the IEC104 Slave block.

Note: The values of selected objects with periodic sending between IEC104 objects are not detected as events. That is, periodic submissions are not added to the log memory when there is no connection.

11.3.10 IEC104 Redundancy Group Specification

Mikrodev RTU can connect with IEC 104 Master as IEC 104 Slave. The number of Master IPs to be connected to this must be defined to the device with the AT command. The Mikroterminal application opens, from the special command entry section.

The command AT+OPTIONS=7,<NUMBER OF MASTER IP TO CONNECT> is sent.

For example, if Edaş has two different server IPs, this command would be as follows:

AT+OPTIONS=7,2

>> AT+OPTIONS=7,2 Write Commad

OPTIONS=OK

>> AT+OPTIONS=7,? Read Command

OPTIONS=2

After entering this parameter, the device must be reset. AT+RESET=1

11.3.11 Ability to Edit the Analog Threshold Value Retained in the Log Recording Memory

While there is no connection, changes can be made on the threshold values of the analog values kept in the log memory.

The Mikroterminal application opens, will be sent from the custom command line

AT+OPTION=8,<ANALOG EVENT MULTIPLIER>

analog event multiplier on the command line, analog log recorded when there is no connection, it allows to operate on the threshold values of the values. Values written here are from 0 if set differently, when there is no connection, the event threshold is multiplied by the coefficient here.

For example;

If AT+OPTION=8.0, analog events are not added to the log memory if there is no connection.

If AT+OPTION=8.1, it records the change in the log memory as much as the value entered in the variable table.

If AT+OPTION=8.10, a change that is 10 times larger than the value entered in the variable table will also be recorded in the log memory.

>> AT+OPTIONS=8,10 Write Command

OPTIONS=OK

>> AT+OPTIONS=8,? Read Command

OPTIONS=10

After entering this parameter, the device must be reset. AT+RESET=1

11.3.12 IEC104 Connection Information Learning Command

IEC104 connection information can be learned with AT command.

The Mikroterminal application opens, from the custom command input

The command AT+COMSTATUS=iec104 is sent.

IEC104 redundancy group number =2 command query example when there is no selected connection;

```
>>  AT+COMSTATUS=iec104
IEC104 CLIENT GROUP[0]:00000000
        isDataTransStarted:0
        NumofActiveConnections:0
        MaxNumberOfEvents:256
        RefInstance:200111b8
        EventItems:1000c800
        ObjMap:10005ab0
                connection[0]:00000000
                connection[1]:00000000
                connection[2]:00000000
                connection[3]:00000000
                connection[4]:00000000
IEC104 CLIENT GROUP[1]:00000000
        isDataTransStarted:0
        NumofActiveConnections:0
```

MaxNumberOfEvents:256

RefInstance:200115f8

EventItems:1000dc00

ObjMap:100064f4

connection[0]:00000000

connection[1]:00000000

connection[2]:00000000

connection[3]:00000000

connection[4]:00000000

COMSTATUS=

IEC104 redundancy group number =2 selected, command query example when there is only one connection;

```
>>  AT+COMSTATUS=iec104
IEC104 CLIENT GROUP[0]:4d0aa8c0
        isDataTransStarted:1
        NumofActiveConnections:1
        MaxNumberOfEvents:256
        RefInstance:2000f4c8
        EventItems:1000c800
        ObjMap:1000518c
                connection[0]:20010b30
                DataTransStarted: 1
```

connection[1]:00000000

connection[2]:00000000

connection[3]:00000000

connection[4]:00000000

IEC104 CLIENT GROUP[1]:00000000

isDataTransStarted:0

NumofActiveConnections:0

MaxNumberOfEvents:256

RefInstance:2000f908

EventItems:1000dc00

ObjMap:100052ac

connection[0]:00000000

connection[1]:00000000

connection[2]:00000000

connection[3]:00000000

connection[4]:00000000

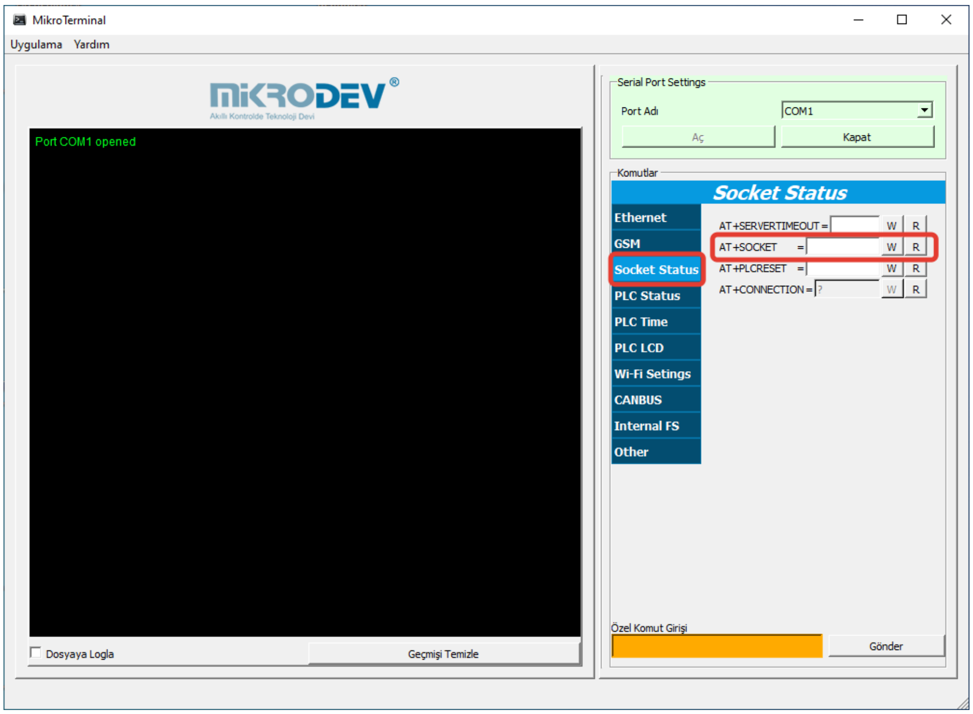
COMSTATUS=

11.3.13 Command to Learn IEC104 Master IPs Connected to TCP Socket Block

With AT command, IEC104 Master IPs connected to TCP Socket block can be learned.

The Mikroterminal application is opened, from the special command entry or from the command line that says AT+STATUS= in the Socket Status section.

The AT+SOCKET=<TCP Socket Block Number> command is sent.



Example of IEC104 Master IP Query connected to TCP Socket block with 4 block numbers;

>> AT+SOCKET=4

Ip: 172.21.1.1, Port: 65063, Status: 2

Ip: 172.21.1.1, Port: 65514, Status: 3

Ip: 172.21.1.2, Port: 46076, Status: 2

Ip: 172.21.1.2, Port: 45799, Status: 2

```
>>  AT+COMSTATUS=iec104

IEC104 CLIENT GROUP[0]:020115ac
        isDataTransStarted:1
        NumofActiveConnections:2
        MaxNumberOfEvents:256
        RefInstance:200110b0
        EventItems:1000c800
        ObjMap:10005ab0
                connection[0]:20014b78
                        DataTransStarted: 1
                connection[1]:00000000
                connection[2]:20013430
                        DataTransStarted: 0
                connection[3]:00000000
                connection[4]:00000000
IEC104 CLIENT GROUP[1]:010115ac
        isDataTransStarted:1
        NumofActiveConnections:2
        MaxNumberOfEvents:256
        RefInstance:200114f0
        EventItems:1000c900
        ObjMap:100064f4
                connection[0]:20013838
                        DataTransStarted: 0
```

connection[1]:00000000

connection[2]:20014770

DataTransStarted: 1

connection[3]:00000000

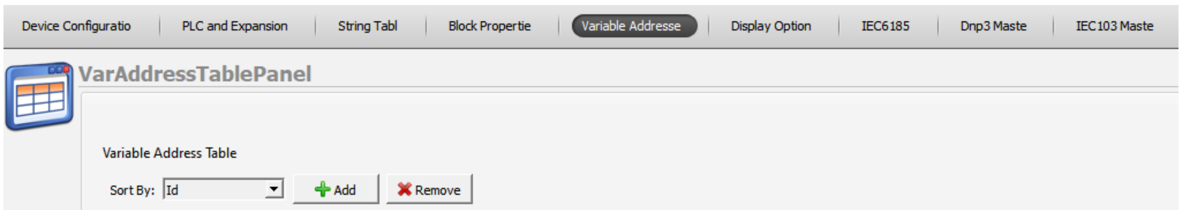
connection[4]:00000000

COMSTATUS=

11.3.14 Variable Mapping with Protocol

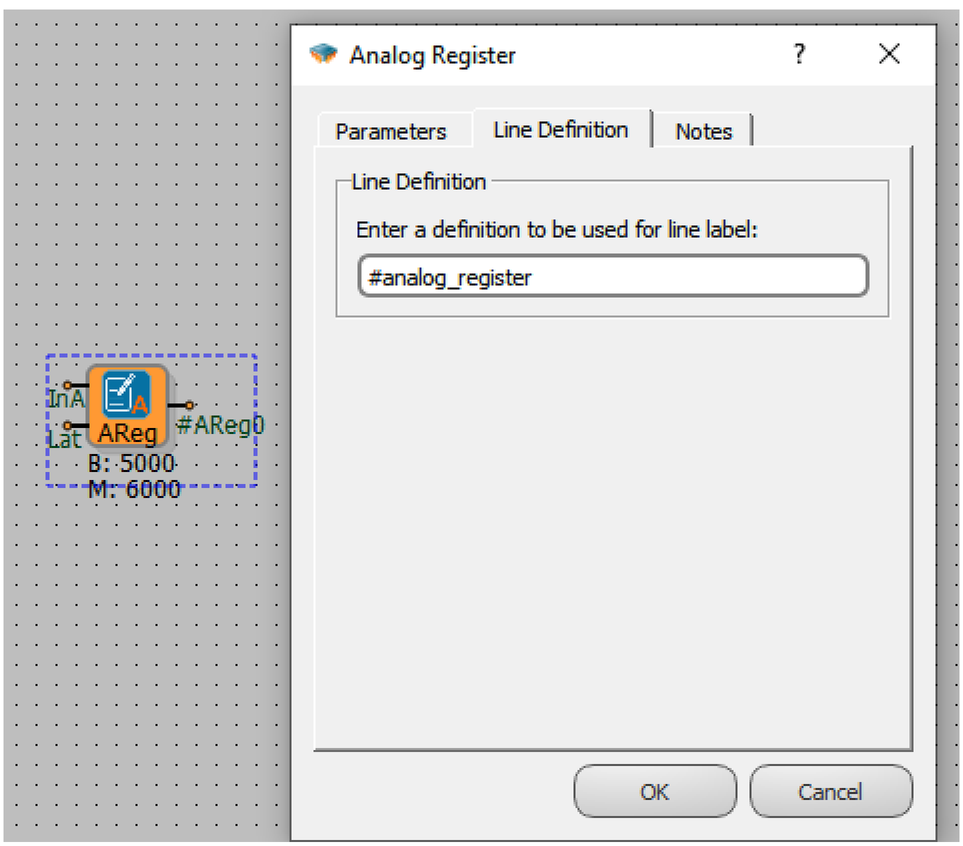
11.3.14.1 Variable Address Table

The relevant protocol is activated in the RTU logic project by adding the protocol block. Variables in the RTU logic Project, association between protocol.is provided in te variable address table



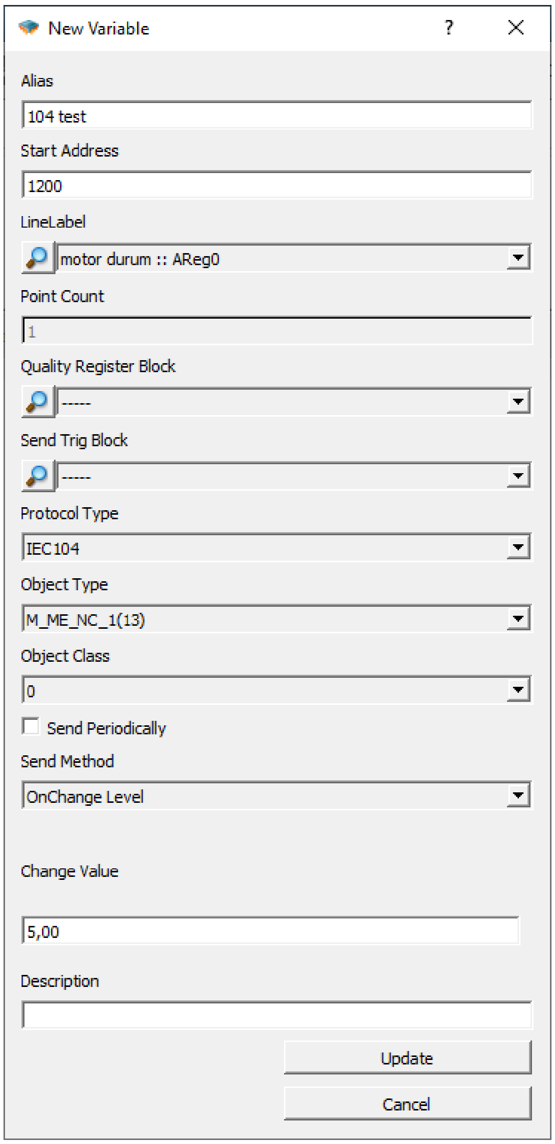
11.3.14.2 Defining Line Labels

Line label can be defined for all blocks defined on the Mikrodiagram. In the variable table, the line label must be defined in order to be able to associate with the protocol addresses.



11.3.14.3 Attaching a Line Label

Associating protocol tags with line labels, variable address is provided from the menu by pressing "Add" button in the address table.



Alias: A special name is given that defines this defined variable.

Start Address: The address allocated for this variable on SCADA is written here. It is written as a decimal value

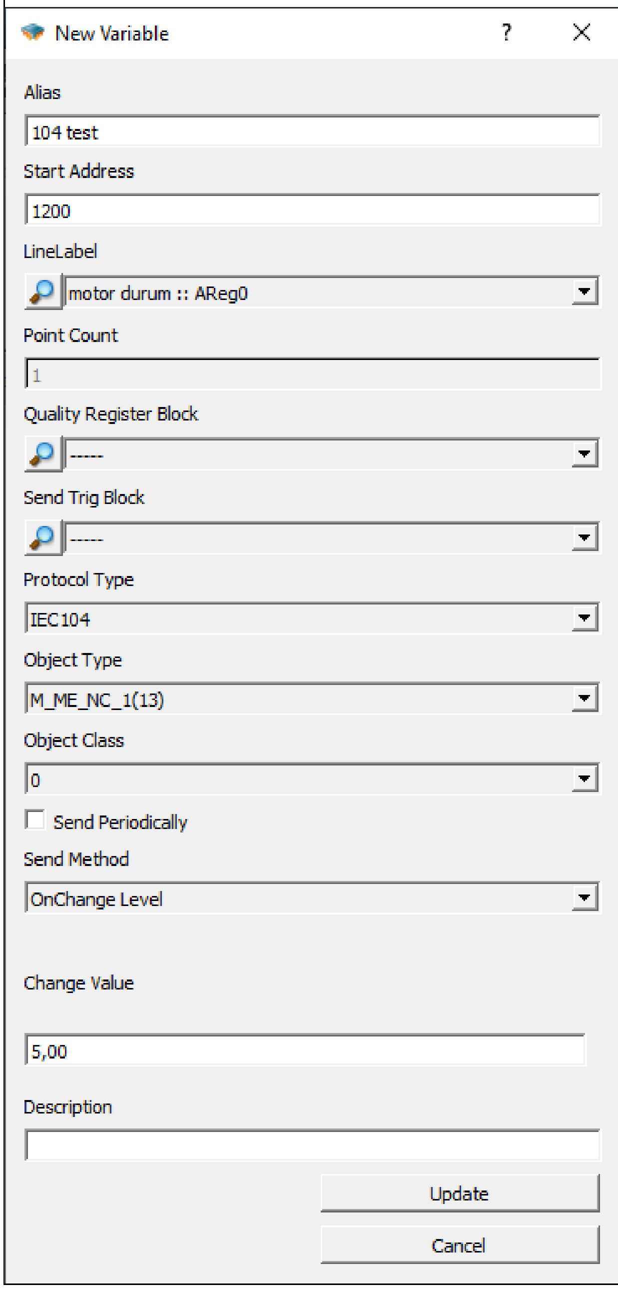
Line Label: The block to be associated on the Mikrodiagram is selected with the line label.

Point Count: Calculated automatically. It makes sense on tables.

Quality Register Block: Block entry to define Quality Register

Send Trig Block: If you want to send IEC104, DNP3 etc. data with an independent trigger from the trigger input of the block, the trigger block is selected from this section and the periodic send option in the block special settings must not be ticked in order to send trigger-dependent data here.

Protocol Type: Modbus, Dnp3, IEC101, IEC104 are selected. Object type will change according to protocol type.

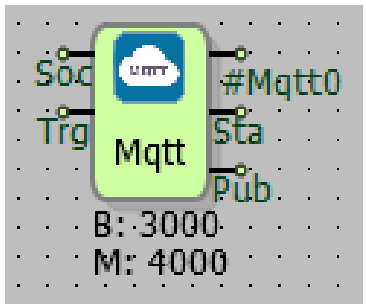


Object Type: IEC104, DNP3 etc. object type information selected. look the protocol type information for detailed information.
Object Class: The class information to which the variable belongs is selected.
Send Periodically: It is the selection of whether to send periodic sending to SCADA in this variable when the trigger is detected from the Trigger input on the IEC104 Slave block.
Send Method: When the value of the defined variable changes, the action to be taken is selected. On No Change/ None: The spin submission is not triggered. Level: When the amount defined in “Change Value” changes, sending is triggered. Percentage: Sending is triggered when there is a change in the percentage defined in “Change Value”.
Change Value: Sets the percentage or level change value together with the “Send method”.

12 MQTT BLOCKS

12.1 MQTT CONFIG BLOCK

12.1.1 Connections

Soc: TCP Socket entry		#Mqtt0: : Block output
Trg: Block trigger input		#Sta: Communication status output
		#Pub: Publish timeout output

12.1.2 Connection Explanations

<u>Soc: TCP Socket entry</u>

It is used for TCP Socket block connection. Mqtt Config block cannot be used without TCP Socket block.

<u>Trg: Block trigger input</u>

When periodic data transfer is desired, a trigger should be given to the mqtt config block from this input. If this entry is left blank, data is transmitted according to other specified conditions.

<u>#Mqtt0: Block output</u>

Output showing the connection status. The information from this output is as follows;

0: TCP Disconnected

1: TCP Connecting

2: MQTT Connecting

3: MQTT Connected

<u>#Sta: Communication status output</u>

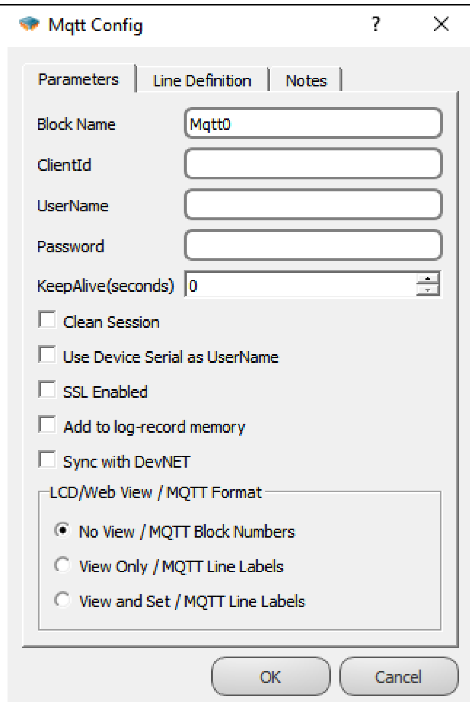
Output showing the communication status. The information from this output means:

- 0: MQTT Send Conn Pack
- 1: MQTT Idle Status
- 2: MQTT Subscribe Status
- 3: MQTT Publish Status

#Pub: Publish timeout output

Output showing Publish timeout

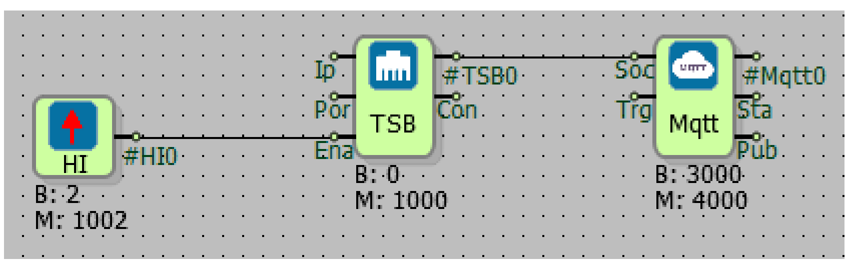
12.1.3 Block Settings:



Client Id: The field where the device is manually given an ID for the broker connection.
User Name: The field where the device is named for the broker connection.
Password: Password field entered in the device for the broker connection.
Keep Alive: If the connection between the broker and the Publisher is lost, the waiting time before reconnecting.
Clean Session: If selected, messages will be broadcast if there is communication between the device and the broker, otherwise the information recorded in communication interruptions will not be sent.
Use Device Serial as User Name: If selected, the serial number of the device is used as the device username.
SSL Enabled: It is marked to make the connection with SSL. (Only active in DM Series.)

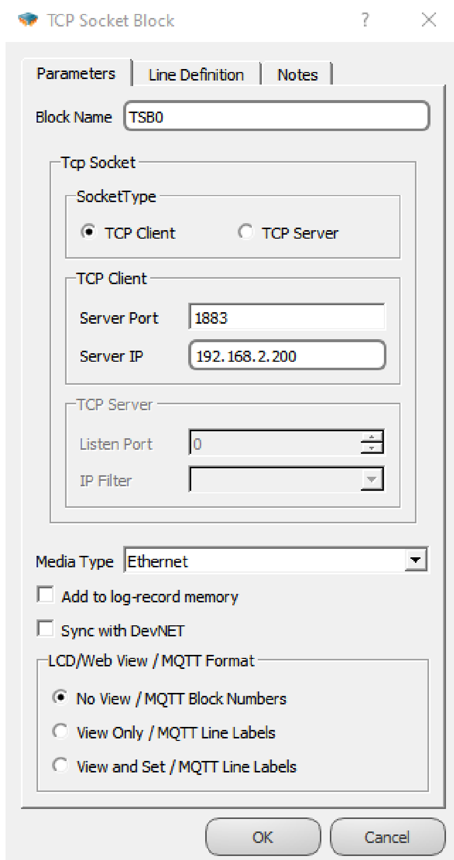
12.1.4 Block Explanation

The output of the TCP Socket block is connected to the Soc input of the Mqtt Config block.



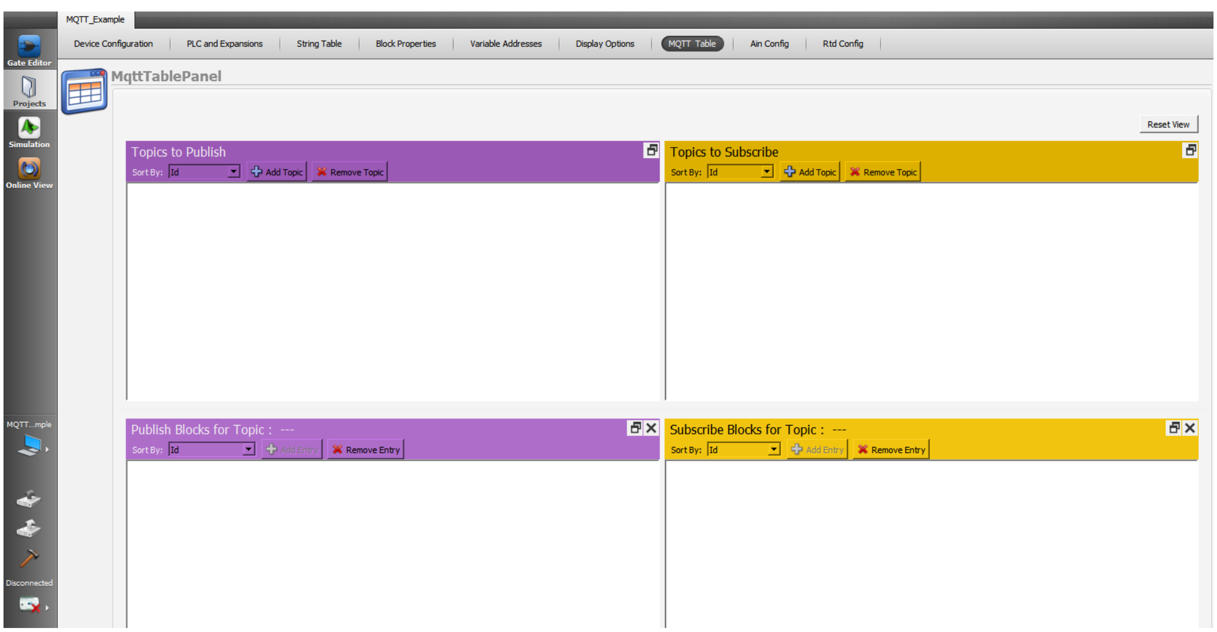
The special settings of the TCP Socket block should be made for mqtt connection as follows;

- TCP Client should be selected as the socket type,
- The mqtt server IP to be connected to the Server IP section must be entered,
- Mqtt server port information should be entered in the Server Port Section,
- As for the media type, Ethernet, GSM or WI-FI can be selected according to the characteristics of the microdev device used.



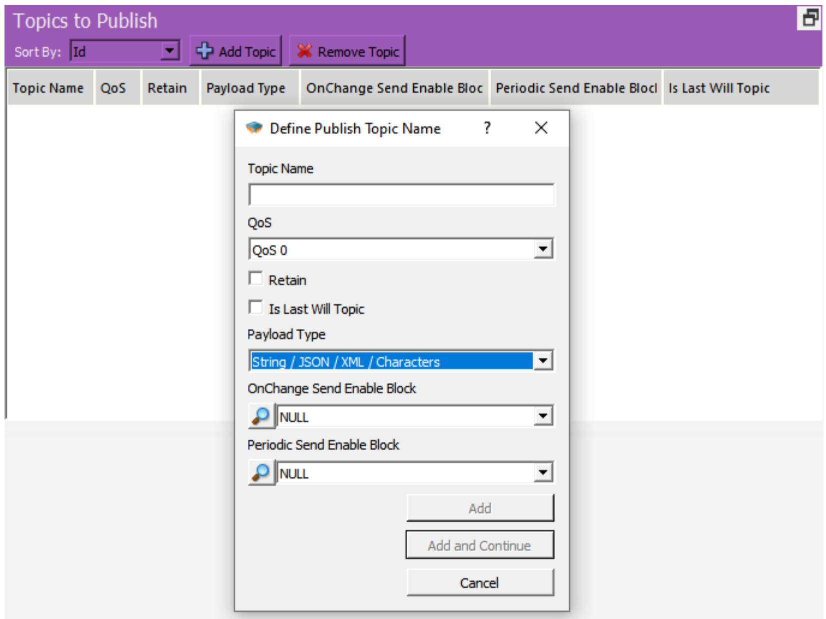
12.2 MQTT TABLE

The table where all MQTT-related adjustments are made can be accessed from the Projects/MQTT Table tab.



12.2.1 Topics to Publish

In this table, the Publish topic is entered to publish the data to the broker. The topic name is entered on the screen that appears by pressing the Add Topic button in the table. Block definitions where you can enable/disable Qos, Retain, Last Will, Payload settings, send on exchange and periodically send options are also made on this page.



Definitions;

• Topic Name: The field where the topics you will send the messages are determined.

• QoS: Quality of Service refers to the agreement between the sender of a message and the receiver of the message. The QoS levels are as follows;

o QoS 0: Minimum data transfer is ensured. At this level, each message is forwarded to a subscriber and no confirmation is received that the message has arrived.

o QoS 1: The broker tries to transmit the message and waits for an acknowledgment response from the subscriber, if no confirmation is received within a specified time frame, the message is sent again.

o QoS 2: The broker receives two acknowledgments to ensure that the subscriber receives the message and only once.

• Retain: If this option is checked, if the connection between the broker and the subscriber is broken, the last value will be saved in memory.

• Is Last Will Topic: Last will topic. If a topic is created and this option is checked, the message under this topic will be forwarded to the subscribers when the device is disconnected from the broker.

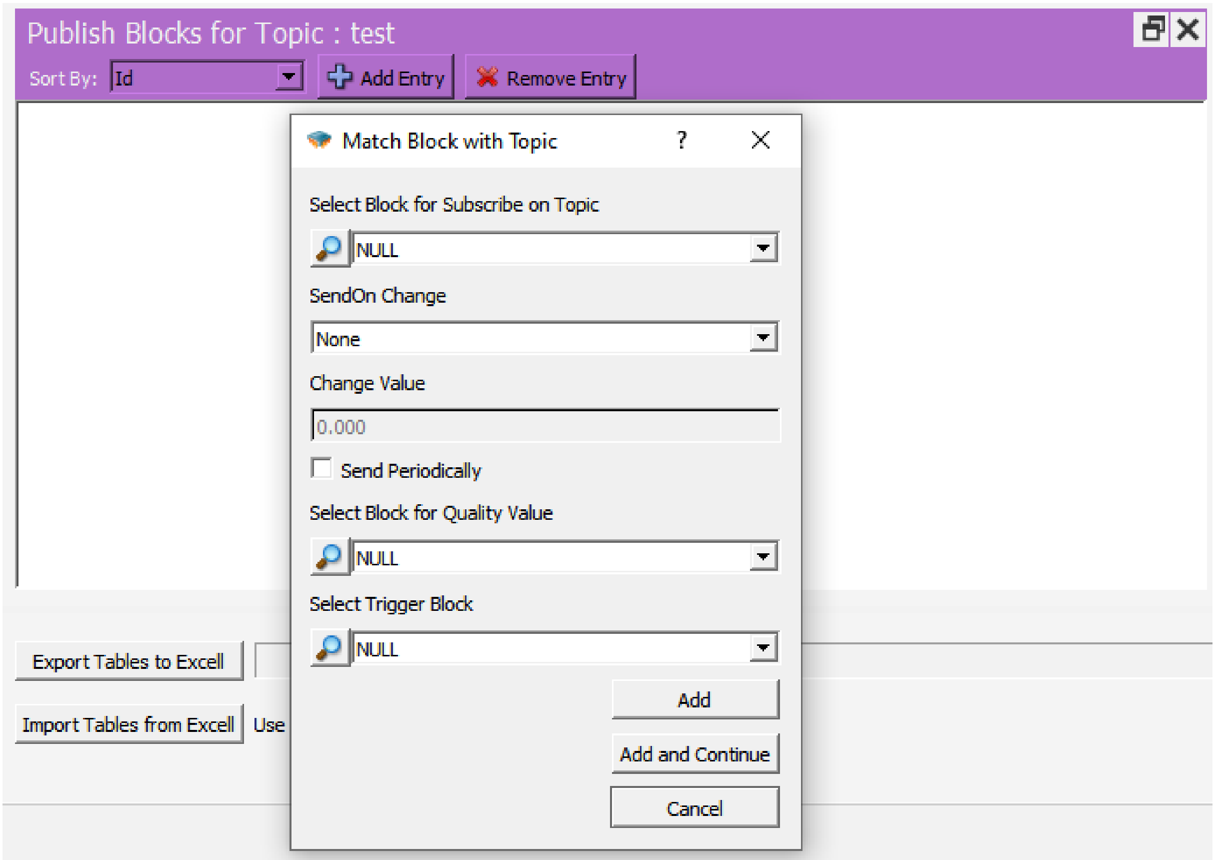
• Payload Type: It is determined in which format the message content will be sent. Subscriber interprets incoming messages with this information. “MJson v1” can be selected if a time stamp is desired to be added to the sent messages.

• On Change Send Enable Block: Block selection added in the diagram to enable or disable the sending feature of the created topic on change.

• Periodic Send Enable Block: Block selection added in the diagram to enable or disable the periodic sending feature of the created topic.

12.2.2 Publish Blocks for Topic

In this table, the blocks to be published for the relevant Topic are selected. After clicking the topic in the Publish to topic table, the Add Entry button becomes active and by pressing this button, the block to be published in the project is selected. How to transmit the data can also be selected from the screen



Definitions;

• Select Block for Subscribe on Topic: The area where the block that you want to send as a message in your project is selected.

• Send On Change: Send selection field on exchange

o On Level Change: Send when there is a change in the value specified in Change Value, if 0 is written, it will be sent in every change.

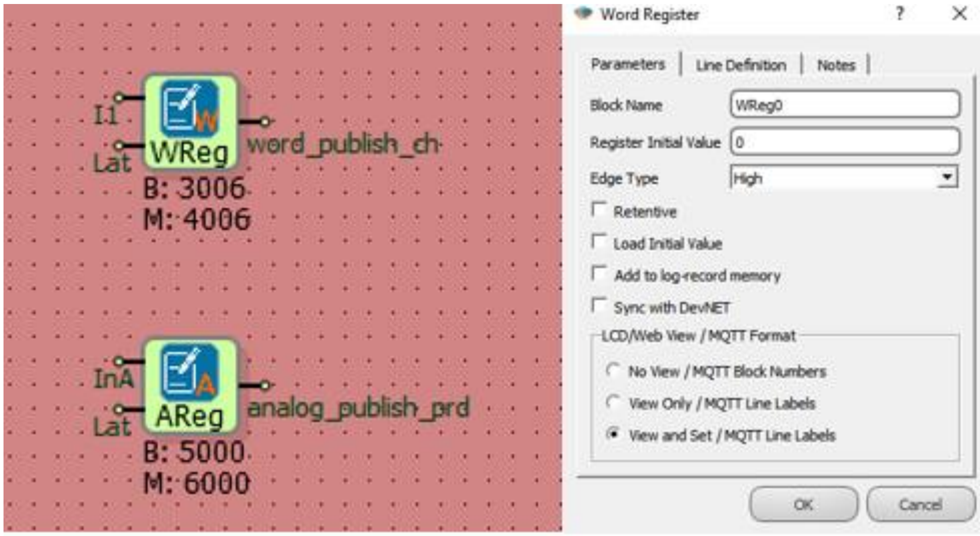
o On Percent Change: Send when there is a percentage change of value specified in Change Value, for example 10%.

• Change Value: Change amount input field.

• Send Periodcally: If checked, a message is sent every time a trigger comes to the trg input of the mqtt config block.

• Select Block for Quality Value: The block in which the Quality value included in the message content is selected in MJson v1 payload type.

• Select Trigger Block: Apart from change or periodicity, we can send the message by triggering the block we will specify here.



Note: Blocks used in messages; It can be sent and received with the block number (B:3006) under the block, or it can be added to the messages with line tags (word_publish_ch). This selection is made under the Mqtt Format tab in the block properties.

- Message that will appear if View and Set is selected;

```
{"word_publish_ch":2}
```

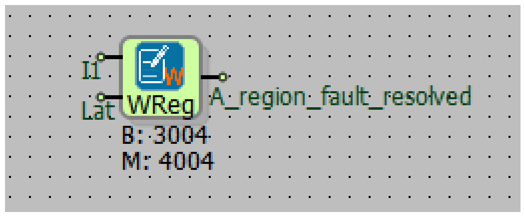
- The message that will appear if No View is selected;

```
{"3006":4}
```

MIKRODEV

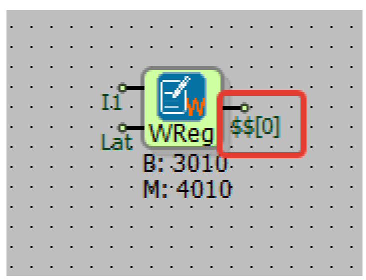
12.2.2.1 Identifying Labels from String Table

If long expressions are to be used in MQTT format for the selected label definitions, a text table can be used. This helps to avoid confusion in PLC projects. For example, if you want to send the expression "A_region_fault_resolved" as a line definition in MQTT format, you can use a string table.

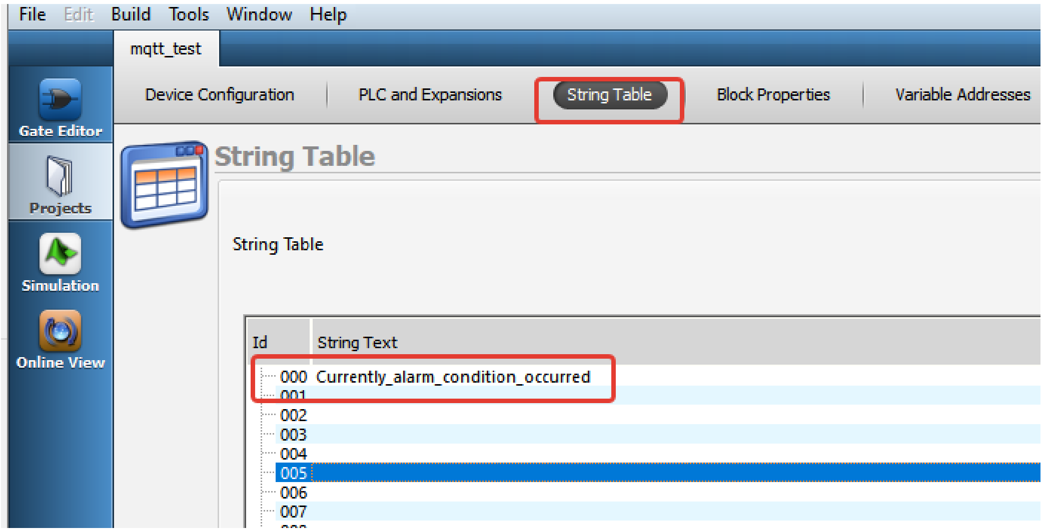


It should be written as shown in Figure 12. This will require more space in PLC projects. To avoid this issue, the string expression to be sent via MQTT can be entered into a text table. For this:

- Select "Mqtt format view and set";
- Enter the expression $$[string offset] into the line label definition, with the string offset part corresponding to the relevant string offset of the text table.

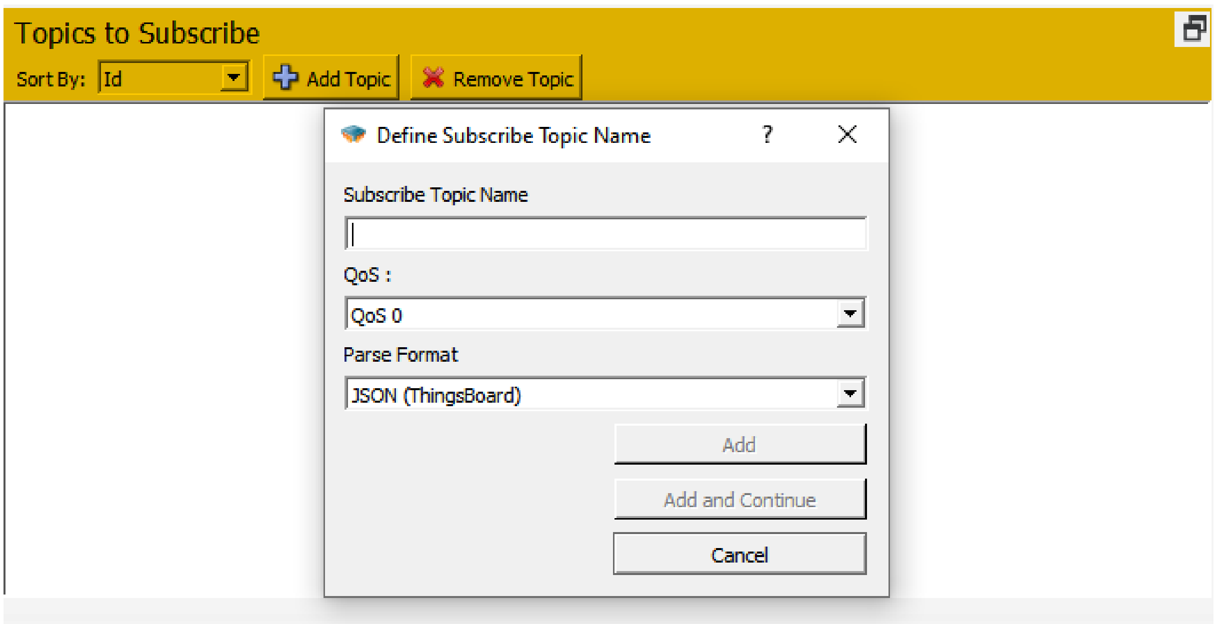


- Enter the expression you want to write into the string table.



12.2.3 Subscribe to Topic

In this table, the relevant subscribe topic is entered to send data from the broker to the device



Definitions;

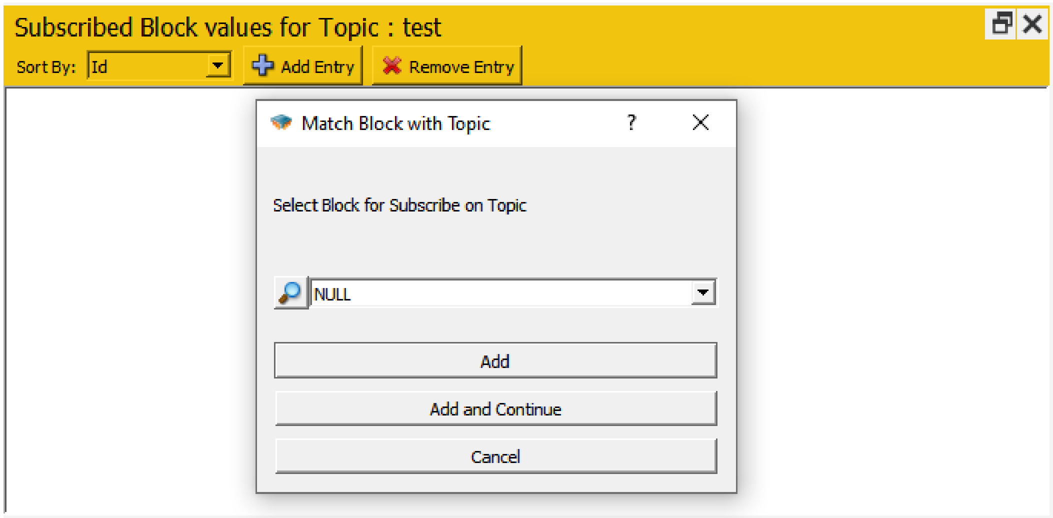
• Subscribe Topic Name: Enter the name of the topic to be subscribed to.

• QoS: Service quality level is selected.

• Parse Format: Select the format in which the messages will be parsed.

12.2.4 Subscribed Block Values for Topic

From this screen, the blocks to be associated for the subscribe topic are added. To use line tags, mqtt format should be selected as view and set from the special settings of the relevant block.

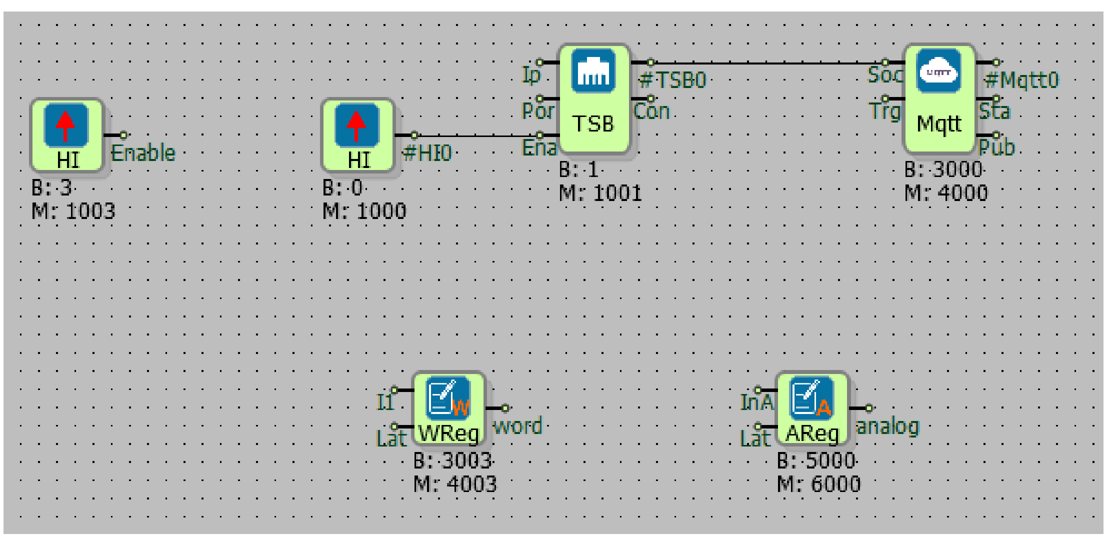


12.3 Sample Application

12.3.1 Topic Publish

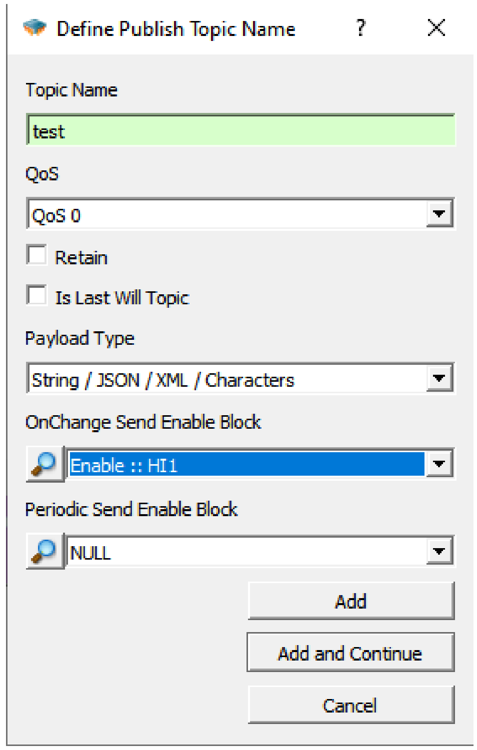
General Configuration;

After the project is created, the diagram is designed as in the figure, the mqtt formats of word and analog registers are selected as view and set.



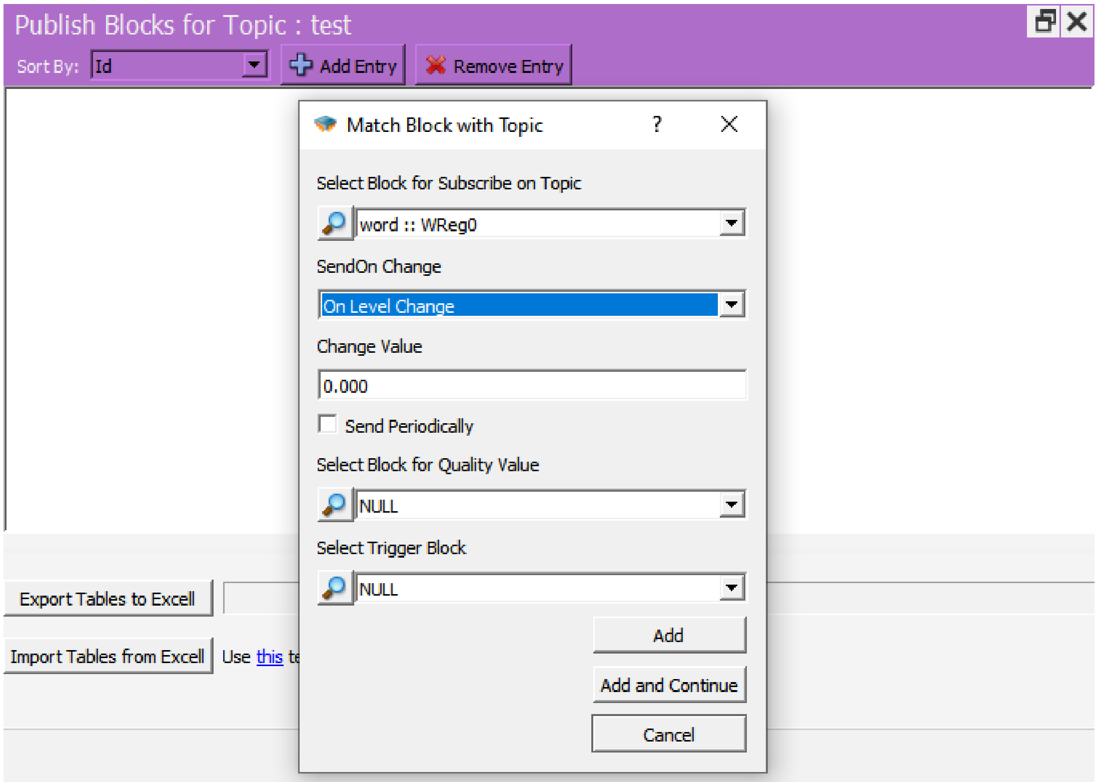
Send On Change;

Follow Projects > MQTT Table >Topics to Publish > Add Topic.



Select the Topic name, enter the High gate we have prepared in the diagram for the OnChange Send Enable Block, and click add.

Then, from the Publish Blocks for Topic section in a subtable, click to the Add Entry.



Select the block in the diagram that you want to broadcast as a message to the Select Block for Subscribe on Topic section.

In the SendOn Change section, On Level Change is selected and Change Value is set to 0 so that it can send a message every time the value changes. Click on Add and continue.

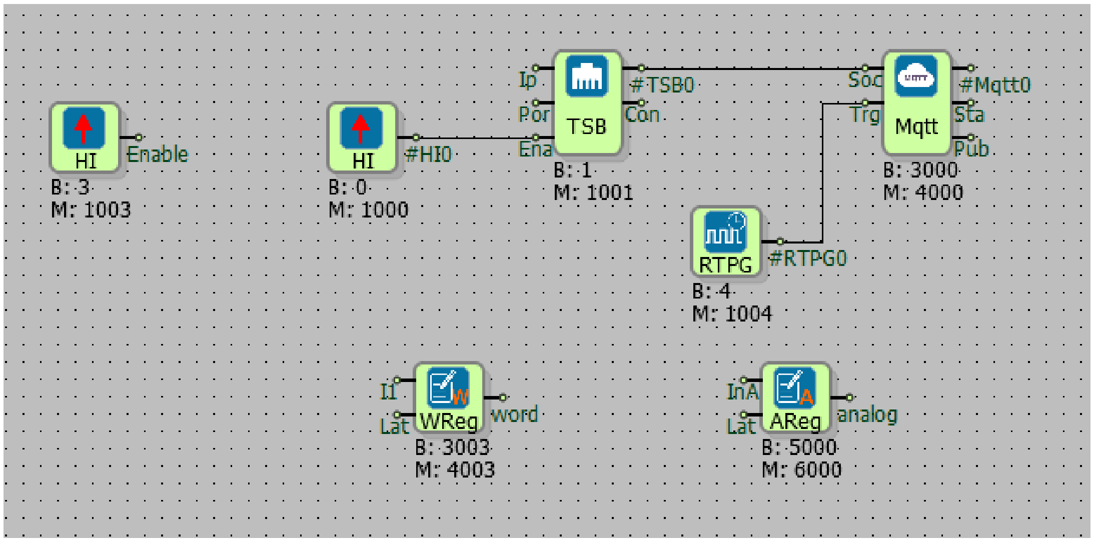
The project is loaded on the device and online monitoring is opened.

Subscribe to the topic opened with a program such as MQTTBox. After watching the mqtt config block value of 3 in online monitoring in the Mikrodiagram, when the value of the register is changed, it is seen that the value is published.

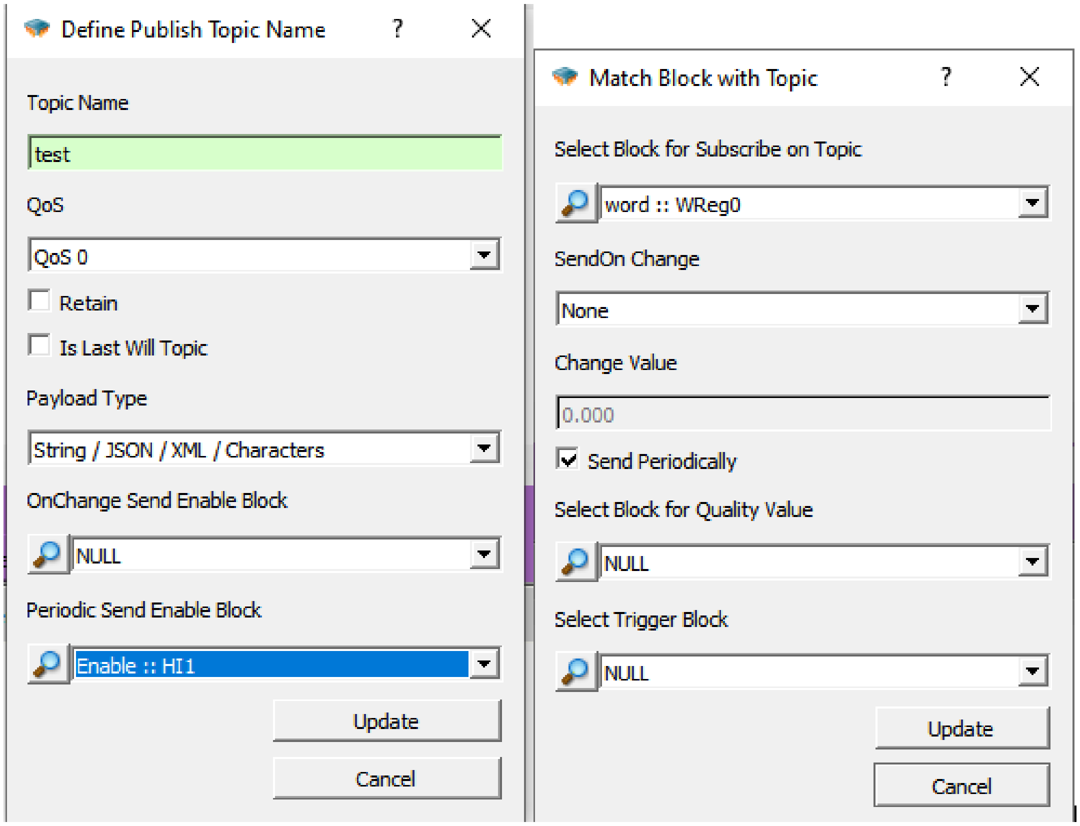

test_sunucu_mosquitto - mqtt://192.168.10.52:1883
test
{"word":23}
qos : 0, retain : false, cmd : publish, dup : false, topic : test, messageId : , length : 17, Raw payload : 12334119111114100345850511 25
{"word":19}
qos : 0, retain : false, cmd : publish, dup : false, topic : test, messageId : , length : 17, Raw payload : 12334119111114100345849571 25
{"word":9}
qos : 0, retain : false, cmd : publish, dup : false, topic : test, messageId : , length : 16, Raw payload : 1233411911111410034585712 5
{"word":0}
qos : 0, retain : false, cmd : publish, dup : false, topic : test, mes

Periodic Send;

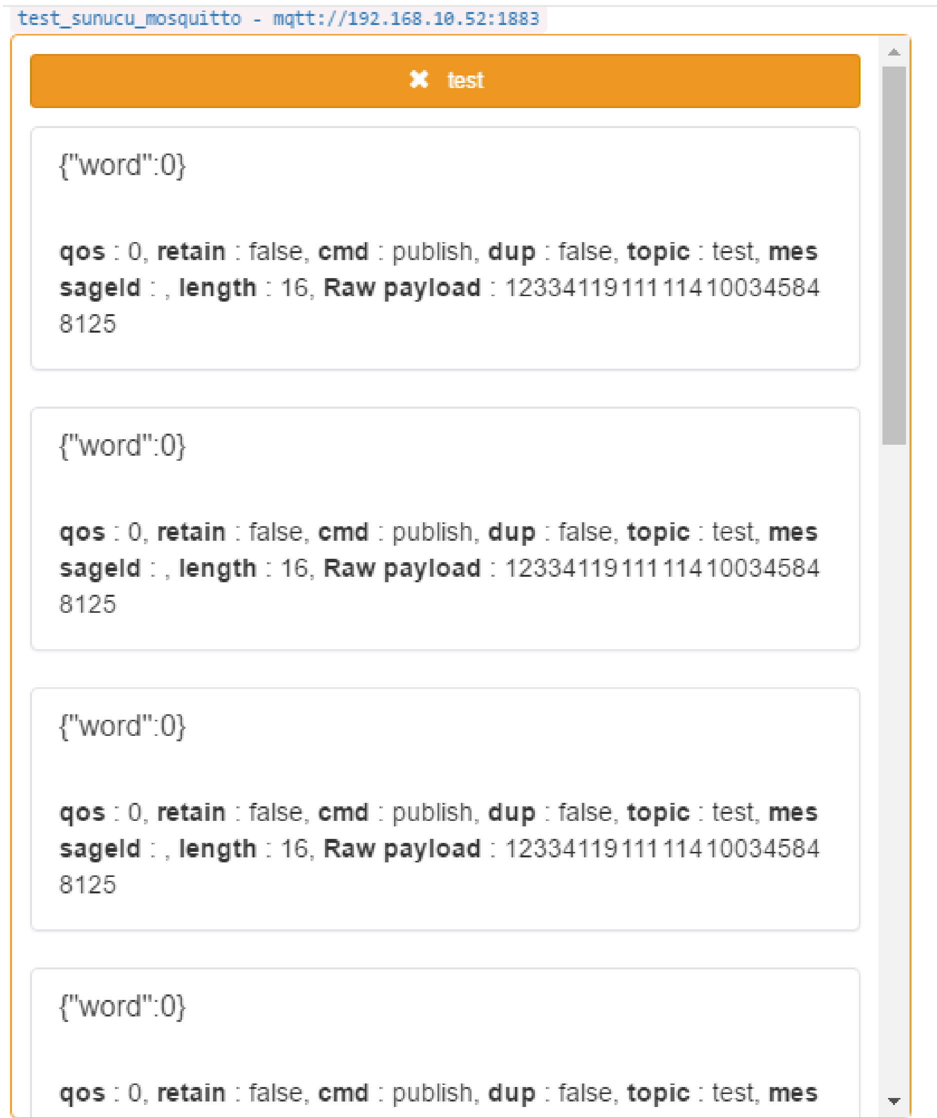
In addition to the configuration sent in the change, a real time pulse generator is added to the trg input of the mqtt config block,



Real time pulse generator is set for 5 seconds to broadcast a message periodically every 5 seconds and the created topic is changed as follows. OnChange Enable Block= NULL and Set the High gate in the Periodic Send Enable Block diagram, In the Select Block for Subscribe on Topic section, select SendOn Change= None and click Send Periodically.



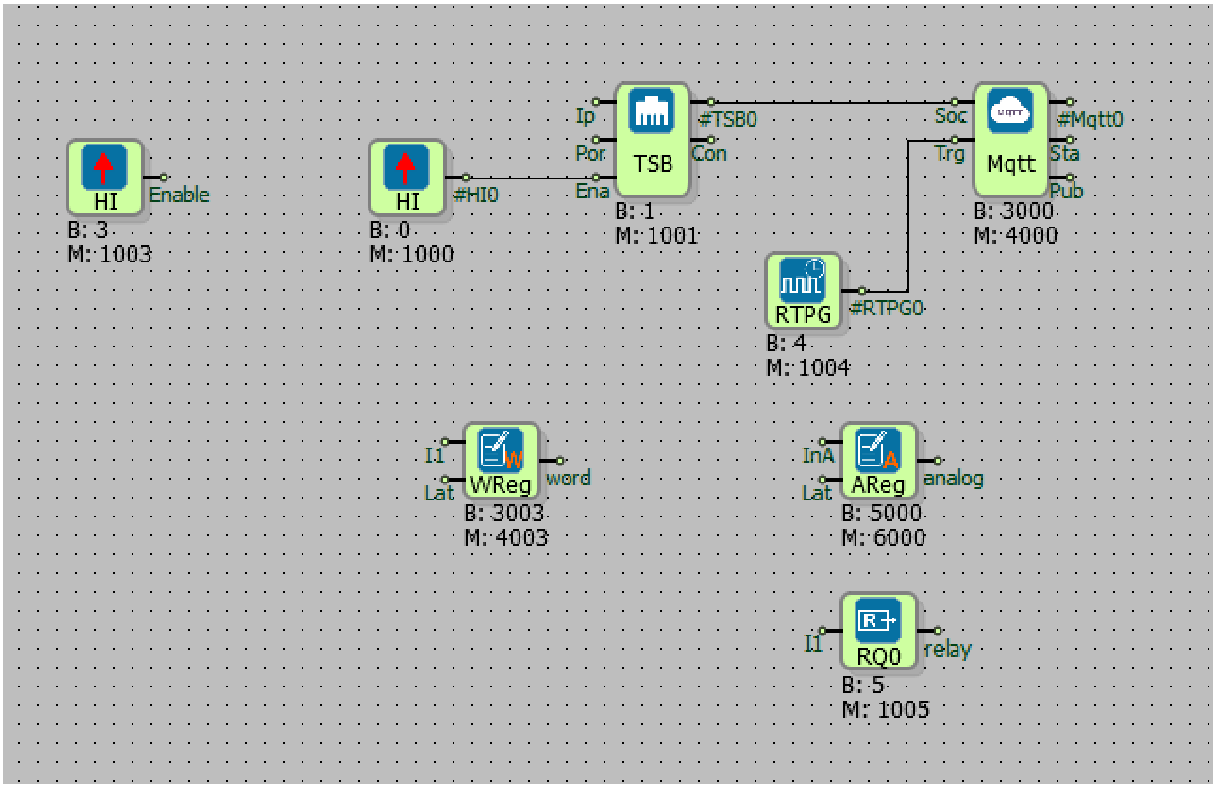
The project is loaded back to the device and incoming messages are observed.



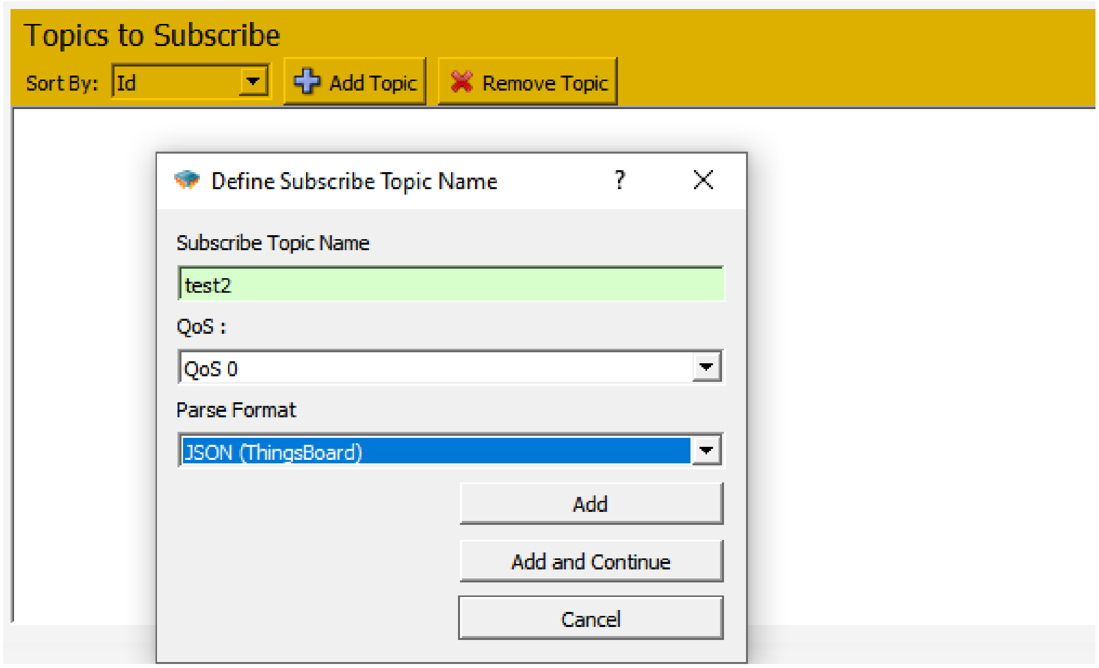
12.3.1.1 Subscribe Topic

General Configuration;

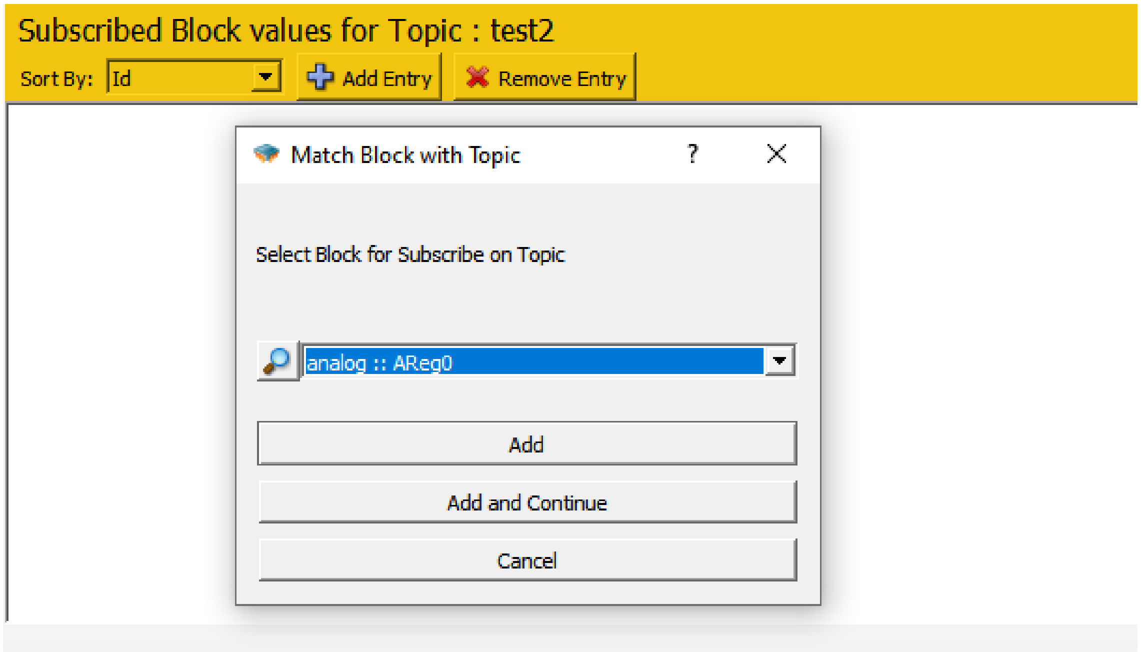
After the project is created, the diagram is designed as in the figure, the mqtt formats of the word and analog registers and the relay output are selected as view and set



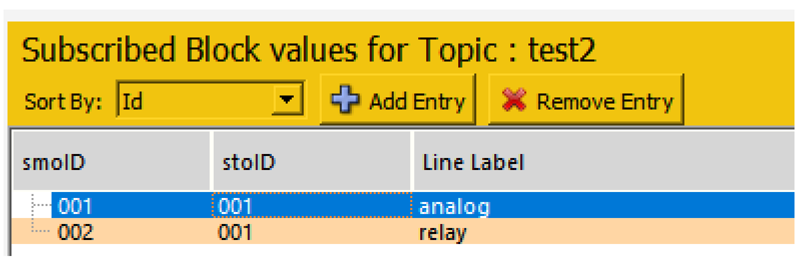
Follow Projects > MQTT Table >Topics to Subscribe > Add Topic.



Enter the topic name and click Add. Then, the add entry is clicked from the Publish Blocks for Topic section in a subtable.



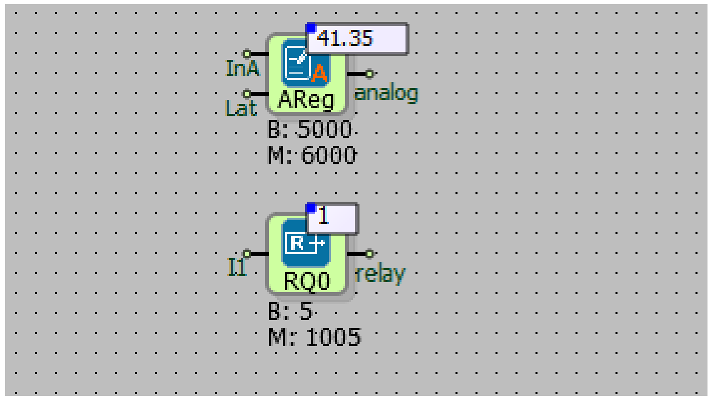
Here, the block to which the subscribed value will be transferred is selected.



After all blocks to be subscribed are determined, the project is loaded into the device.

{"relay":1, "analog":41.35}

When the message is published to the test2 topic with the above format, the final state of the variables is as follows;

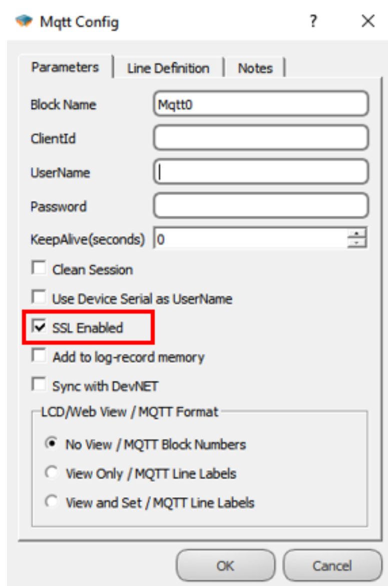


12.4 SETTING UP MQTT CONNECTION WITH SSL

Secure Sockets Layer (SSL) and Transport Layer security (TLS) are protocols that provide secure communications over a computer network or link. SSL/TLS provides data encryption, data integrity and authentication.

“SSL Enabled” option in block special settings of Mqtt Config Block; It provides secure MQTT connection with SSL Certificate. This option only active in DM Series.

In order to use this feature, an SSL Certificate must be uploaded on the device and the "SSL Enabled" option of the Mqtt Config block must be checked.



12.4.1 Uploading SSL Certificate File to Device

The SSL Certificate file can be uploaded on the device in two ways.

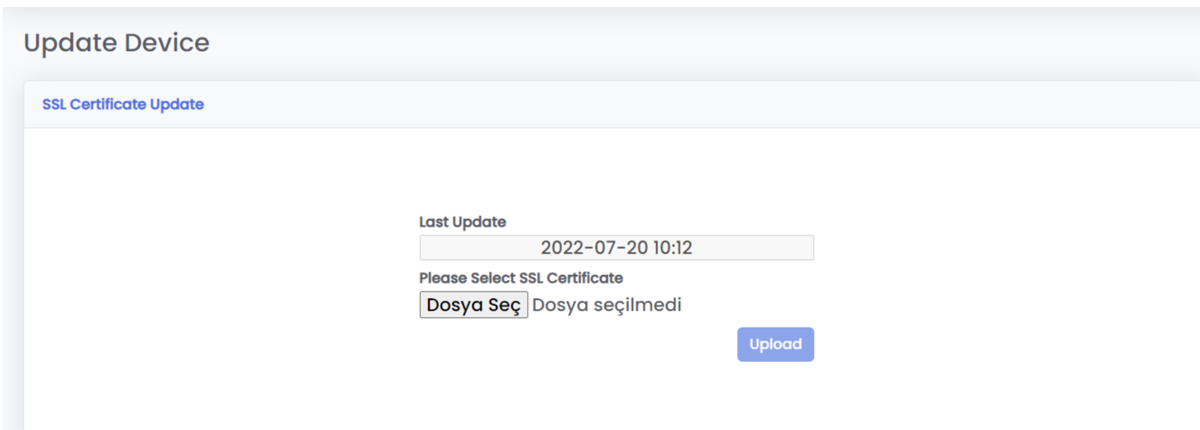
• SSL Certificate file can be uploaded via Web Server. For this, the following instructions are followed.

i. Login to the Web Server interface as an admin.

ii. Go to the Upload tab in the Web Server left sidebar.

iii. Click "Choose file" in the SSL Certificate Update section and select the SSL file you want to upload. Click "Upload" in the bottom right.

The certificate will be successfully installed on the device.



• SSL Certificate file can be installed via command prompt. For this, the following instructions are followed.

i. Change the certificate filename to combinet.crt

ii. Go to the windows command line.

iii. scp combined.crt root@<Cihaz IP>:/root //enter the command.

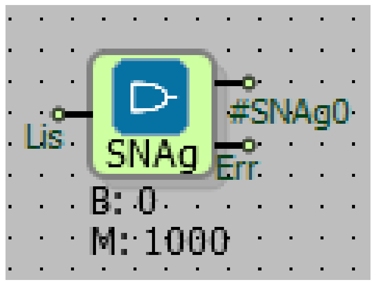
iv. After entering the command, it will ask for password, type your SSH password.

The certificate will be successfully installed on the device.

13 SNMP PROTOCOL BLOCKS

13.1 SNMP AGENT BLOCK

13.1.1 Connections

Lis: Listen port		#SNAg0: Block output
		Err: Error output

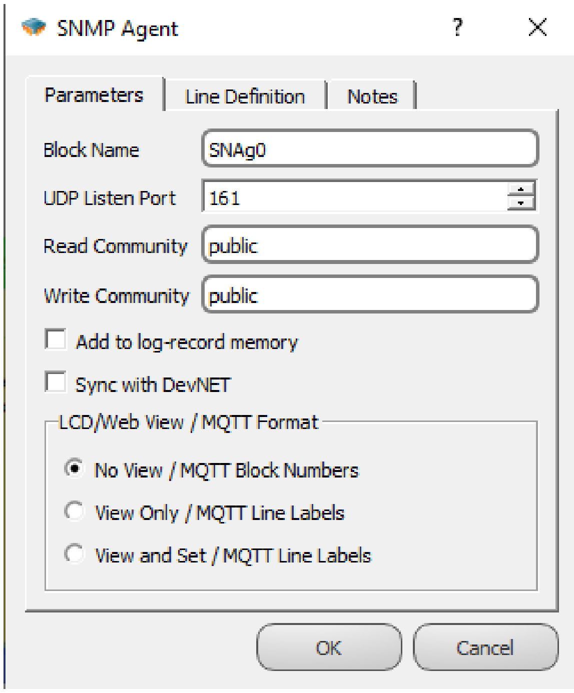
13.1.2 Connection Explanations

Lis: Listen port

#SNAg0: Block output

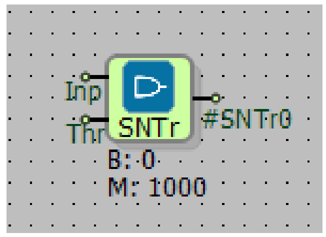
Err: Error output

13.1.3 Block Settings

	UDP Listen Port: Number of listen port, SNMP default is 161.
	Read Community: It works like username and password. Should be same on the manager.
	Write Community: It works like username and password. Should be same on the manager.

13.2 SNMP Trap Block

13.2.1 Connection

Inp: Input value input		#SnTr0: Block output
Thr: Threshold value input		

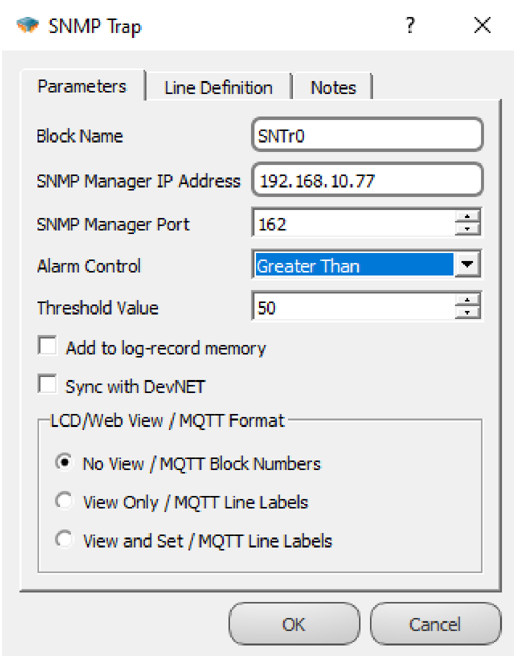
13.2.2 Connection Explanations

Inp: Input value

Thr: Threshold value input

#SnTr0: Block output

13.2.3 Block Settings

<table>
<tr><td rowspan="4">
</td><td>SNMP Manager IP: The manager's IP address.</td></tr>
<tr><td>SNMP Manager Port: Port number of manager.</td></tr>
<tr><td>Alarm Control: Conditions of send data.</td></tr>
<tr><td>Threshold Value: Input threshold value.</td></tr>
</table>

13.2.4 Block Explanation

- Alarm Control: Conditions of send data;
  - Greater Than: If the input value is greater than the threshold value, data is sent.
  - Smaller Than: If the input value is smaller than the threshold value, data is sent.
  - Level Change: If the input value changes by the threshold value, data is sent.
  - Percent Change: If the input value changes by the %threshold, data is sent.

This technology, called Simple Network Management Protocol, is designed to control units on computer networks as they grow. Various information from the temperature on the device to the users connected to the device, from the internet connection speed to the system operating time are kept in the tree structure defined in SNMP.

The components that make up SNMP are;

• Agent application: The name given to the application that runs the SNMP service on the device and saves the necessary information and transfers it to the administrator unit or applies the change from the administrator unit to the device.

• Admin application: The application that receives the information needed from the agent application and displays it to the user and sends the values that the user wants to change to the device.

• Network Management System (NMS): It is the name given to the application that runs in the administrative unit and provides monitoring and management of all devices connected to a network.

Mikrodev products work as SNMP Agent application.

SNMP Management Information Units (MIBs): MIBs are collections of information stored in a hierarchical structure. To reach the value of a particular variable in SNMP, the relevant unit of this universally determined collection is used. For example, for the device description assigned by the manufacturer of a device, it is necessary to access the information in the 1.3.6.1.2.1.2 unit.

13.2.5 OID CALCULATION

The query to be sent to the agent device by the manager in the SNMP protocol should be in the following structure;

.1.3.6.1.2.1.Block Type.Block Number

The block type can be the following values;

- 1: Bool
- 2: Word
- 3: Float
- 4: Long

The block number should not be written as in the Microdiagram, but should be converted to the following format;

Block No: 3004 >> 30*100+4 = 30.4

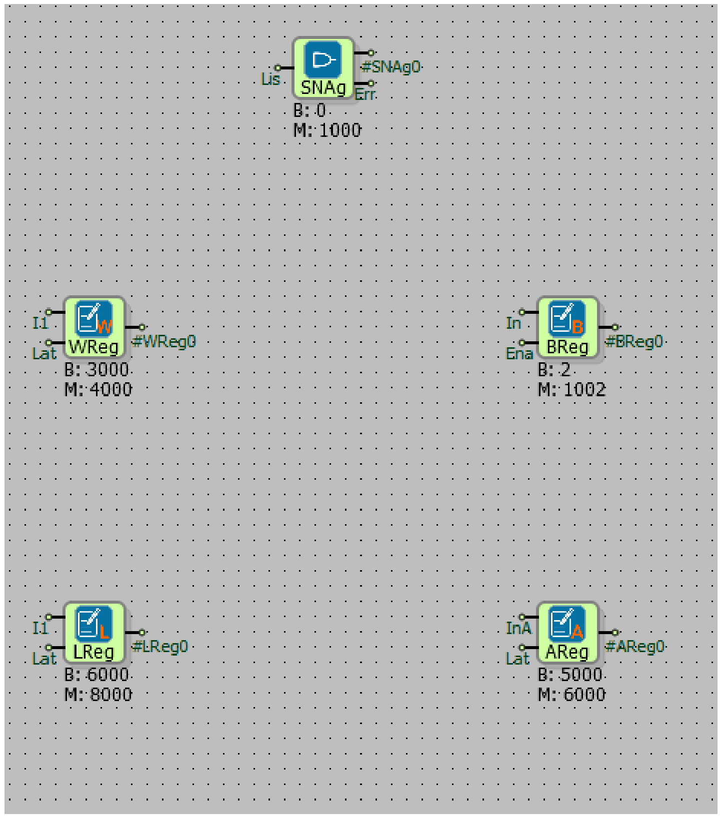
Block No: 3000 >> 30*100+0 = 30.0

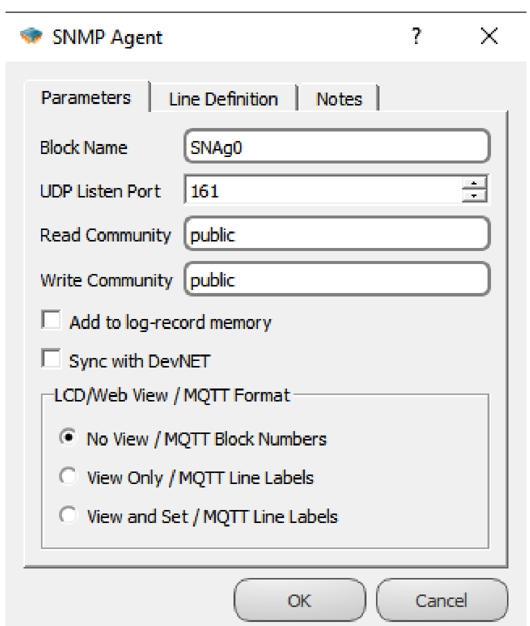
Block No: 6000 >> 60*100+0 = 60.0

13.2.6 Sample Application

13.2.6.1 SNMP Agent Sample Application

An SNMP Agent block and Word, binary, long, analog registers are added to the Project as in the Figure.





In Figure, the OID query that the manager will send to get the values of the registers in the project should be as follows;

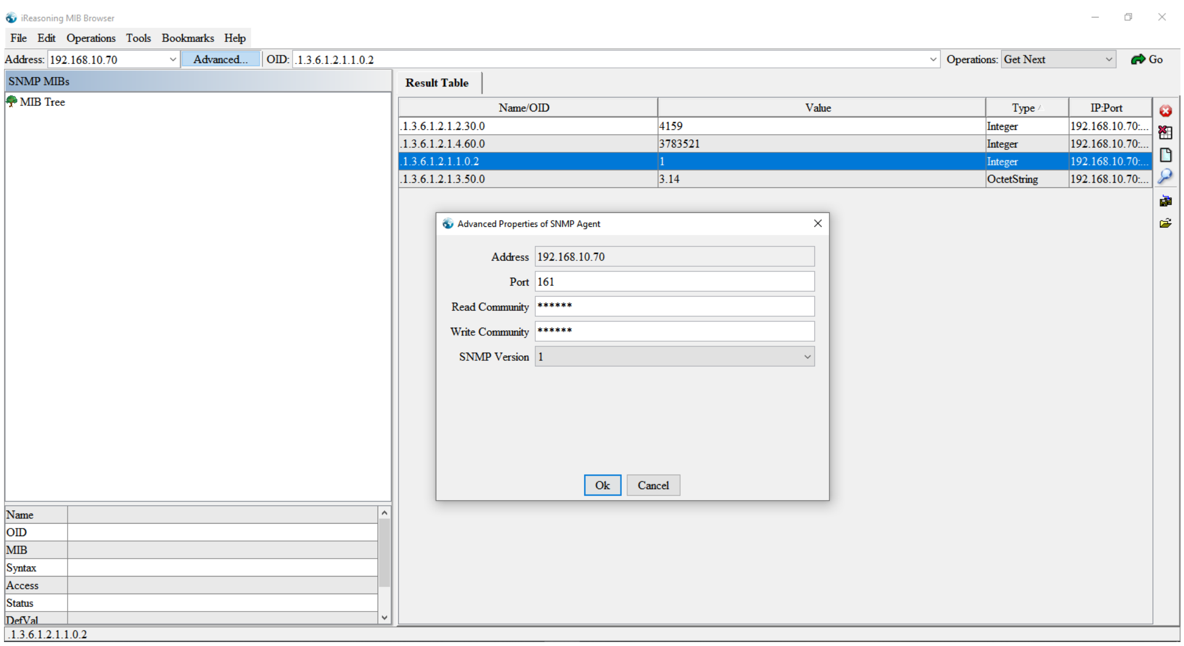
Binary register with Block Number 2: .1.3.6.1.2.1.1.0.2

Word register with Block Number 3000: .1.3.6.1.2.1.2.30.0

Analog register with Block Number 5000: .1.3.6.1.2.1.3.50.0

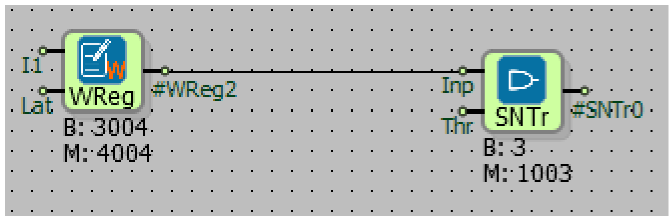
Long register with Block Number 6000: .1.3.6.1.2.1.4.60.0

MIB Browser program can be used as manager. After the IP and port settings are made, if the above queries are sent from the OID section, the values of the registers will appear as in the Figure.

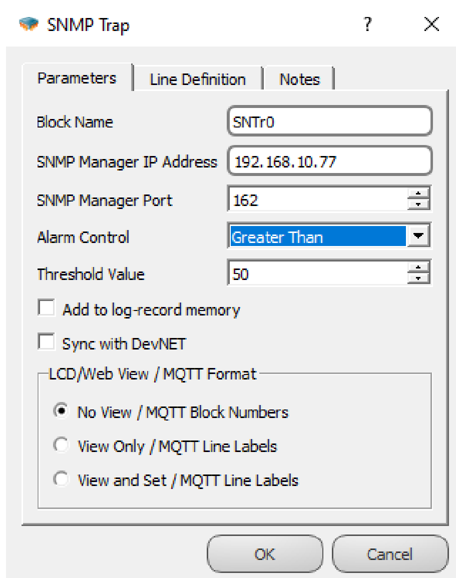


13.2.6.2 SNMP Trap Sample Application

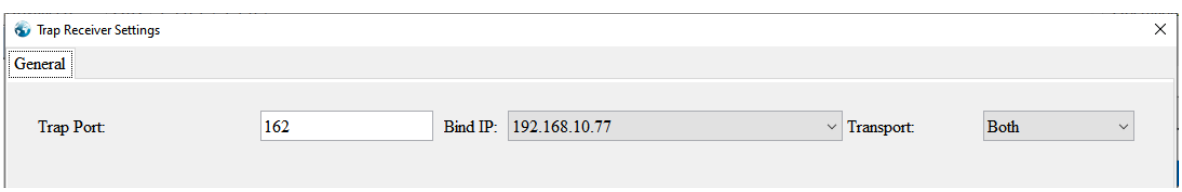
The way the agent sends data to the manager without sending a query is the Trap method. In this method, the register connected to the input of the Trap block is sent according to the conditions in the Trap block.



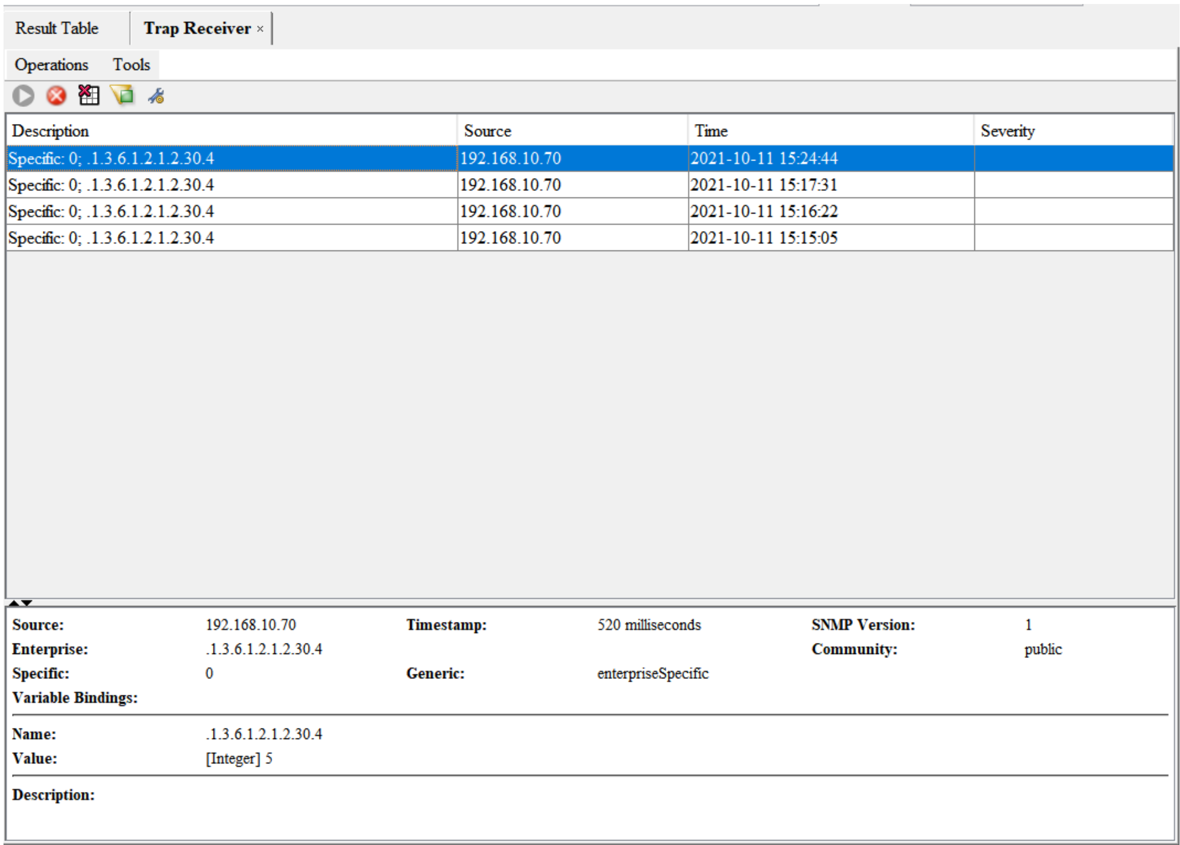
Trap settings are entered as follows.



MIB Browser Trap settings are also made as follows.



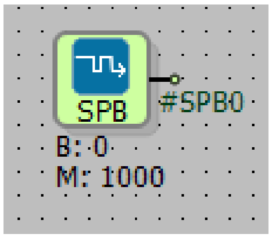
If the register value at the input of the trap block is changed and the condition specified in the block settings is met, it will appear on the trap receiver as follows.



14 COMMUNICATION BLOCKS

14.1 SERIAL PORT BLOCK

14.1.1 Connections

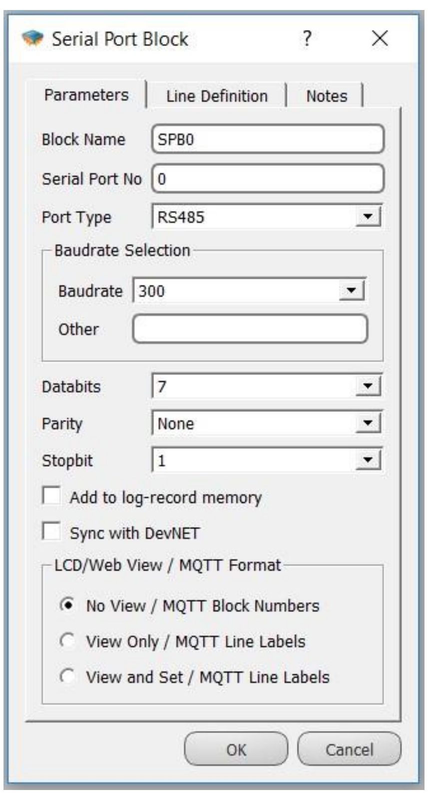
	#SPB0: Serial connection output

14.1.2 Connection Explanations

<u>Out: Serial Connection Output</u>

Output of the block which is connected to the protocol blocks.

14.1.3 Block Settigns

	Serial Port No: Port number is entered here.
	Port Type: Communication type is selected here.
	Baudrate: Baud rate is entered here.
	Other: Different Baudrates entered here.
	Databits: Data bits number.
	Parity: Parity is entered here.
	Stopbit: Stopbit is entered here.

14.1.4 Block Explanation

Any protocol supported by Mikrodev PLC can be configured to communicate over serial port. For this purpose, Serial Port block must be connected to related protocol block in PLC project. Serial Port Block can be used with following protocols of Mikrodev PLC/RTU:

- Modbus RTU Master
- Modbus RTU Slave
- DNP3
- IEC101
- Modbus Gateway mode
- Transparent Serial Gateway mode

Note: Only one serial port block can be defined for the same serial port on a device.

Serial Port Block Settings:

Serial Port No

Serial port no is used to select which serial port of PLC will be used. To learn the correct port number for this selection, which is related to the PLC hardware, refer to the Hardware Manual of the corresponding PLC model.

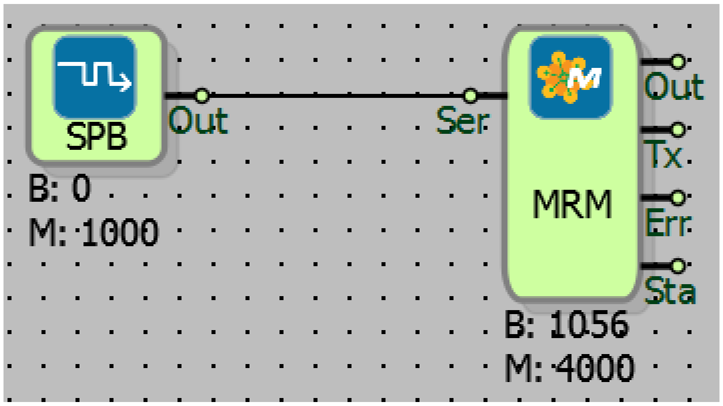
MP110 series has 1 serial port. So, Serial Port no must be 0.

MP211 series has 2 serial port. Serial Port No 0 is used for RS485 port, Serial Port No 1 is used for RS232.

Port Type

Port Type selection is only active for PLC hardware series that are capable of RS232/RS485 configurable serial port. If PLC doesn't have RS232/RS485 configurable serial port, selection of Port Type is ignored.

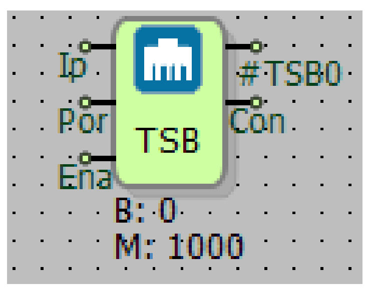
14.1.5 Sample Application



RTU Master block and serial port block is connected to use the device in Master mode in serial communication.

14.2 TCP SOCKET BLOCK

14.2.1 Connections

<table>
<tr><td>Ip: IP is entered here.</td><td rowspan="3"></td><td rowspan="2">#TSB0: Output of the block</td></tr>
<tr><td>Por: Port is entered here.</td></tr>
<tr><td>Ena: Enable pin.</td><td>Con: State of the connection</td></tr>
</table>

14.2.2 Connection Explanations

Ip: IP input

Server IP or Filter IP is entered here.

Por: Port input

Server or Client port is entered here.

Ena: Enable input

To activate TCP client socket, this input should be high(1).

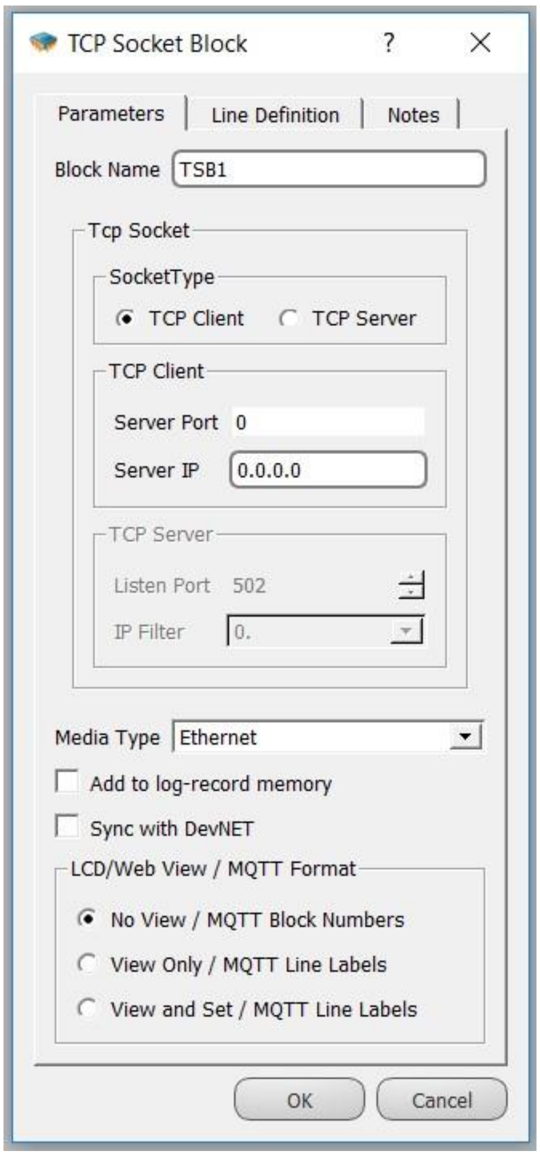
#TSB0: Output of the block

Block output which is connected to the protocol blocks which perform TCP communication.

Con: Connection

Indicates are there any established socket connection provided by the block. If there is active socket, it is high(1) and if there is no, it is low(0).

14.2.3 Block Settings

	Socket Type: One of the TCP Client or TCP Server options can be selected in Block Settings.
	Server Port: Client port input.
	Server IP: Client IP input.
	Listening Port: Server port input.
	IP Filter: IP filter input of the server.
	Media Type: Ethernet, GSM or WİFİ is chosen here.

14.2.4 Block Explanation

TCP Socket Block is used to provide the communications with Ethernet, GSM or Wi-Fi, with supported protocols.

“#TSB0” output of the block can be connected to the TCP Communication Protocol Blocks such as.

Modbus TCP Slave, Modbus TCP Master, DNP3 Slave, IEC101 Slave and IEC104 Slave.

“Con” output of the block is “1” when there exists a communication connection and “0” when there is no connection.

TCP Socket Block can be used as “TCP Client” or ”TCP Server”.

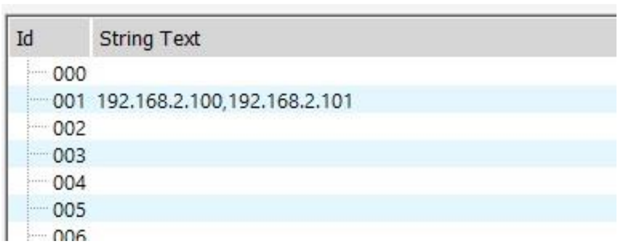
When you want to program the device as “TCP Client”, the “Server Port” and the “Server IP” of the TCP Server must be defined.

When the device is programmed as a "TCP Server", "Listen Port" that "TCP Client" would be connected must be defined.

When the device is programmed as a “TCP Server”, “TCP Client IP's that have connected can be filtered.

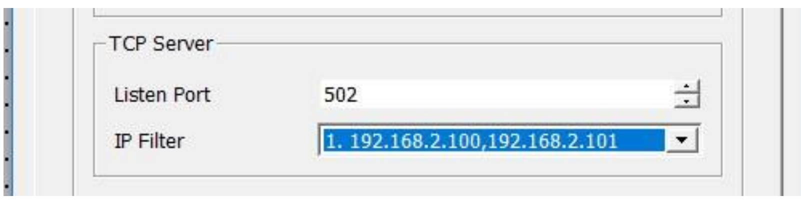
IP Filtresi

To select the IP filter from the TCP socket block, the IPs that will be allowed to connect to this device are first defined in the Projects / Text Table section. (Figure 1)



(1)

In the TCP socket blog, the allowed IPs index defined in the Text Table is selected from the IP Filtering option. (Figure 2)

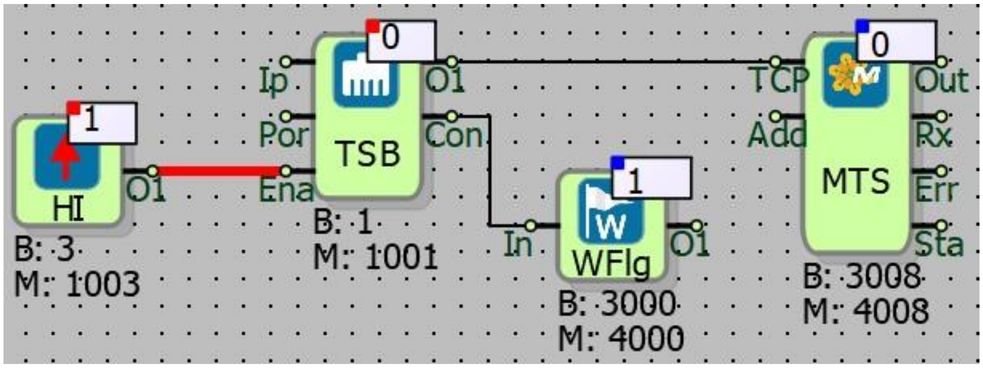


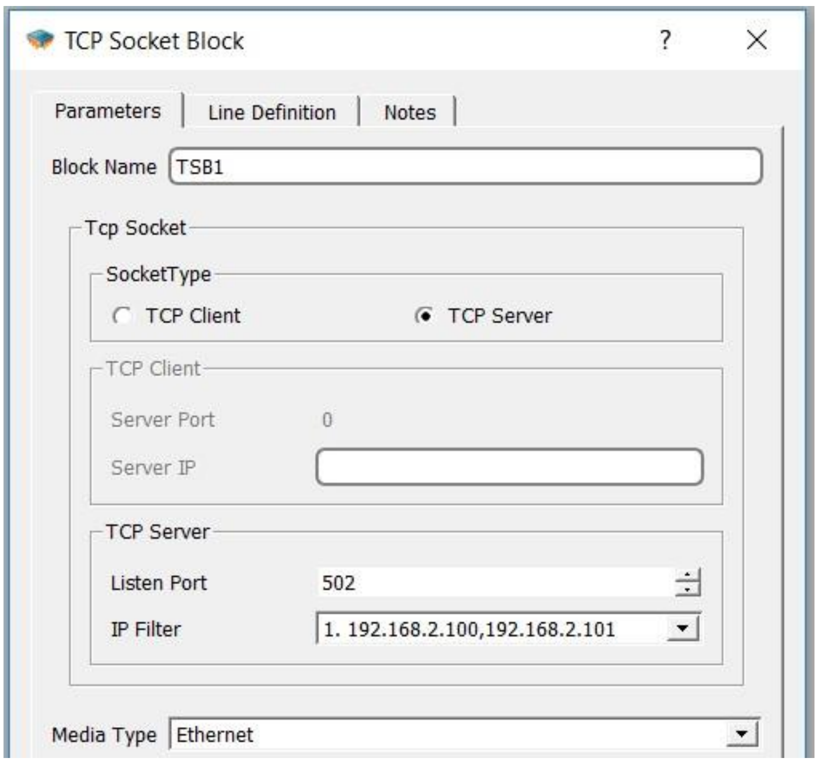
(2)

Thus, only filtered IP can connect to this device.

14.2.5 Sample Application

TCP Server Mode



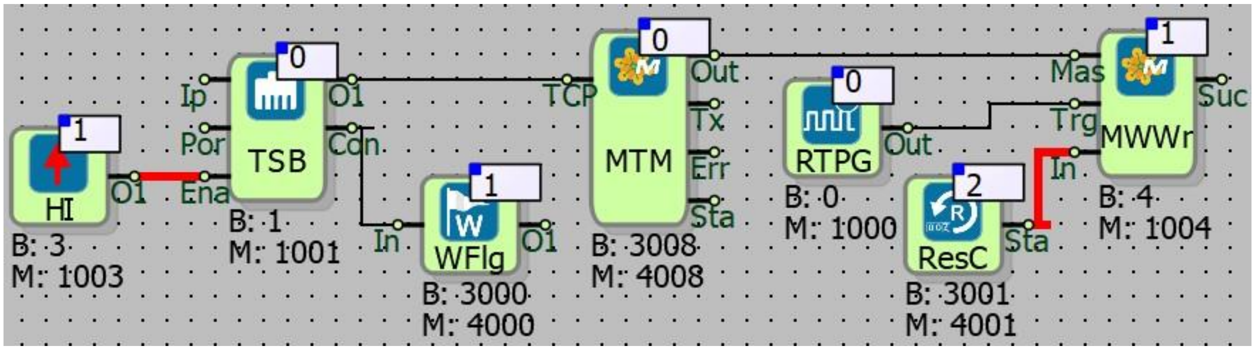


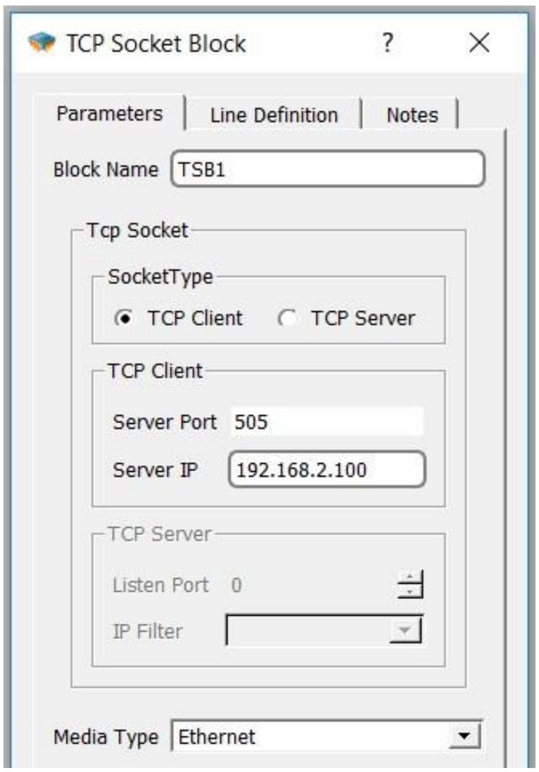
TCP Socket block is connected to the Modbus TCP slave block, so Modbus communication will be performed.

For this, the TCP server is selected from the block options, the listening port (502) is also defined. IP filter is enabled and 2 different IPs are allowed to connect. (192.168.2.100 and 192.168.2.101)

In this case, the device can be connected to the Modbus TCP Client with one of the IPs in the IP filter.

TCP Client Mode





When TCP is programmed as Client, a TCP Master block must be connected to the block output. Modbus TCP Master block is connected in the example.

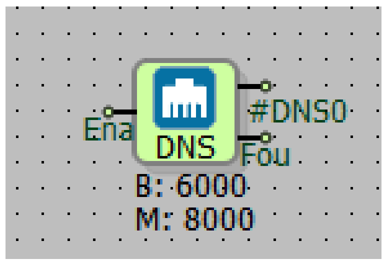
The Port of the Server to which the device will establish a communication connection is defined. The IP of the Server to which the device will establish a communication connection is defined.

After the IP and Port numbers are defined in the TCP Socket Block, the TCP Master block of the corresponding communication protocol (In Example Modbus TCP Master) is connected to the O1 output of TCP Socket Block.

Modbus Reader or Writer blocks that are connected to the Modbus TCP Master block output are also defined for addresses to be read or written.

14.3 DNS BLOCK

14.3.1 Connections

Ena: Enable pin		#DNS0: Output of the block Fou: Link status output

14.3.2 Connection Explanations

Ena: Enable pin

It is the input that needs to be given a logic (1) signal to activate the DNS block.

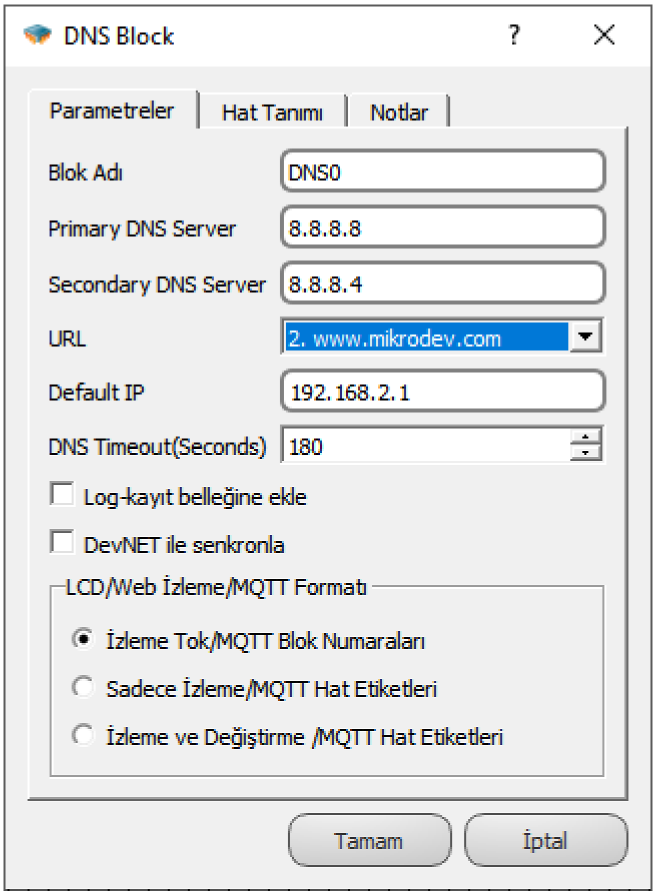
#DNS0: Output of the block

TCP Connects to the IP input of the Socket block.

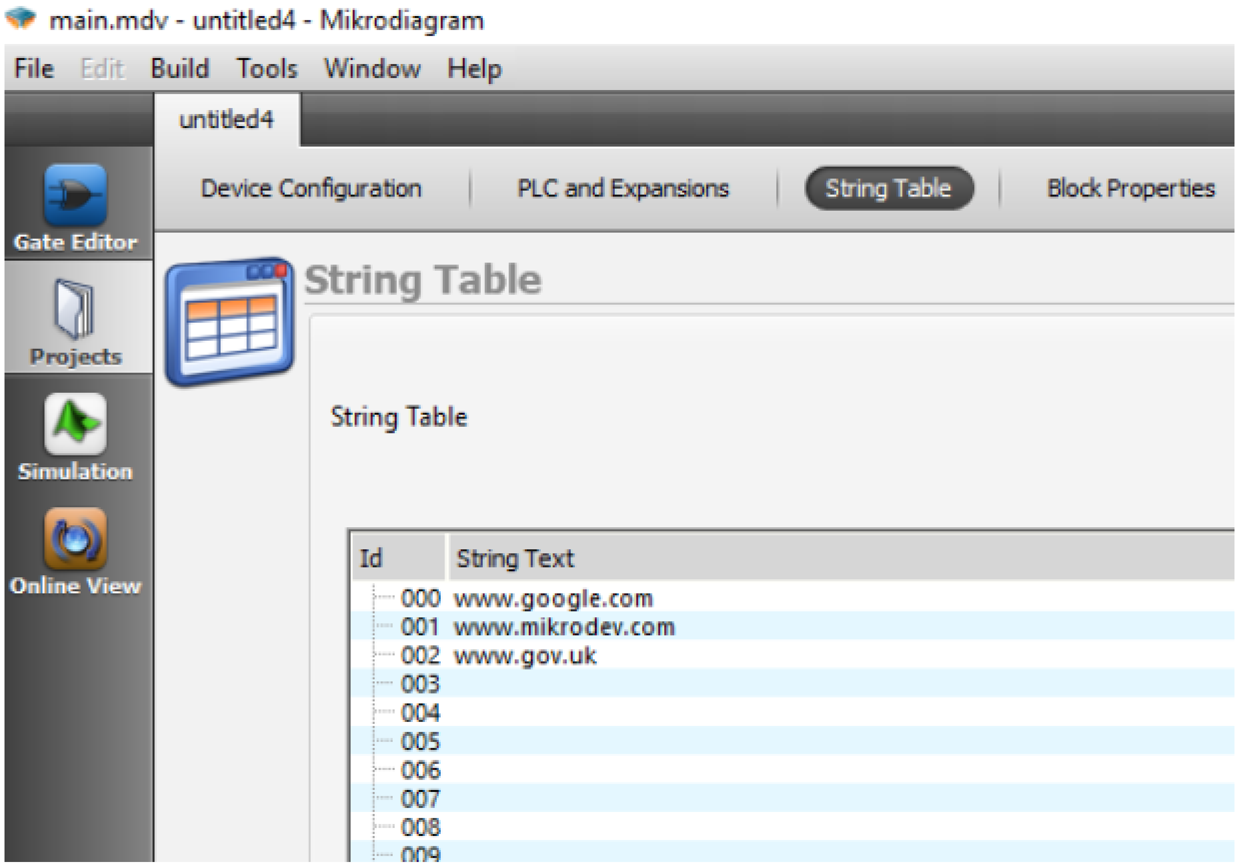
Fou: Link status output

It is the output that gives the logic 1 signal when the IP determined from the block special settings is found.

14.3.3 Block Settings

<table>
<tr><td rowspan="5">
</td><td>Primary DNS Server: The Primary DNS server is entered in this section.</td></tr>
<tr><td>Secondary DNS Server: Secondary DNS Server is entered to this section.</td></tr>
<tr><td>URL: The URL to use is entered in the string table. The ID of the URL entered in the string table is selected here.</td></tr>
<tr><td>Default IP: Default IP is entered in this section. If the DNS block cannot convert the URL to the IP address, Default IP is used.</td></tr>
<tr><td>DNS Timeout: The DNS timeout value is entered in this section</td></tr>
</table>

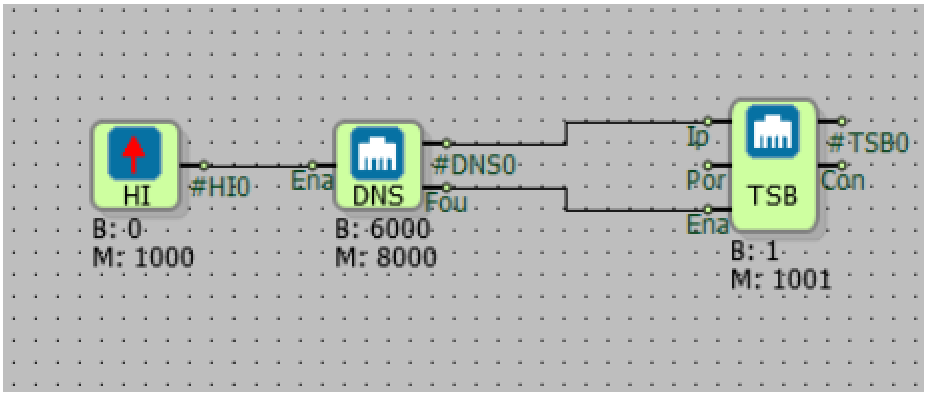
14.3.4 Block Explanations



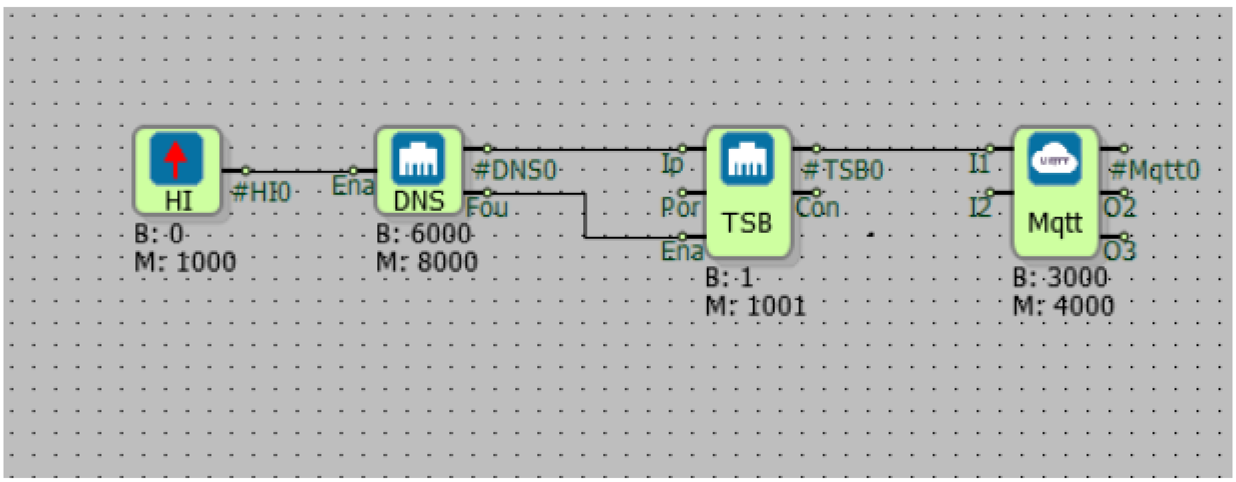
Click the Projects tab on the left of the Mikrodiagram software. String Table is selected from the top menu.

You can enter a URL under the String Text heading.

14.3.5 Sample Application



The DNS Block output is connected to the ip pin of the TCP Socket block. The DNS block will convert the URL to IP, Fou. pin is active. The TCP block uses the IP address from the DNS block.

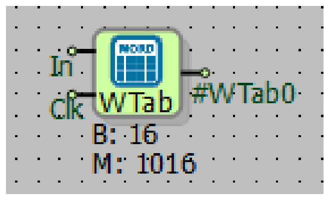


The TCP socket block can be used with the Mqtt block

15 TABLE BLOCKS

15.1 WORD TABLE

15.1.1 Connections

In: Word input value to add Clk: Clock signal input		#WTab0: Block output

15.1.2 Connection Explanations

In: Word value input to add

It is Word input value to add to the table.

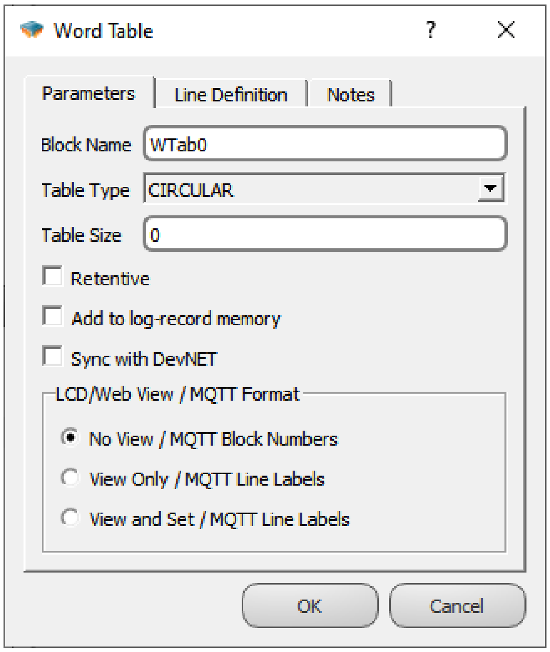
Clk: Clock signal input

When “Clk” signal is high, the data in the “In” input is added into the table.

#WTab0: Block output

The output block which carries the table reference.

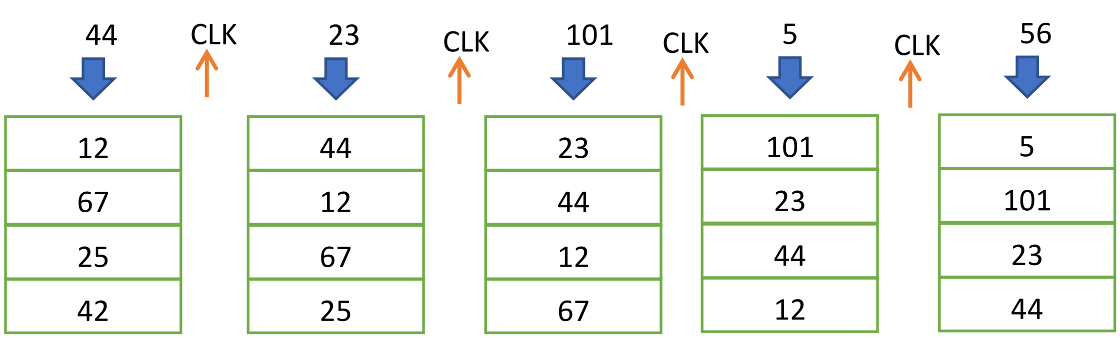
15.1.3 Block Settings

	Type Table: It can be selected as “CIRCULAR” or “FIFO”.
	Table Size: Table size can be determined with this option. The unit of the table size is “Byte”.

15.1.4 Block Explanation

Table size and table type can be chosen like below by user. Here, one of the options “Circular or FILO (First In Last Out)” must be chosen.

In FILO Mode; the data added into the table with smaller index is always added like a new one. While the new data becomes 0th data, the oldest one becomes the last element. For a table which has 4 word data, adding data in FILO mode works like below.



In applications where the order of addition of the data on the table is important, a FILO type table is required.

On large tables, adding data to FILO type table takes more processing time. Therefore, FILO type table should be used just if necessary.

For a table which has 4 word data, adding data in Circular mode works like below:

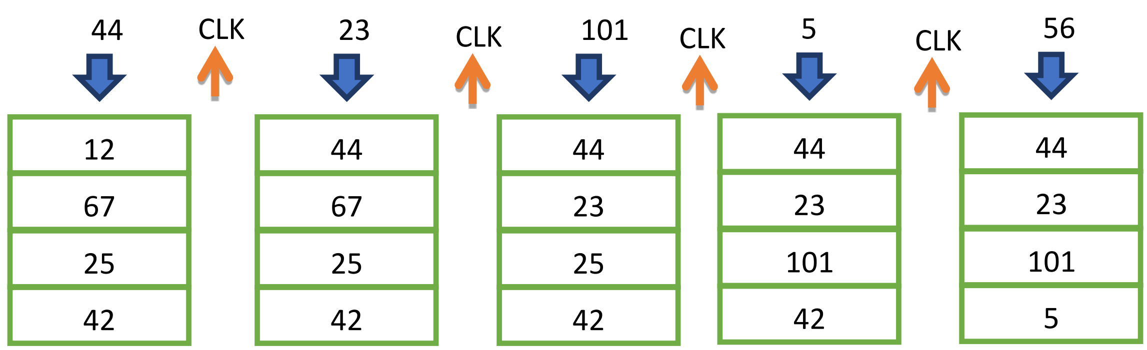
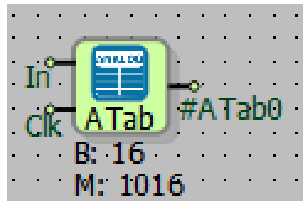


Table Size is the total byte size that the datas of the table cover in memory. Since the word datas are 2 bytes long, the size of the table should be 2 times the number of Word datas to be kept in the table.

Retentivity can be activated in the table blocks. After all PLC loops, the data in table blocks which retentivity is activated, recorded to the retentive memory of PLC. If PLC is somehow restarted then data in the table is read from the retentive memory and the initial values are filled. Thus, the data in the table becomes retentive. If it is also desired to record the order of data addition into the table, FILO must be selected as the table type. In the table which retentivity is activated an optimum table size must be selected to prevent wasting retentivity memory.

15.2 ANALOG TABLE

15.2.1 Connections

In: Analog input value to add Clk: Clock signal input		#ATab0: Block output

15.2.2 Connection Explanations

<u>In: Analog input value to add</u>

It is the analog input value to is added into the table.

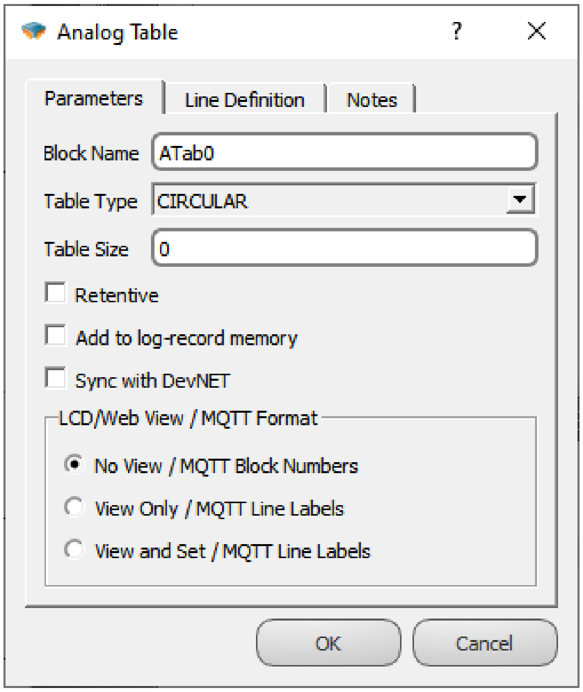
<u>Clk: Clock signal input</u>

In the rising edge of “Clk” signal, the data in the “In” input added to table.

<u>#ATab0: Block output</u>

The block output which is carry the table reference.

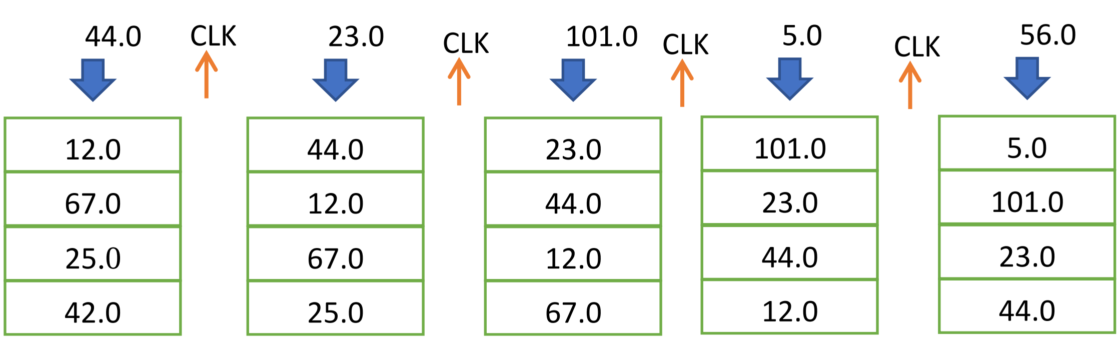
15.2.3 Block Setting

	Table Type: Table type can be determined as “CIRCULAR” or “FILO”.
	Table Size: The value of table size can be determined here. Its unit is “Bytes”.

15.2.4 Block Explanation

Table size and table type can be chosen like below by user. Here, “Circular or FILO (First In Last Out)” should be chosen.

In FILO Mode; the data with smaller index which is the data is added into the table always added like a new one. While the new data become 0th data, the oldest one become the last data. For a table which have 4 analog data, adding data in FILO mode works like below:



In applications where the order of addition of the data into the table is important, a FILO type table is required.

On large tables, adding data to FILO type table takes more processing time. Therefore, FILO type table should be used just if necessary.

For a table which have 4 analog data, adding data in Circular mode works like below:

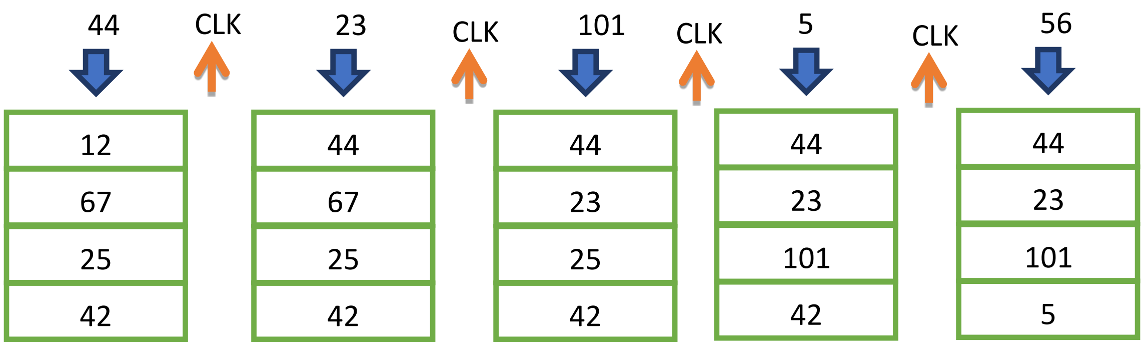
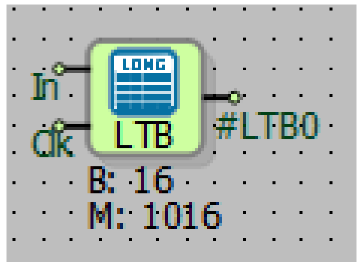


Table Size is the total byte size that the datas of the table cover in memory. Since the analog datas 4 bytes long, the size of the table should be 4 times the number of analog datas to be kept in the table.

Retentivity can be activated in the table blocks. After all PLC loops, the data in table blocks which retentivity is activated, recorded to the retentive memory of PLC. If PLC is somehow restarted then data in the table is read from the retentive memory and the initial values are filled. Thus, the data in the table becomes retentive. If it is also desired to record the order of data addition into the table, FILO must be selected as the table type. In the table which retentivity is activated an optimum table size must be selected to prevent wasting retentivity memory.

15.3 LONG TABLE

15.3.1 Connections

In: Long input value to add Clk: Clock signal input		#LTB0: Block output

15.3.2 Connection Explanations

In: Long input value to add

It's the long input value to be added to table.

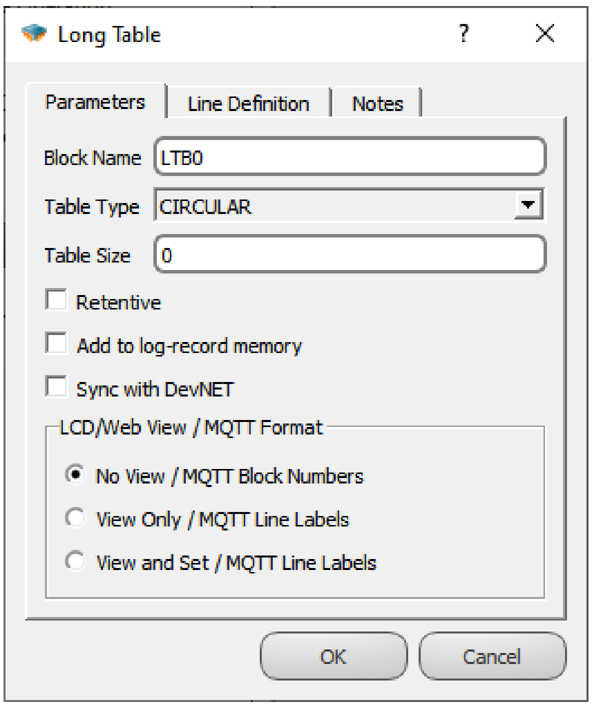
Clk: Clock signal input

In the rising edge of "Clk" signal, the data in the "In" input is added to table.

#LTB0: Block output

The block output which carries the table reference.

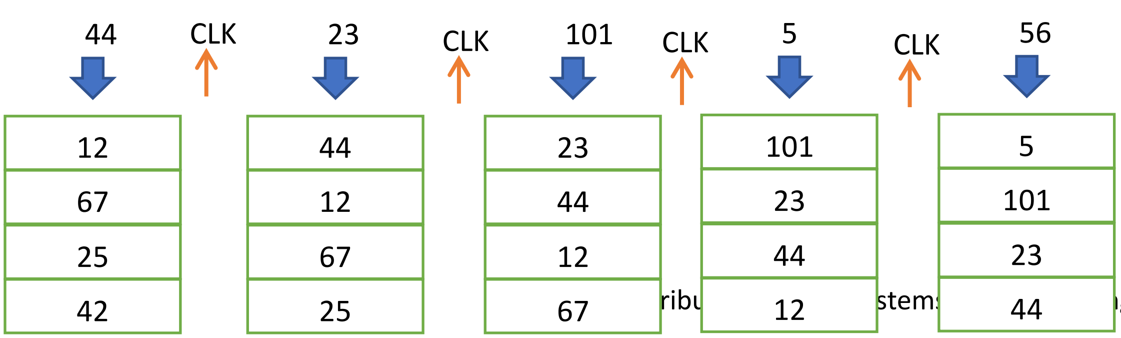
15.3.3 Block Setting

	Table Type: Table type can be determined as "CIRCULAR" or "FILO" in here.
	Table Size: The value of table size can be determined here. Its unit is "Bytes".

15.3.4 Block Explanation

Table size and table type can be chosen like below by user. Here, one of the options "Circular or FILO (First In Last Out)" should be chosen.

In FILO Mode; the data with smaller index which is the data is added into the table always add like a new one. While the new data become 0thh data, the oldest one become the last data. For a table which have 4 long data, adding data in FILO mode works like below:



In applications where the order of addition of the data on the table is important, a FILO type table is required.

On large tables, adding data to FILO type table takes more processing time. Therefore, FILO type table should used just if necessary.

For a table which have 4 long datas, adding data in Circular mode works like below:

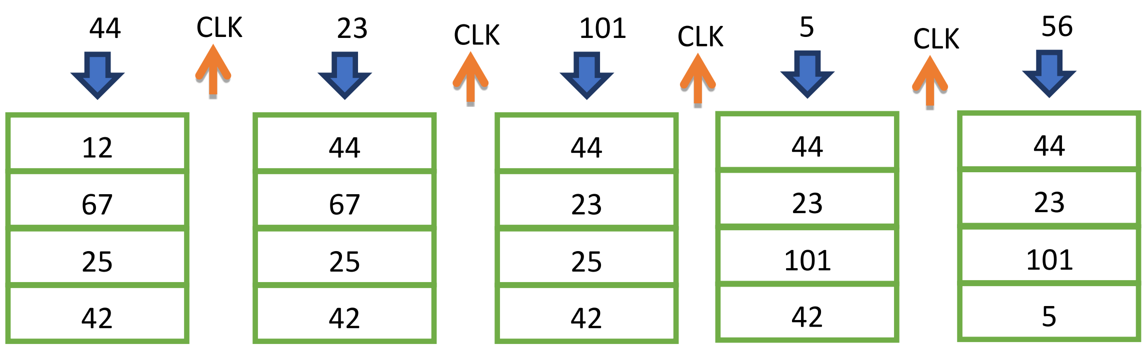
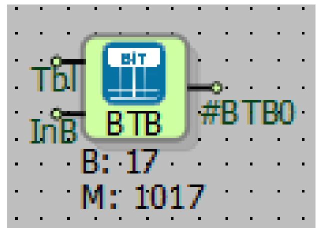


Table Size is the total byte area that the datas of the table cover in memory. Since the long datas 4 bytes long, the size of the table should be 4 times the number of long datas to be kept in the table.

Retentivity can be activated in the table blocks. After all PLC loops, the data in table blocks which retentivity is activated, recorded to the retentive memory of PLC. If PLC is somehow restarted then data in the table is read from the retentive memory and the initial values are filled. Thus, the data in the table becomes retentive. If it is also desired to record the order of data addition into the table, FILO must be selected as the table type. In the table which retentivity is activated an optimum table size must be selected to prevent wasting retentivity memory.

15.4 BIT TABLE

15.4.1 Connections

Tbl: Binary input value to add		#BTB0: Block output
InB: Clock signal input		

15.4.2 Connection Explanations

In: Binary input value to add

It is Binary input value to be added into table.

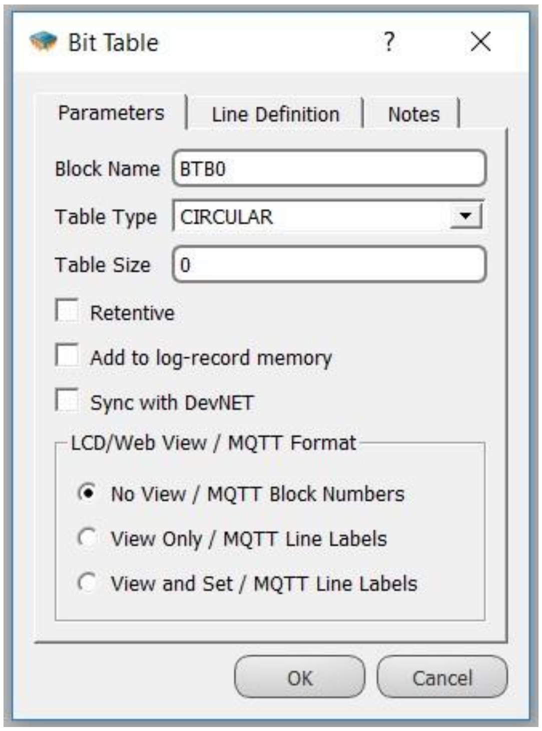
InB: Clock signal input

In the rising edge of “InB” signal, the data in the “Tbl” input is added into the table

#BTB0: Block output

The block output which is carry the table reference.

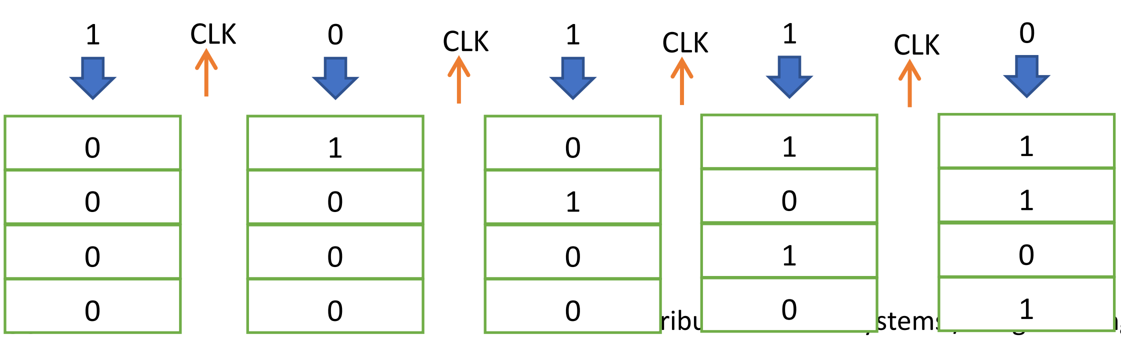
15.4.3 Block Setting

<table>
<tr>
<td rowspan="2">
</td>
<td>Table Type: Table type can be determined as “CIRCULAR” or “FILO” in here.</td>
</tr>
<tr>
<td>Table Size: The value of table size can be determined here. Its unit is Byte.</td>
</tr>
</table>

15.4.4 Block Explanation

Table size and table type can be chosen like below by user. Here, one of the options “Circular or FILO (First In Last Out)” should be chosen.

In FILO Mode; the data with smaller index which is the data is added into the table always add like a new one. While the new data become 0.data, the oldest one become the last data. For a table which have 4 bit data, adding data in FILO mode works like below:



In applications where the order of addition of the data on the table is important, a FILO type table is required.

On large tables, adding data to FILO type table takes more processing time. Therefore, FILO type table must used just it necessary.

For a table which have 4 bit data, adding data in Circular mode works like below:

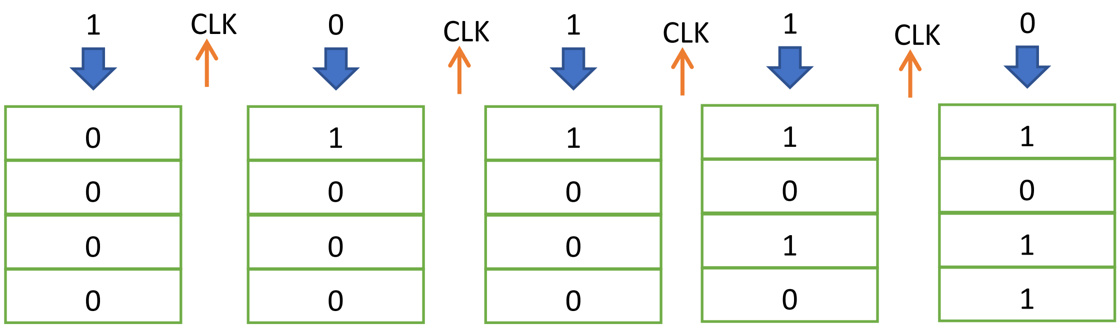
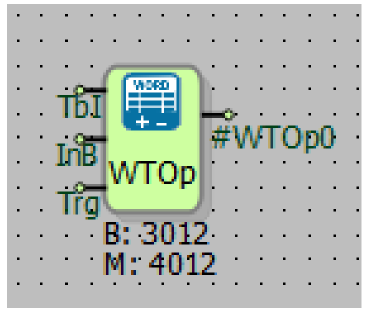


Table Size is the total byte area that the datas of the table cover in memory. Since the Bit datas are 1 byte long, the size of the table should be same as the number of Bit datas to be kept in the table.

Retentivity can be activated in the table blocks. After all PLC loops, the data in table blocks which retentivity is activated, recorded to the retentive memory of PLC. If PLC is somehow restarted then data in the table is read from the retentive memory and the initial values are filled. Thus, the data in the table becomes retentive. If it is also desired to record the order of data addition into the table, FILO must be selected as the table type. In the table which retentivity is activated an optimum table size must be selected to prevent wasting retentivity memory.

15.5 WORD TABLE OPERATION

15.5.1 Connections

TbI: Table reference connection InB: Parameter of operation Trg: Operation trigger signal		#WTOp0: Output of the block

15.5.2 Connection Explanations

TbI: Table reference connection

It's connected with the output of the table which is processed.

InB: Parameter of operation

It's the input parameter data used in some operations.

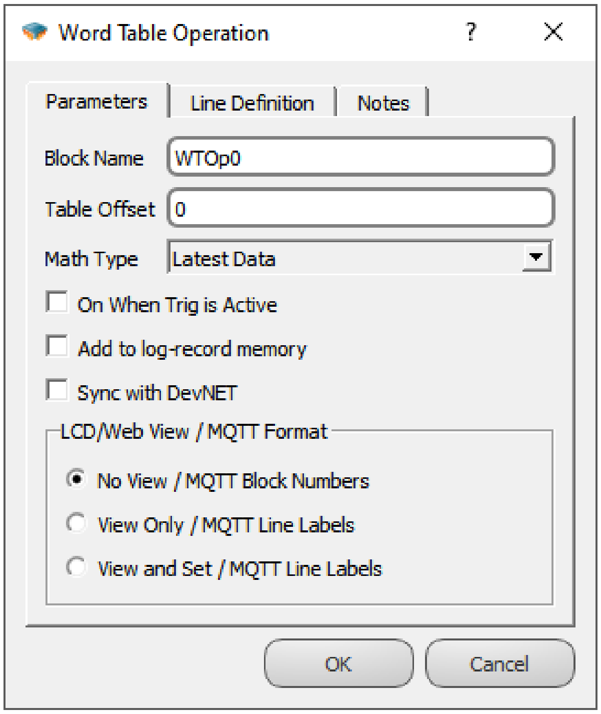
Trg: Operation trigger signal

It's the operation trigger input signal.

#WTOp0: Output of the block

The output for the result of table operation.

15.5.3 Block Settings

	Table Offset: It's used to select the data offset to be processed in the table data.
	Math Type: The operation type to be processed on the table data is selected.
	On When Trig is Active: If it is selected, the operation to be processed on the table data is executed only on the rising edge of the "Trg" input.

15.5.4 Block Explanation

It executes the operation which is defined on the table data and writes the result to output of the block.

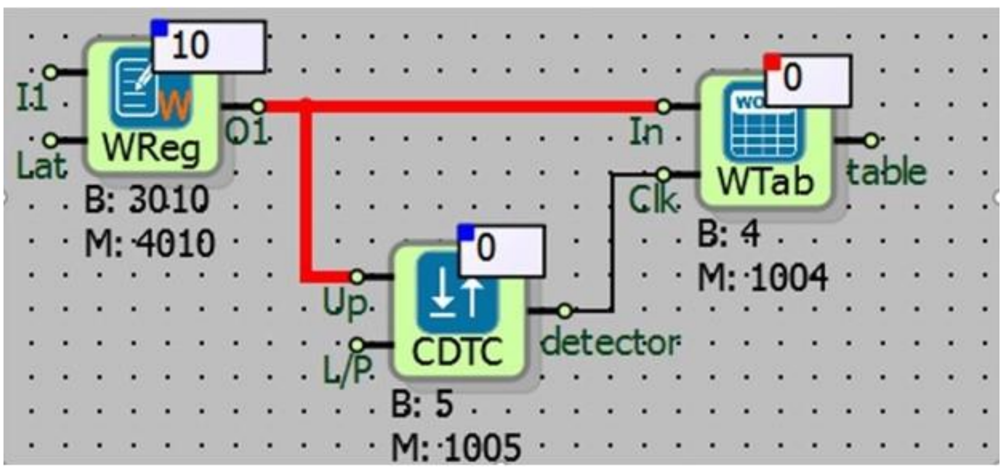
The types of operations that can be performed on the table and their explanations are as follows:

Latest Data	It fetches the data value which is the last value added to table.
Sum	It calculates the summary of all data on the table.
Mean	It calculates the average of the data on the table.
Max	It finds the maximum value on the table data.
Min	It finds the minimum value on the table data.
Median	The data on the table is ordered small to large, after that the value which is on the middle of order write to output of the block. If the number of values that can be written to the table is an even number, the arithmetic mean of the two middle values is written to the block output after small to large sorting.
Direction	It calculates increase or decrease on the trends from the data which is added to table then if it increases then write 1 or if it decreases then write 0 to output. Note: All table data must be filled in for the direction function to work.
Read Offset	It returns the value in the index which is defined with table offset, from the data on the table.
Read Byte Offset	Without looking the type of the value on the table, it returns the value in the offset when it is ordered as straight byte array.
Circular Left Shift	It shifts the data in the table left 1 index, and its transfer the leftmost data to right.
Shifting Left	İt shift the data in the table 1 index to left, write 0 to rightmost index.

Circular Right Shift	It shifts the data in the table right 1 index, and its transfer the rightmost data to left.
Shifting Right	İt shift the data in the table 1 index to right, write 0 to leftmost index.
Put Offset	The value in the InB entrance is written onto the data in the index which is defined by the table offset.
Clear Table	Resets the data in the table.
Search	The block output is written in which index of the table the value entered from the "InB" input among the table data is located.

Note: If the median is selected in the table operation, the values in the table indices are changed since the data in the table is sorted from small to large.

15.5.5 Sample Applications



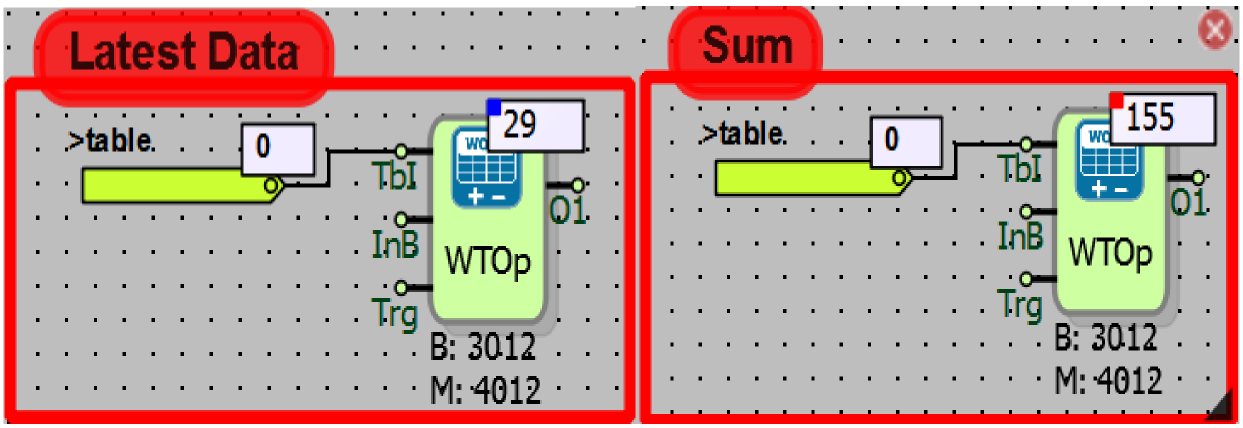
In the sample applications, the table type is selected as FILO and the table size is selected as 20 bytes, each word value is 2 bytes so 10 word values can be recorded in the table. Each time the value in the change detector block and Word table “In” input change, it is written on the table.

In the example,8 random integer is written on the table.

29	16	5	20	18	33	10	24	0	0

“Tbl” input of word table operation blocks is connected with the output of word table

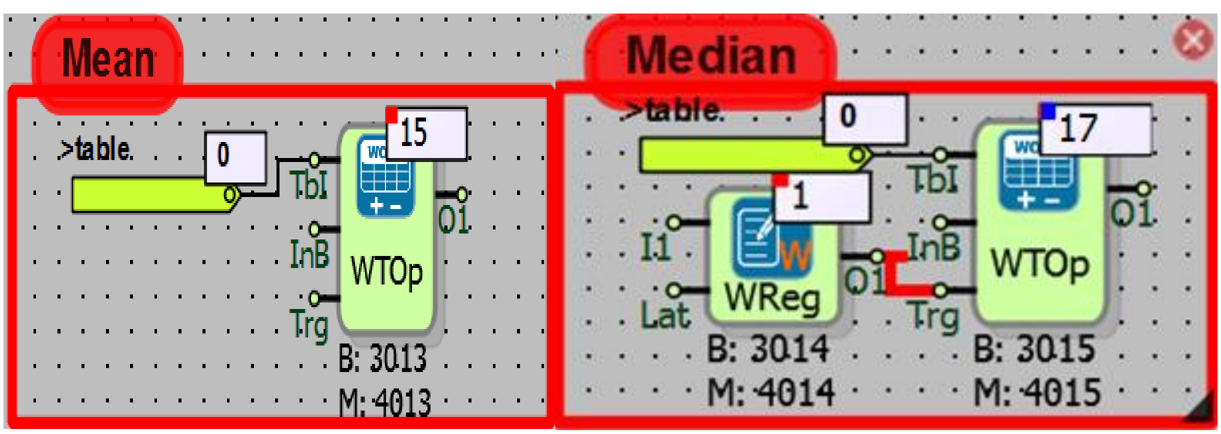
blocks. When “**LatestData and Sum** “is selected in the word table operation block:



When **“LatestData”** is selected: Since last value added to the table is 29, the value is written on the output of block.

When **“Sum”** is selected: integers written on the table are collected and summery is written in the output of the block.

When “**Mean and Median** “are selected in the word table operation block;



In the mean operation, the values in the table are summed and divided by 10 because the table size is selected by 10 word values. (155/10=15 decimal part is filtered because it is word table operation block.)

There are 10 word value (even number) on the table in the median operation.

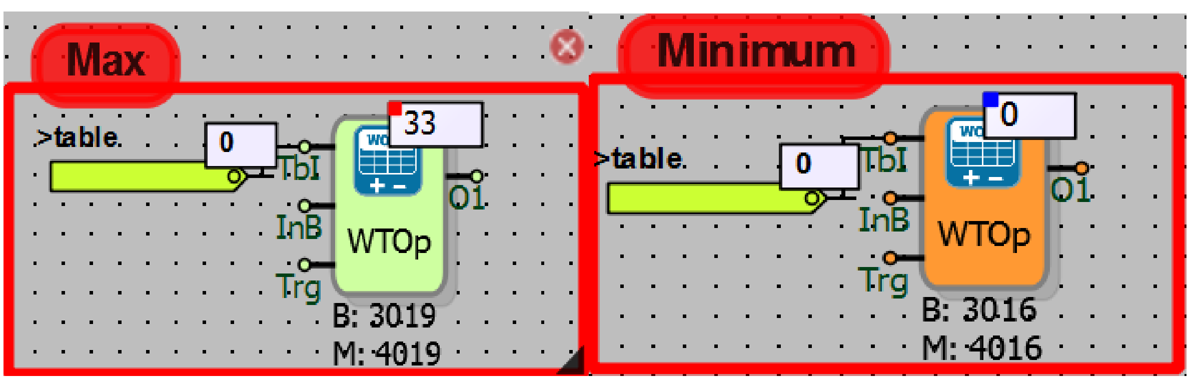
With median operation, the values on the table are ordered as small to large.

0	0	5	10	16	18	20	24	29	33

The arithmetic mean of the values at the 4th and 5th offset (16 and 18) of the table, which is sorted from small to large, is taken and written at the output of the block.

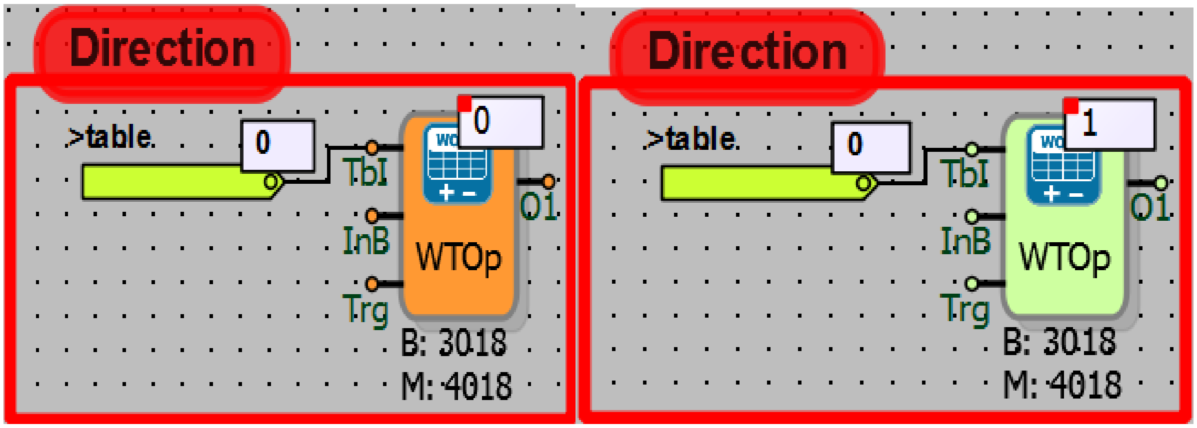
Note: In the median operation, the values in the table are reordered from the smallest to the table, and the value in the middle of the table is written to the block output as the median value. If there are an even number of values in the table (for example 10 values), the arithmetic mean of the two values in the middle of the table is written to the block output as the median value.

When "**Max and Min**" are selected in the word table operation block;



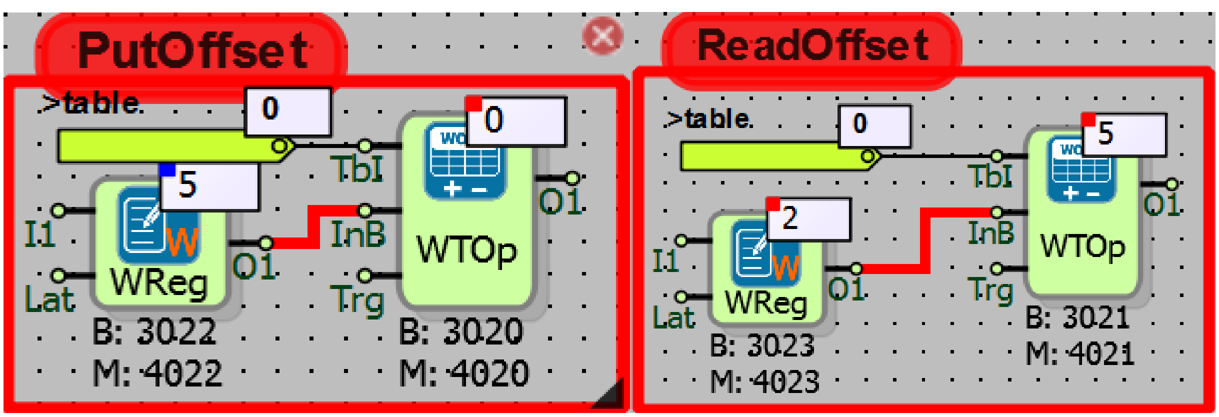
The largest integer written on the table is 33 so maximum value is 33; the smallest integer written on the table is 0 so minimum value is 0.

When “**Direction**” is selected in the word table operation block;



When the direction operation is selected, the last value added to the table is compared with the previous value from the last. İf last value is bigger, than 1 is written on the output of block otherwise it will be 0.

When “**PutOffset and ReadOffset**” are selected in the word table operation block;

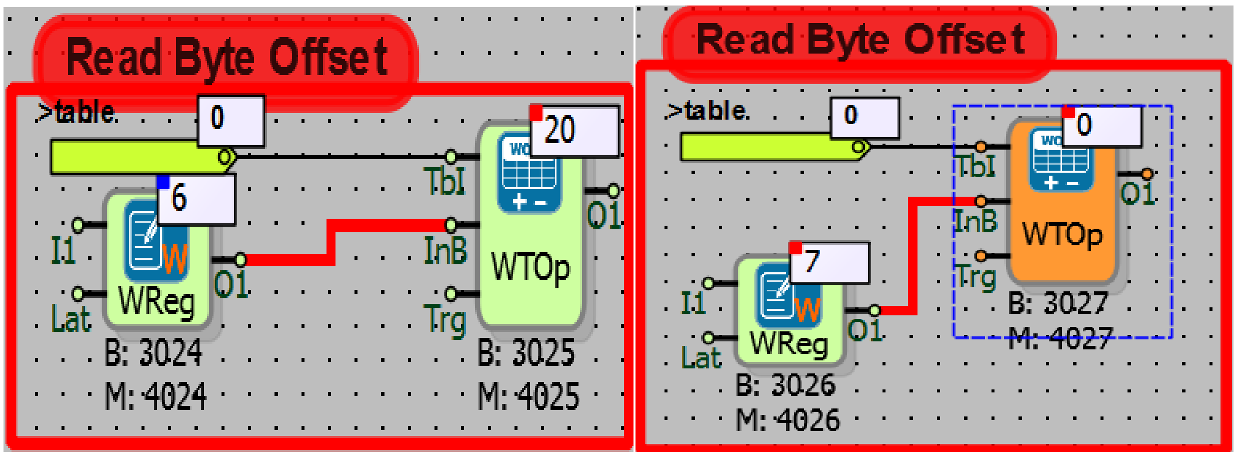


Put Offset: “Table offset” is selected as “2” from inside of the word table operation block. In this case, the value in the input of “In” will be written to the 2nd offset of the table.

Read Offset: The table offset to be read in the word table operation block can be selected from inside and outside of the block. In the example, table offset is selected as “2” from outside of the block.

In this case the value which is written on the 2. offset of the table by Put Offset is read on the 2. offset of the table by Reading Offset.

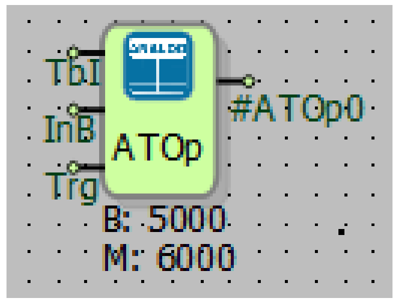
When “**ReadByteOffset**” is selected in the word table operation block;



In the example, the 6th and 7th bytes of the 20 byte long word table are read. The 6th and 7th bytes correspond to the 3rd table offset in the table. In this case, the 6th bit indicates the LSB (least significant bit) bits and the 7th bit indicates the MSB (most significant bit) bits. “20” value at the third table offset are written in LSB bits that can carry 0-255 values. Since the value at the third table offset is less than 256, the MSB bits are 0.

15.6 ANALOG TABLE OPERATION

15.6.1 Connections

Tbl: Table reference connection		#ATOp0: Output of the block
InB: Operation parameter		
Trg: Operation trigger signal		

15.6.2 Connection Explanations

Tbl: Table reference connection

The output of the table to be processed is connected.

InB: Operation parameter

It is the input of the parameter data which is used in some operation.

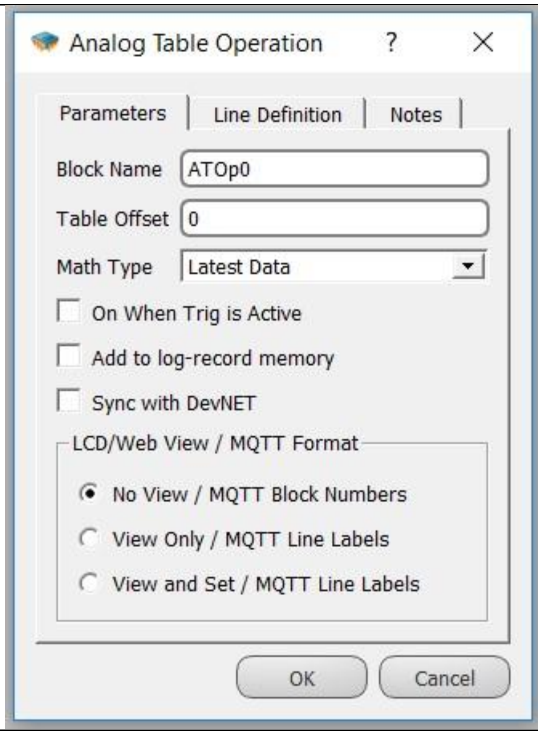
Trg: Operation trigger signal

Input of the operation trigger signal.

#ATOp0: Output of the block

The output of the table operation result.

15.6.3 Block Setting

<table>
<tr>
<td rowspan="3">
</td>
<td>Table Offset: It’s used to select the data offset to be processed in the table data.</td>
</tr>
<tr>
<td>Math Type: The operation type to be processed on the table data is selected.</td>
</tr>
<tr>
<td>On When Trig is Active: If it is selected, the operation to be processed on the table data is executed only on the rising edge of the "Trg" input.</td>
</tr>
</table>

15.6.4 Block Explanation

It executes the operation which is defined on the table data and write the solution to output of the block.

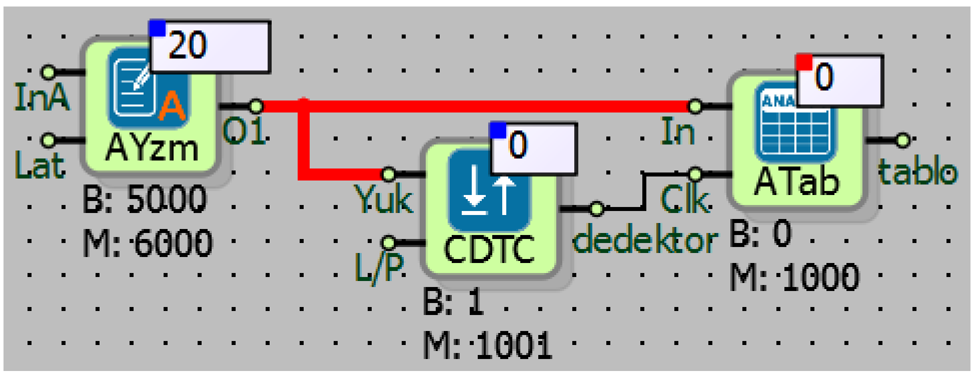
The types of operations that can be performed on the table and their explanations are as follows:

Latest Data	It fetches the data value which is the last value added to table.
Sum	It calculates the summery of all data on the table.
Mean	It calculates the average of the data on the table.
Max	It finds the maximum value on the table data.
Min	It finds the minimum value on the table data.
Median	The data on the table is ordered small to large, after that the value which is on the middle of order write to output of the block. If the number of values that can be written to the table is an even number, the arithmetic mean of the two middle values is written to the block output after small to large sorting.
Direction	It calculates increase or decrease on the trends from the data which is added to table then if it increases then write 1 or if it decreases then write 0 to output.
Read Offset	It returns the value in the index which is defined with table offset, from the data on the table.
Read Byte Offset	Without looking the type of the value on the table, it returns the value in the offset when it is ordered as straight byte array
Circular Left Shift	It shifts the data in the table left 1 index, and its transfer the leftmost data to right.
Shifting Left	İt shift the data in the table 1 index to left, write 0 to rightmost index.
Circular Right Shift	It shifts the data in the table right 1 index, and its transfer the rightmost data to left.

Shifting Right	İt shift the data in the table 1 index to right, write 0 to leftmost index.
Put Offset	The value in the InB entrance is written onto the data in the index which is defined by the table offset.
Clear Table	Resets the data in the table.
Search	The block output is written in which index of the table the value entered from the "InB" input among the table data is located.

Note: If the median is selected in the table operation, the values in the table indices are changed since the data in the table is sorted from small to large.

15.6.5 Sample Applications

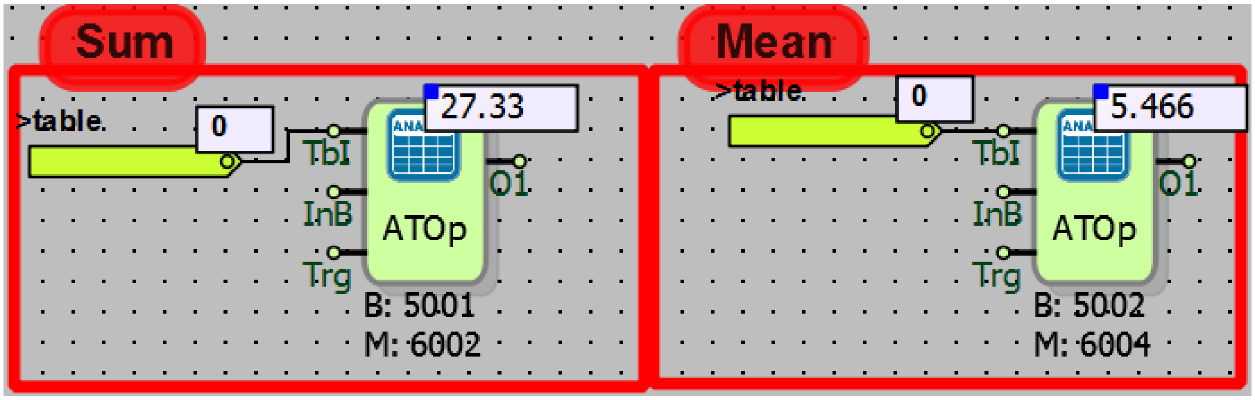


In the sample applications, the table type is selected as Circular and the table size is selected as 20 bytes, each analog value is 4 bytes so 5 word values can be recorded in the table. Each time the value in the change detector block and Analog table “In” input change, it is written on the table.

In the example, 3 analog values are randomly written in the table.

10,45	-4,12	21	0	0

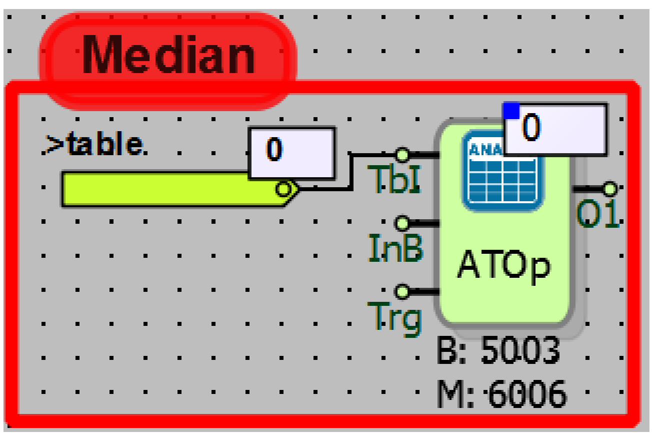
When "**Sum and Mean**" is selected in the analog table operation block;



When “Sum” is selected; The analog numbers written in the table are summed and the total value is written to the block output.

When “Mean” is selected; the values in the table are summed and divided by 5 because the table size is selected according to the 5 analog values. (27.33/5=5.466)

While "**Median"** is selected in the analog table operation block;



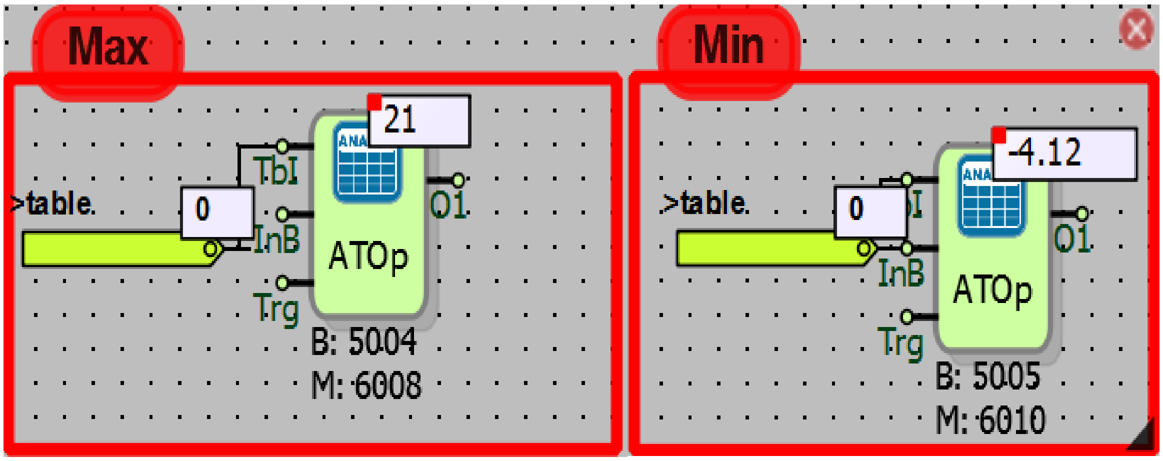
Median operation has 5 analog values on table.

With median operation, values in the table are sorted from small to large.

-4,12	**0**	**0**	**10,45**	**21**

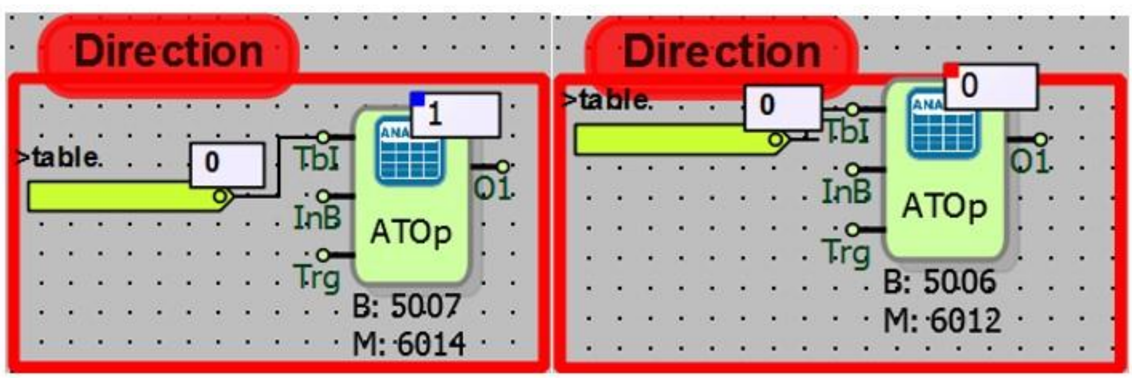
The value at the middle point of the table (0 value in the 2nd offset) is written to the block output.

While the analog table operation block is selected as "**Max and Min**";



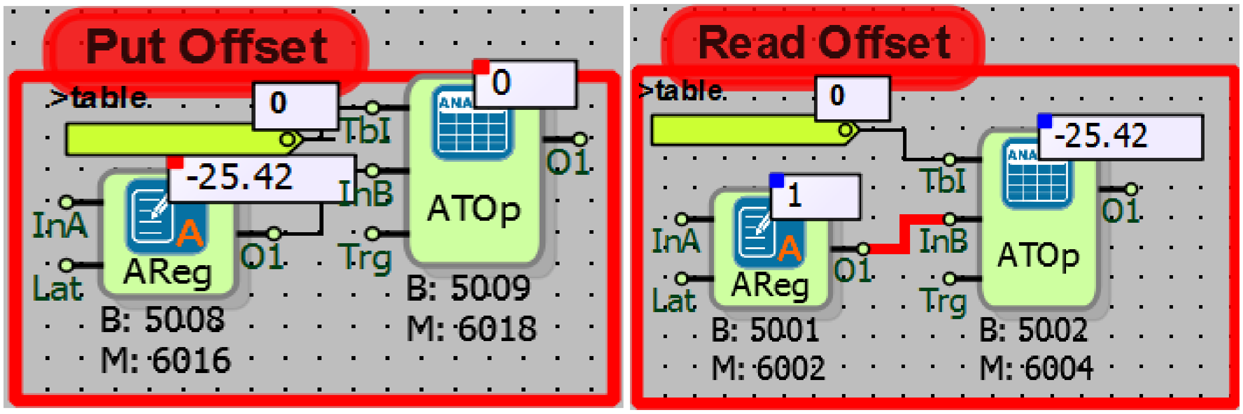
The maximum value written to the table is 21, the maximum value is 21, and the smallest integer in the table is -4.12, the minimum value is -4.12.

While "**Direction**" is selected in analog table operation block



When the direction operation is selected, the last value added to the table is compared with the previous value from the last. If the last value is greater, "1" is written to the block output. If the last value is smaller, "0" is written to the block output.

In the analog table operation block, "**PutOffset and ReadOffset**" are selected;



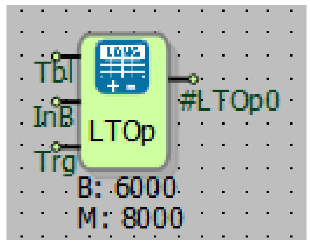
Put Offset: "Table offset" has been selected as 1 in analogue table operation block. In this case the value in InB will be written to the 1st offset of the table.

Read Offset: The table to be read in the analog table operation block can be selected from inside and outside the offset block. In the example, the offset is chosen as 1 from out of the block.

In this case, the value written to the 1st offset of the table with Put Offset is read from the 1st offset of the table with Read Offset.

15.7 LONG TABLE OPERATİON

15.7.1 Connections

TbI: Table reference connection		#LTOp0: Output of the block
InB: Operation Parameter		
Trg: Operation trigger signal		

15.7.2 Connection Explanations

TbI: Table reference connection

The output of the table to be processed is connected.

InB: Operation parameter

The parameter data input used in some operations.

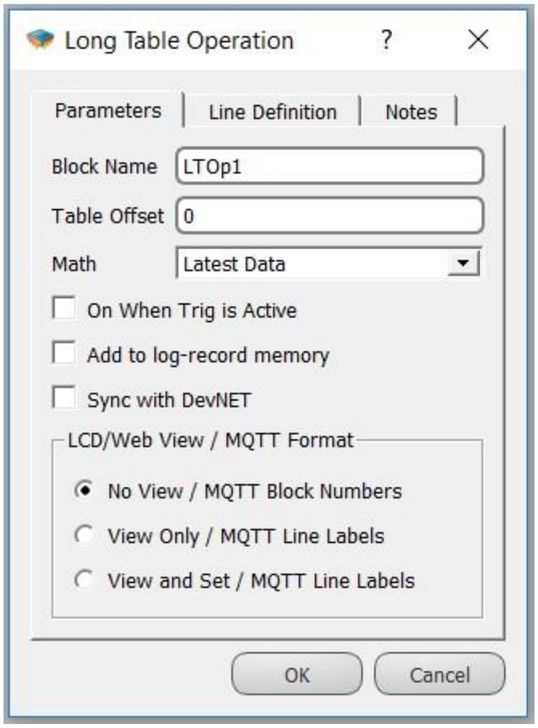
Trg: Operation trigger signal

Operation trigger signal input.

#LTOp0: Output of the block

Output of the result of table operation.

15.7.3 Block Settings

	Table Offset: It is used in the table data to select the data offset to be processed.
	Math Type: The operation type to be performed on the table data is selected.
	On When Trig is Active: If it is selected, the operation to be performed on the table data is executed only on the rising edge of the "Trg" input.

15.7.4 Block Explanations

It writes the result of the operation to the output of the block by performing the operations defined on the table data.

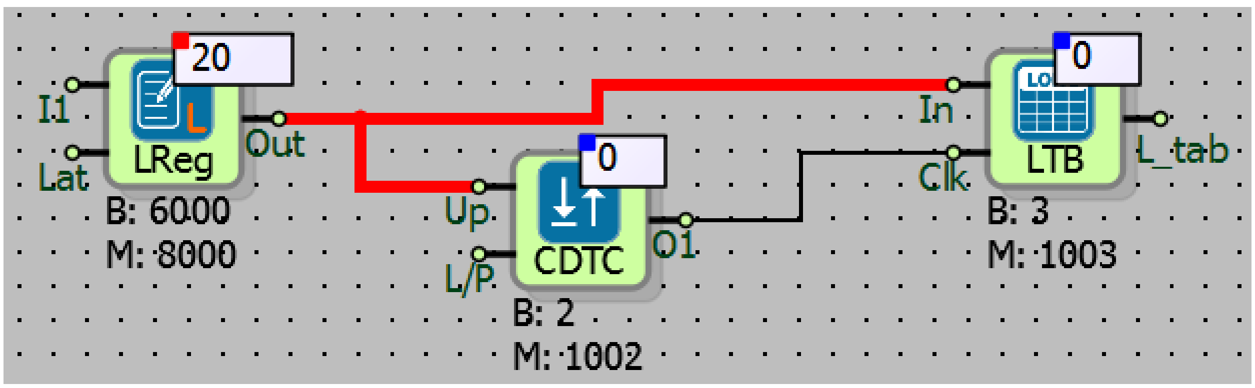
The types of operations that can be performed on the table and their explanations are as follows:

Latest Data	Returns the most recently added data value to the table
Sum	The table calculates the sum of all the data.
Mean	It calculates the average of the data in the table.
Max	İt finds the greatest value from the table data.
Min	İt finds the smallest value from the table data.
Median	The data in the table is sorted from small to large, the value in the middle of the table is written to the block exit after sorting. If the number of values that can be written to the table is an even number, the arithmetic mean of the two middle values is written to the block output after small to large sorting.
Direction	It calculates increase or decrease on the trends from the data which is added to table then if it increases then write 1 or if it decreases then write 0 to output.
Reading Offset	Returns the value of the indexed value defined by the table offset from the table data.
Read Byte Offset	Regardless of the type of data in the table, the value in the offset at which it is ordered as a straight byte array is returned.
Circular Left Shift	Shifts the data in the table to the left by 1 index and moves the leftmost indexed data to the far right.
Shifting Left	The table data is shifted left by 1 index and 0 is written to the rightmost.
Circular Right Shift	Move the table data to the right by 1 index and move the rightmost indexed data to the left.

Shifting Right	Move the table data 1 index right and write 0 to the leftmost value
Put Offset	The value in the input “In” is written on top of the indexed data defined by the table offset.
Clear Table	Resets the data in the table.
Search	The block output is written in which index of the table the value entered from the "InB" input among the table data is located.

Note: If the median is selected in the table operation, the values in the table indexes change because the table data is sorted from small to large.

15.7.5 Sample Applications

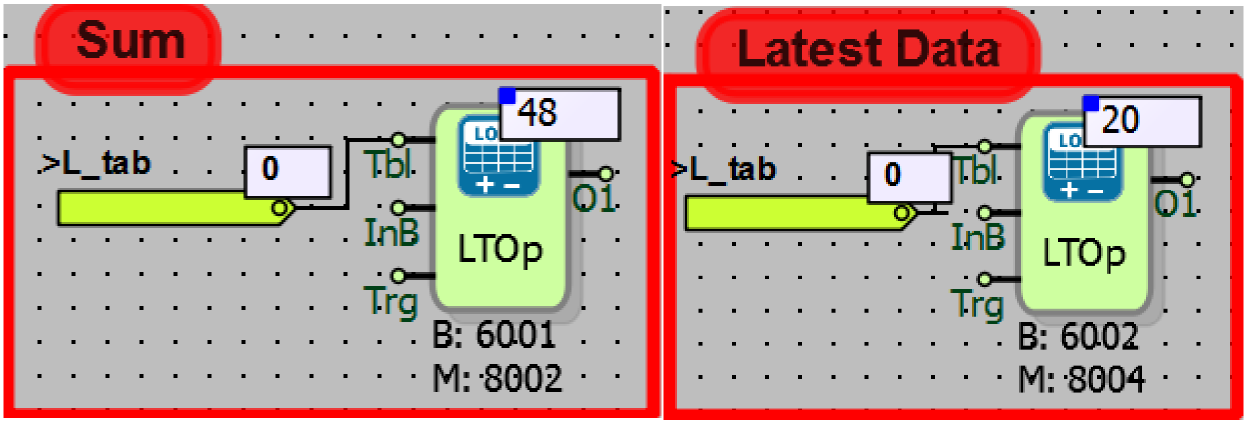


In the example applications, the table type is selected as “Circular”, the table size is selected as 20 bytes, and 1 long value is 4 bytes, 5 long value tables can be saved. The change detector block and the value of the input in the long table “In” are changed each time the value changes.

In the example, 5 random values are written randomly in the table.

52	-32	12	-4	20

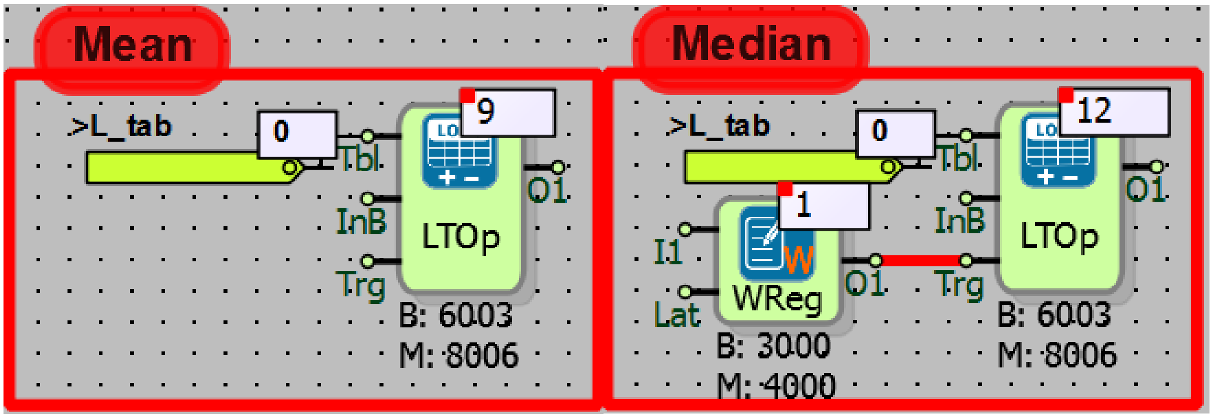
While "**LatestData and Sum**" is selected in the long table operation block;



When **"Latest Data"** is selected; Since the last 20 values are stored in the table, the value is written to the block output.

When **"Sum"** is selected; the numbers written in the table are summed and the total value is written at the output of the block.

When "**Mean and Median**" is selected in the long table operation block;



While "Mean" is selected; the values in the table are summed and divided by 5 because the table size is selected according to 5 long values (Since the 48/5 = 9 long operation is performed, the decimal part of the operation result is filtered.)

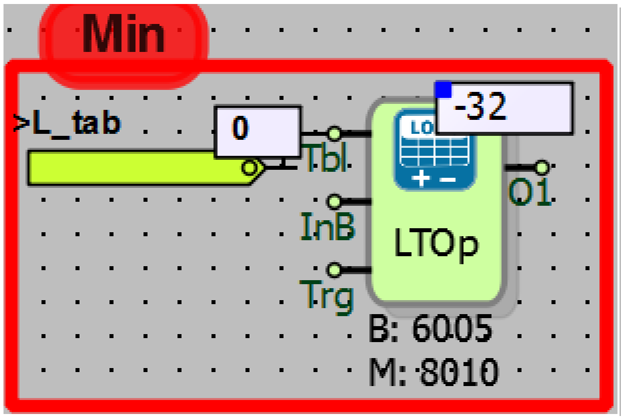
Median process has 5 long value on table.

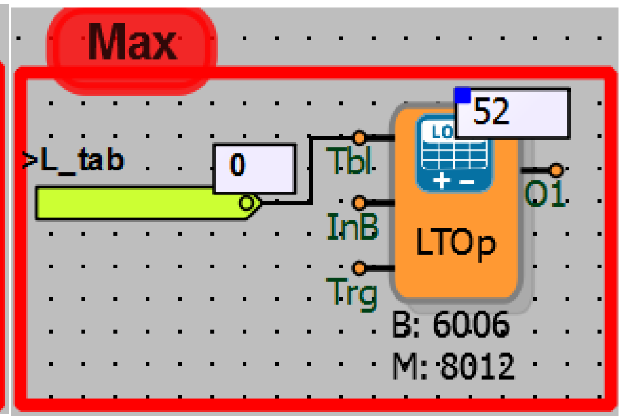
With median operation, values in the table are sorted from small to large.

-32	-4	12	20	52

The value at the middle point of the table (12 values in the 2nd offset) is written to the block output.

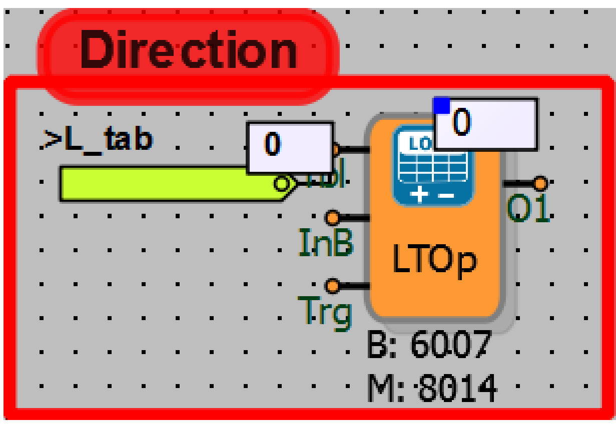
When "**Max and Min**" is selected in the long table operation block;

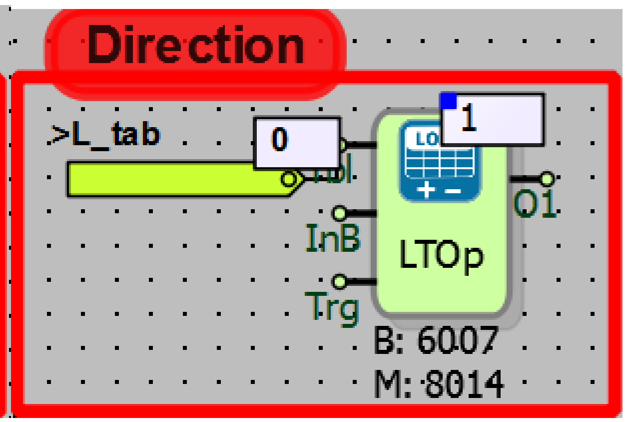




The maximum value written to the table is 52, the maximum value is 52, and the smallest integer in the table is -32, the minimum value is -32.

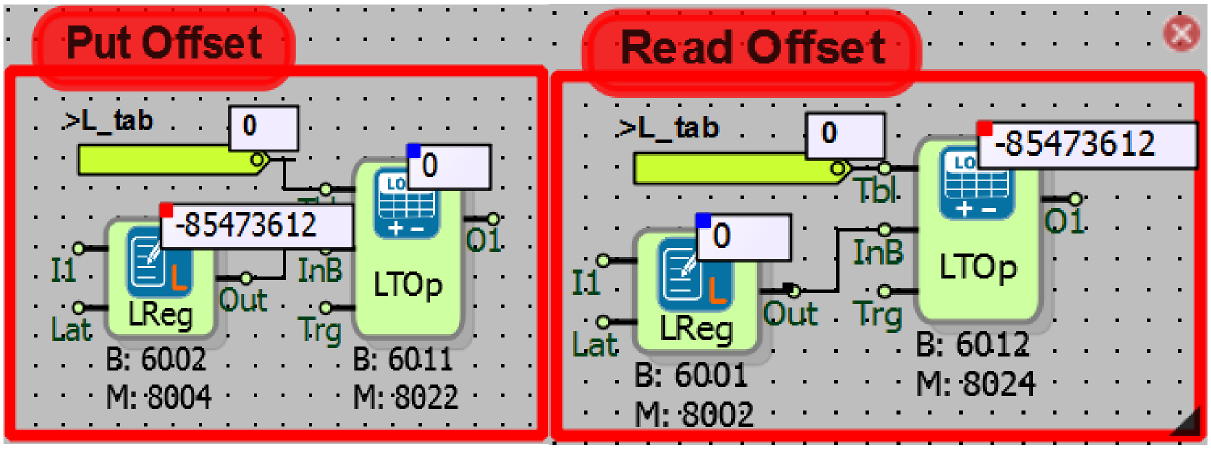
While "**Direction**" is selected in the long table operation block;





When the direction operation is selected, the last value added to the table is compared with the previous value from the last. If the last value is greater, "1" is written to the block output. If the last value is smaller, "0" is written to the block output.

While “**PutOffset and ReadOffset**” are selected in the long table operation block;

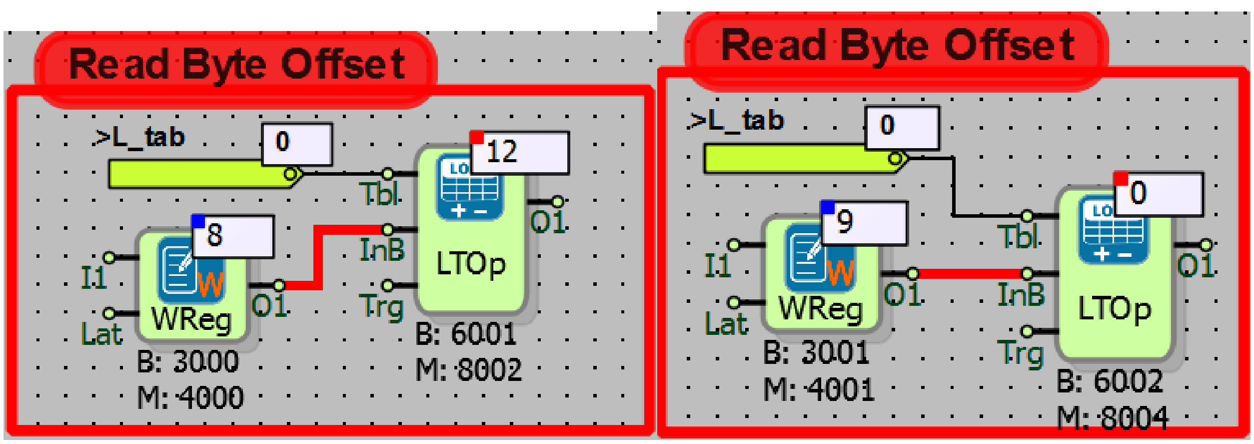


Put Offset: The "table offset” is selected as 0 from the long table operation block. In this case, the value in “InB” will be written to the 0th offset of the table.

Read Offset: The table to be read in the long table operation block can be selected from inside and outside the offset block. In the example, the offset is chosen as 0 from out of the block.

In this case, the value written to 0th offset of the table with Put Offset is read from 0th offset of the table with Read Offset.

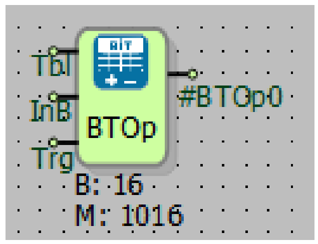
When "**ReadByteOffset**" is selected in the long table operation block;



In the example, the 8th and 9th bytes of the long table which is 20 bytes long are read. 8th, 9th, 10th, 11th byte corresponds to the 2nd table offset in the table. In this case, the 8th, 9th bits indicate the LSB bits, and the 10th, 11th bits indicate the MSB bits. “12” value at the 2nd table offset are written to the 8th byte which can carry 0-255 values. Since the value is less than 256 9th, 10th, 11th bytes are all 0.

15.8 BIT TABLE OPERATİON

15.8.1 Connections

TbI: Table reference connection		#BTOp0: Output of the block
InB: Operation parameter		
Trg: Operation trigger signal		

15.8.2 Connection Explanations

TbI: Table reference connection

The output of the table to be processed is connected.

InB: Operation parameter

The parameter data input used in some operations.

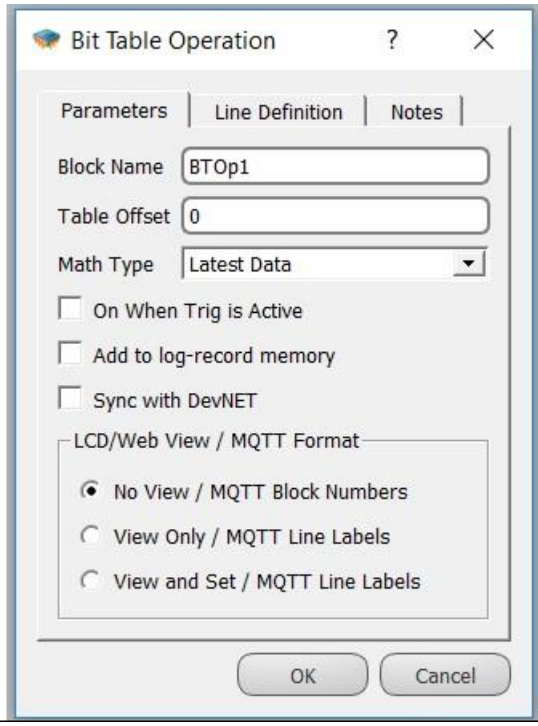
Trg: Operation trigger signal

Operation trigger signal input

#BTOp0: Output of the block

Output for the result of table operation.

15.8.3 Block Setting

	Table Offset: It is used in the table data to select the data offset to be processed.
	Math Type: The operation type to be performed on the table data is selected.
	On When Trig is Active: If it is selected, the operation to be performed on the table data is executed only on the rising edge of the "Trg" input.

15.8.4 Block Explanations

It writes the result of the operation to the output of the block by performing the operations defined on the table data.

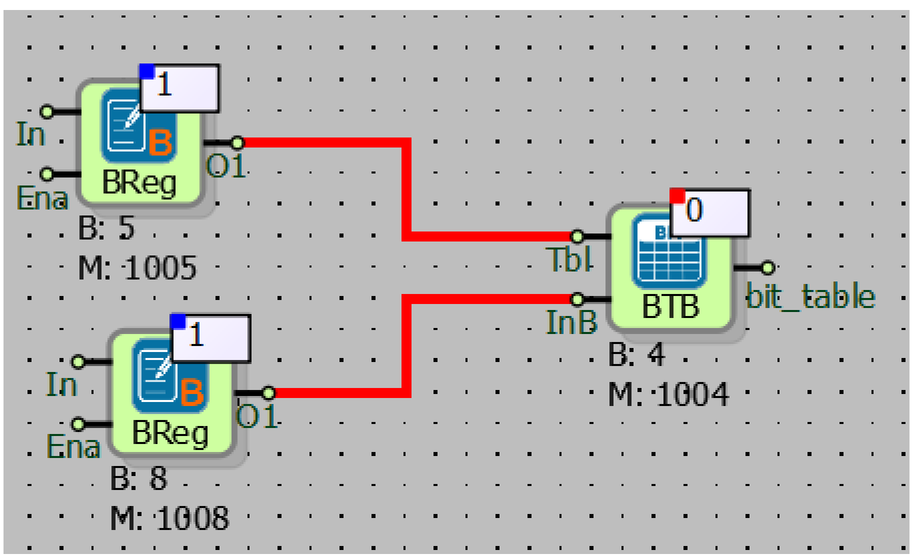
The types of operations that can be performed on the table and their explanations are as follows:

Latest Data	Returns the most recently added data value to the table.
Sum	If any of the data in the table is 1, the result is 1, if all 0, the result is 0.
Mean	If any of the data in the table is 0, it is 0, and if all of them are 1, the result is 1.
Max	If any of the data in the table is 1, the result is 1, if all 0, the result is 0.
Min	If any of the data in the table is 0, the result is 0, all 1 are the result 1.
Median	The data in the table is sorted from small to large, the value in the middle of the table is written to the block exit after sorting. If the number of bit values that can be written to the table is an even number, then if the middle two values are 1 after the sorting process, 1 is written to the block output. If either or both of the middle values are 0, 0 is written to the block output.
Direction	It calculates increase or decrease on the trends from the data which is added to table then if it increases then write 1 or if it decreases then write 0 to output.
Reading Offset	Returns the value of the indexed value defined by the table offset from the table data.
Read Byte Offset	Regardless of the type of data in the table, the value in the offset at which it is ordered as a straight byte array is returned.
Circular Left Shift	It shifts the data in the table left 1 index, and its transfer the leftmost data to right.

Shifting Left	The table data is shifted left by 1 index and 0 is written to the rightmost.
Circular Right Shift	Move the table data to the right by 1 index and move the rightmost indexed data to the left.
Shifting Right	Move the table data 1 index right and write 0 to the leftmost value
Put Offset	The value in the input “In” is written on top of the indexed data defined by the table offset.
Clear Table	Resets the data in the table.
Search	The block output is written in which index of the table the value entered from the "InB" input among the table data is located.

Note: If the median is selected in the table operation, the values in the table indexes change because the table data is sorted from small to large.

15.8.5 Sample Applications

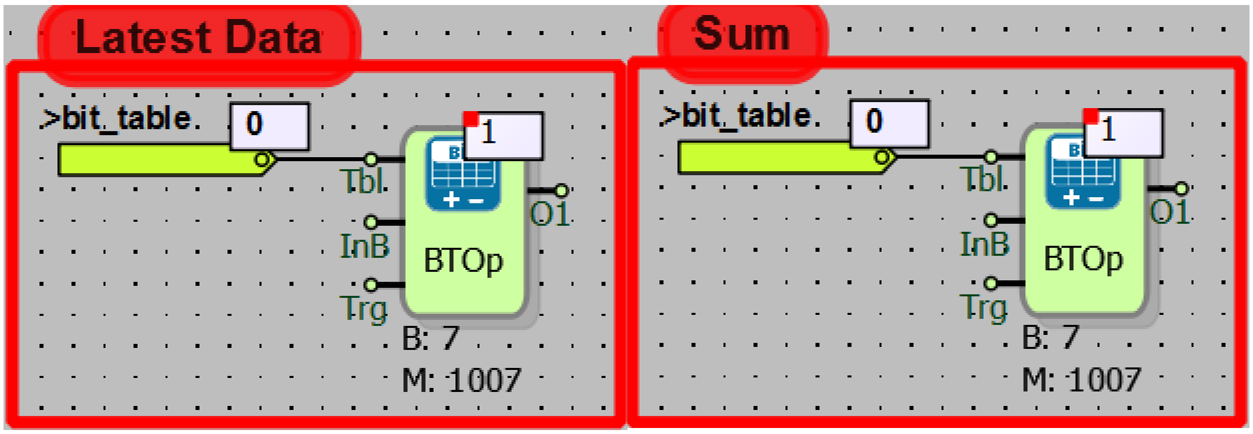


The table type "FILO" is selected in the sample applications, the table size is selected as 5 bytes and 5 bit value can be saved in the table.

İn the example,5 bit value is written on the table.

1	0	1	1	0

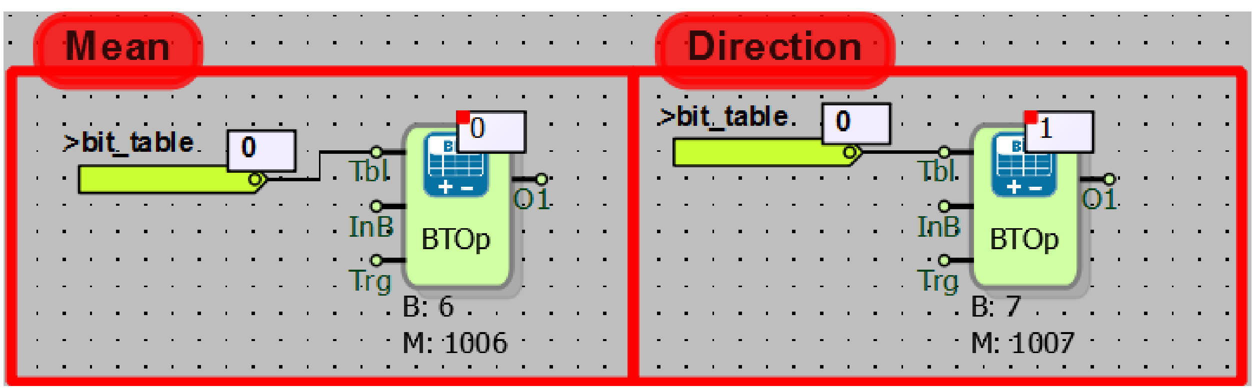
When "**LatestData and Sum"** is selected in the bit table operation block;



When "Latest Data" is selected; the value is written to the block output since the most recent value is 1 on the FILO.

When “Sum” is selected; since any of the bit values written to the table is 1, the result which is written on the output of the block is 1 as a result of the bit table average operation.

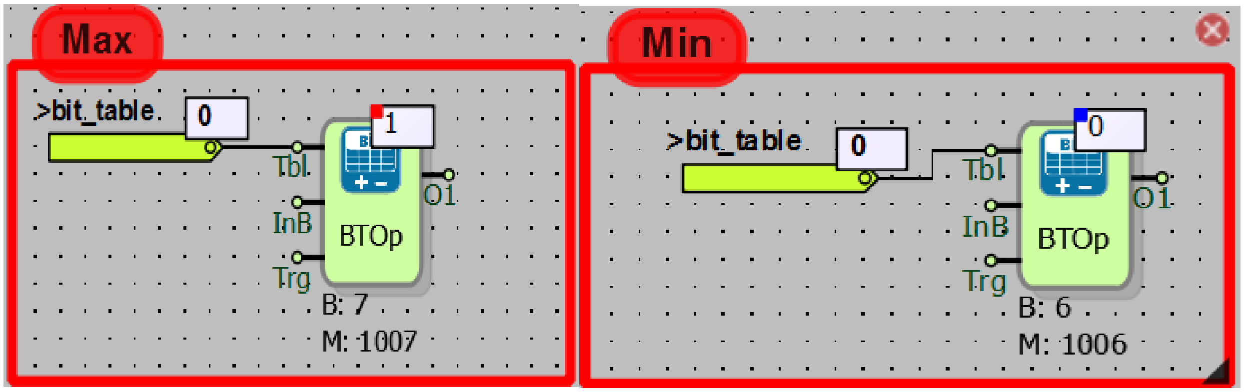
While "**Mean and Direction**" is selected in the bit table operation block;



When “Mean” is selected; since the values in the table are not all 1, the result is written as 0 in the output of the block as a result of the bit table average operation feature.

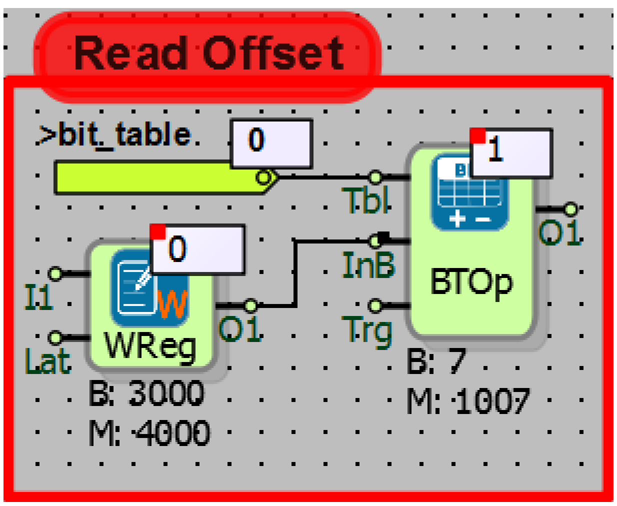
When “Direction” is selected; the last value added to the table is compared to the previous value. Since the previous value of the last one is 0, and the last value is 1, the result is written to the output of the block as the result of the increasing trend.

When "**Max and Min**" is selected in the bit table operation block;



When the table has bit value 1, the maximum value is 1 and when the table has bit value 0, the minimum value is written as 0 in the block outputs.

While “**ReadOffset**” is selected in the bit table operation block;

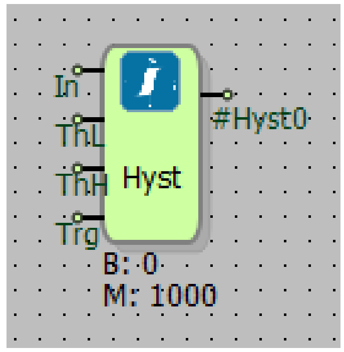


Read Offset: The table offset to be read in the bit table operation block can be selected from inside and outside of the block. In the example, the offset is chosen as 0 from out of the block. In this case, the value in the 0th offset is read as 1 with Reading Offset.

16 CONTROLLER BLOCKS

16.1 HYSTERESIS

16.1.1 Connections

<table>
<tr><td>In: Hysteresis block input</td><td rowspan="4">
</td><td rowspan="4">#Hyst0: Hysteresis block output</td></tr>
<tr><td>ThL: Bottom threshold</td></tr>
<tr><td>ThH: Upper threshold</td></tr>
<tr><td>Trg: Trigger input</td></tr>
</table>

16.1.2 Connection Explanations

<u>In: Hysteresis block input.</u>
It is hysteresis block input. Cannot be left blank.

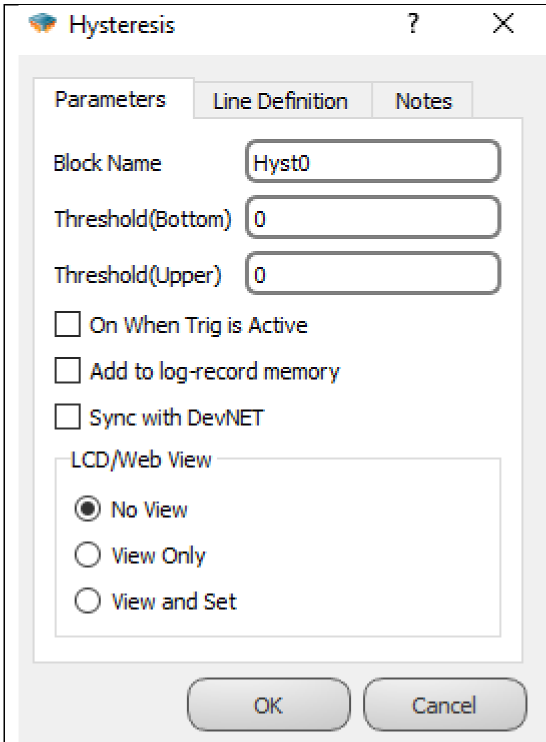
<u>ThL: Bottom threshold</u>
It is the input for bottom threshold value.

<u>ThH: Upper threshold</u>
It is the input for upper threshold value.

<u>Trg: It is trigger input.</u>
It is the trigger input. It can be left blank.

<u>#Hyst0: Hysteresis block output</u>
The hysteresis block output is logic high(1) or logic low(0) output.

16.1.3 Block Settings

	Threshold(Bottom): The bottom threshold value can be determined within the hysteresis block
	Threshold(Upper): The upper threshold value can be determined within the hysteresis block.
	On When Trig is Active: The incoming signal to block's "Trg" input will activate the block. If selected, block's "Trg" input cannot be left blank.

16.1.4 Block Explanation

It is used to create the switching range by switching on and off at the end points of the "bottom threshold and upper threshold" determined in on/off controlled systems.

"In" input is the hysteresis input to be referenced. It can not be left blank.

The "ThL" input is the lower threshold input, and if the input value "In" is less than the "ThL" then O1 output will become logic low(0).

The "ThH" input is the upper threshold input. If the input value is greater than the "ThH" input, the O1 output will become logic high(1).

The "ThL" and "ThH" inputs can be left blank and set in the block options.

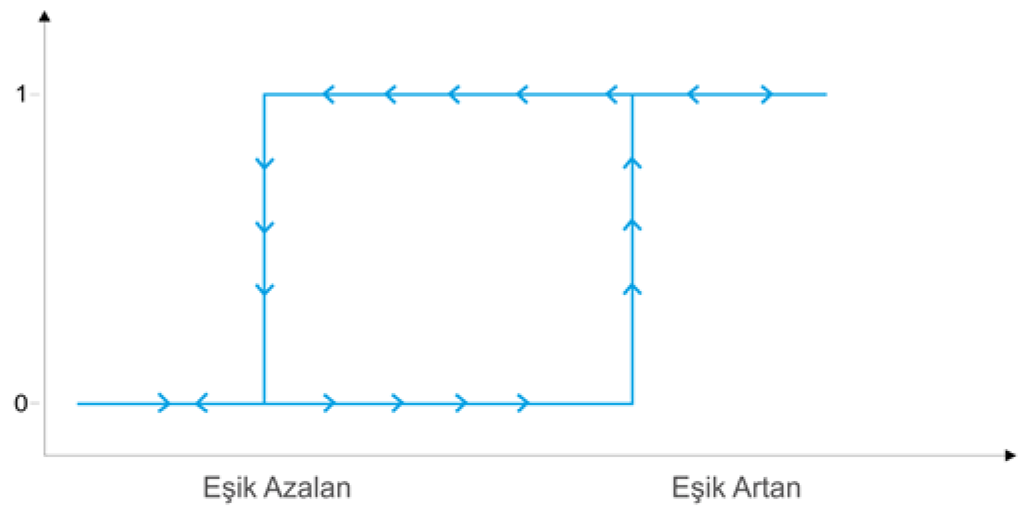
When the input value "In" is greater than the "upper threshold" value, the output O1 is logic high(1) until the input value "In" is a value smaller than the "bottom threshold" value.

When the input value "In" is less than the "bottom threshold" value, O1 output is logic low(0) until the input "In" is greater than the "upper threshold" value.

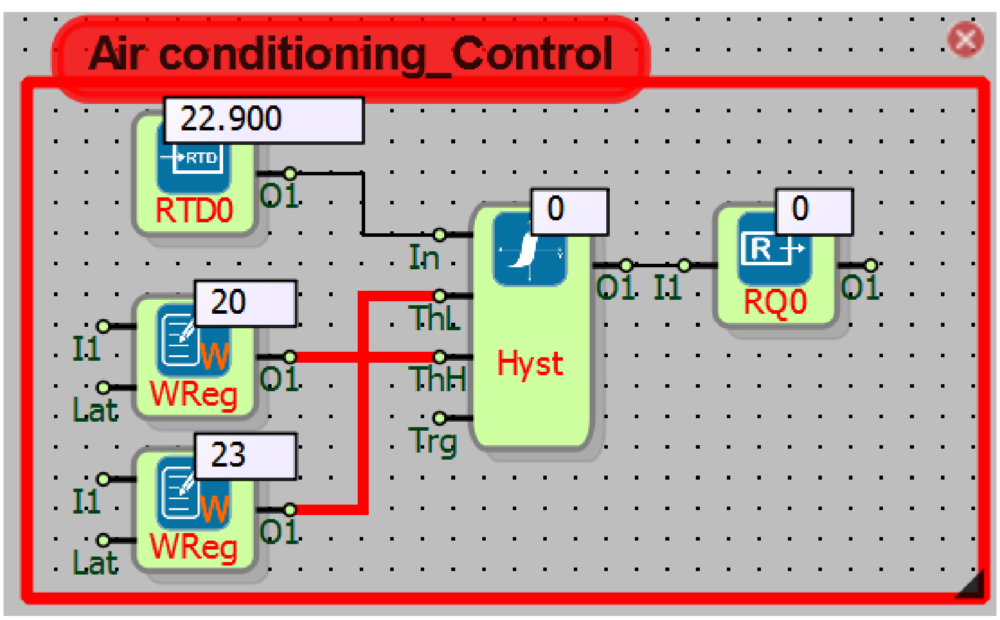
In a system where On/Off ambient temperature control is performed, if the ambient temperature is above the "upper threshold" value, the cooling system is started and the cooling system is shut down when the temperature value becomes lower than the "lower threshold". System is run to keep the temperature in a certain range. The larger the range "bottom threshold" to "upper threshold" range, the less the On Off frequency (the temperature sensor is connected to the block input "In" to measure ambient temperature).

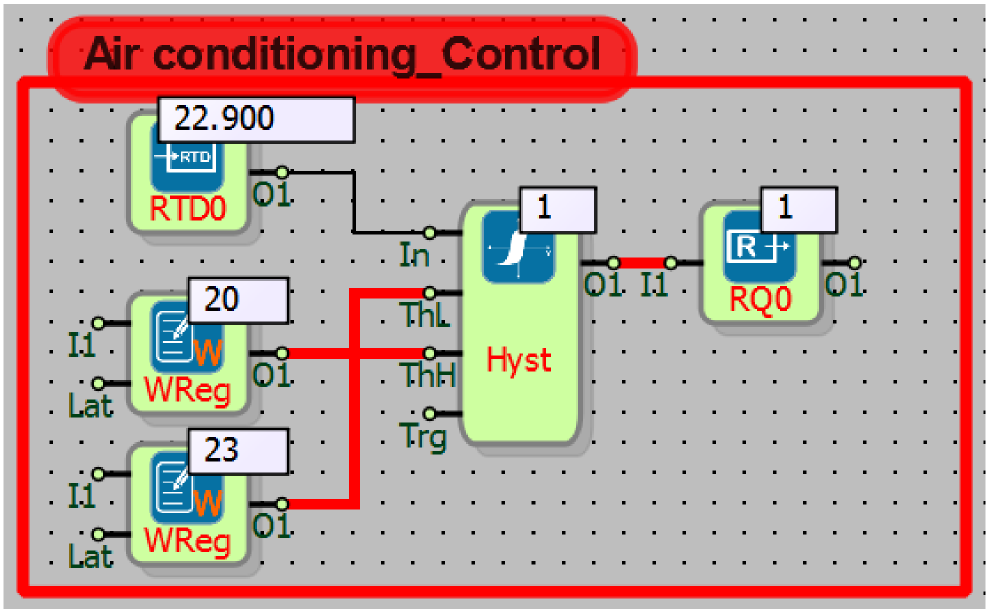
“Trg input” is trigger input, can be left blank. If "On When Trig is Active" is selected, the block becomes active at every rising edge triggered to Trg input. If "On When Trig is Active" is selected, the block Trg input can not be left blank.

16.1.5 Working Chart



16.1.6 Sample Application





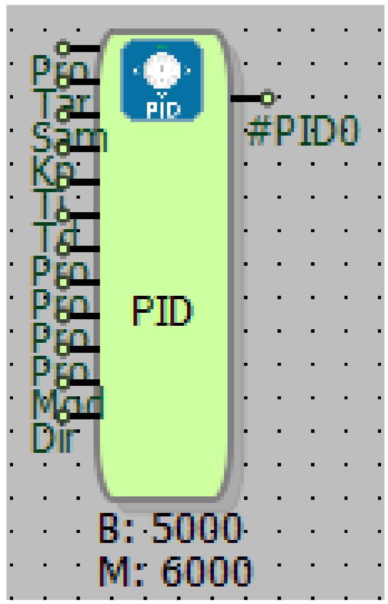
In the example,

It is aimed to turn on/off the air conditioner with RQ0 connected to the output of the hysteresis block. The temperature sensor is connected to the “In” input. The minimum temperature that the environment should be, is set by the "bottom threshold" and the maximum temperature by the "upper threshold".

The air conditioner turns on when the ambient temperature has risen above 23 ° C and then turns off when it is below 20 ° C and it doesn’t turn on until the temperature rises above 23 ° C again. The same cycle was repeated when the temperature rises above 23 ° so that the ambient temperature is kept constant between 20 ° and 23 °.

16.2 PID CONTROLLER

16.2.1 Connections

<table>
<tr><td>Pro: Process value analog data input</td><td rowspan="12">
</td><td rowspan="12">#PID0: Block exit</td></tr>
<tr><td>Tar: Target point entry</td></tr>
<tr><td>Sam: Sampling time</td></tr>
<tr><td>Kp: P coefficient input (%)</td></tr>
<tr><td>Ti: I coefficient input (sec)</td></tr>
<tr><td>Td: D Coefficient input (sec)</td></tr>
<tr><td>Pro: Process input min. entry</td></tr>
<tr><td>Pro: Process input max. entry</td></tr>
<tr><td>Pro: Process “Output Min.” entry</td></tr>
<tr><td>Pro: Process output max. entry</td></tr>
<tr><td>Mod: Mode selection</td></tr>
<tr><td>Dir: Direction selection</td></tr>
</table>

16.2.2 Connection Explanations

Pro: Process value analog data input

The instant value read from the system, eg the value read from a device in a temperature control application is connected here.

Tar: Target point input

The target point value input.

Sam: Sampling time (sec)

Sampling time value input.

It is the frequency of processing of the PID by reading the Pro input value to be controlled.

Kp: P coefficient input (%)

P value input. The Kp coefficient is proportional. This means that it determines, the PID controller will become active when PID process reaches to “what percentage” of the target point

Ti: I coefficient input (sec)

The I coefficient value input.

The present value and the target point are measured by integral effect and calculates the energy to be given to the system in order to reduce the error. The meaning of the “seconds” in the integral coefficient is that the errors will be referenced by how many seconds before.

Td: D coefficient input (sec)

The coefficient D input value.

Derivative acts as the opposite of the integral coefficient. It has an impact on braking effect in the system. The derivative and integral coefficients are in seconds. In the derivative process, the PID estimates the future states of the system. The “seconds” value indicates a how many seconds long forecast will be made.

Pro: Process input min. input

Process min value input.

The minimum value that the process input value can take is determined.

Ex: 4.0 for an input range of 4-20mA, 0 for an input range of 0-100, 0 for an input range of 0-65535, -100.0 for an input range of -100 ... + 100.

By this means, the PID block will automatically scale the input value.

Pro: Process input max. input

Process input max value input

The maximum value that the process input value can take is determined.

Ex: 20.0 for an input of range of 4-20mA, 100.0 for an entry from 0-100, 65535 for an input range of 0-65535., 100.0 for an input range of -100 ... + 100

By this means, the PID block will automatically scale the input value.

Pro: Process “Output Min.” entry

Process “Output Min.” value input.

The minimum value for the range that the actuator controlled by the PID accepts is determined. For example, if the PID process is connected to a frequency converter controlled by 4-20 mA, this value should be entered as 4.0.

By this means, the PID block will automatically scale the output value.

Pro: Process output max. input

The maximum value for the range that the actuator controlled by the PID accepts is determined. For example, if the PID process is connected to a frequency converter controlled by 4-20 mA, then max. 20.0 should be entered as the value.

By this means, the PID block will automatically scale the output value.

Mod: Mode selection (Autotunning, Automatic)

Mode selection block value input

Automatic: If selected, the PID starts to operate according to the defined block parameters. "1" must be entered when selecting from outside the block.

Autotunning: If selected, the PID block will autotune to determine the P, I, and D parameters. If you want to select from outside the block, "100" should be entered.

Direction: Direction selection (Forward, Backward)

Direction selection is block input.

If the direction input value is 1; The error information used in the PID process is calculated as follows:

En = ProcessInput - TargetPoint;

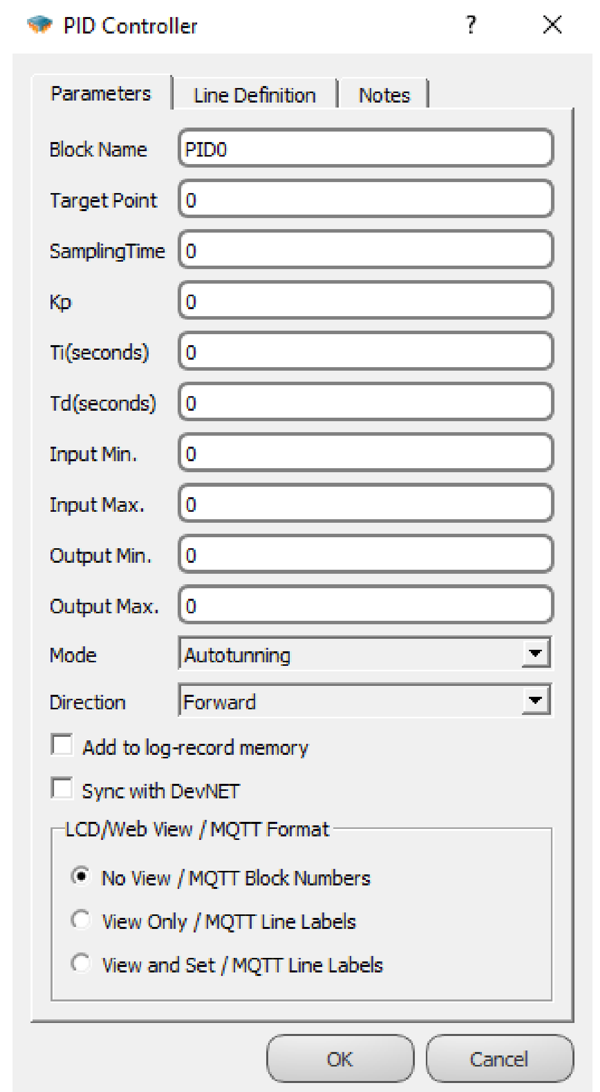
If the direction input value is 0; The error information used in the PID process is calculated as follows:

En = TargetPoint - ProcessInput;

#PID0: Block Output

It is the block output. It outputs values between the min-max range defined in the “process output“ inputs.

16.2.3 Block Settings



Target Point: Can be selected from inside or outside the block.
Sampling Time: Can be selected from inside or outside the block.
Kp: Can be selected from inside or outside the block.
Ti (seconds): Can be selected from inside or outside the block.
Td (seconds): Can be selected from inside or outside the block.
Input Min: Can be selected from inside or outside the block
Input Max: It can be selected from inside or outside the block.
Output Min: It can be selected from inside or outside the block.
Output Max: Max: Can be selected from inside or outside the block.
Mod: Outside the block, enter 100 for "Autotunning", 1 for "Auto" .
Direction: 0 for "forward" selection from outside the block, 1 for "back" selection.

16.2.4 Block Explanations

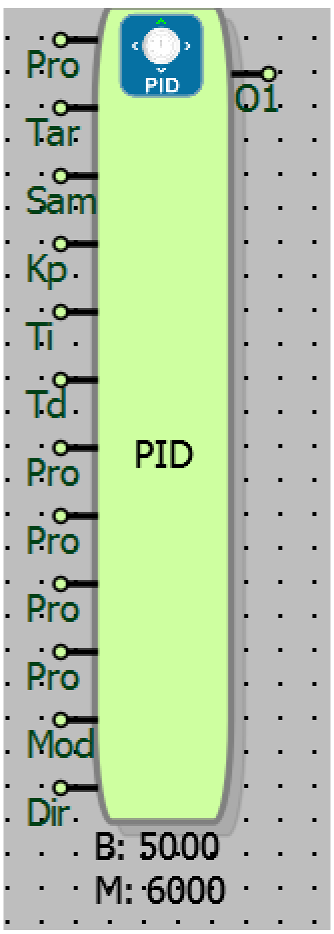
PID controller is one of the frequently used automatic control mechanisms in industrial and automatic control field. The PID controller performs Proportional Integrative and Derivative operations. A PID controller is a controller that is designed to stabilize a mechanism at a constant value in the most optimal time and to keep the value constant in the ideal values.

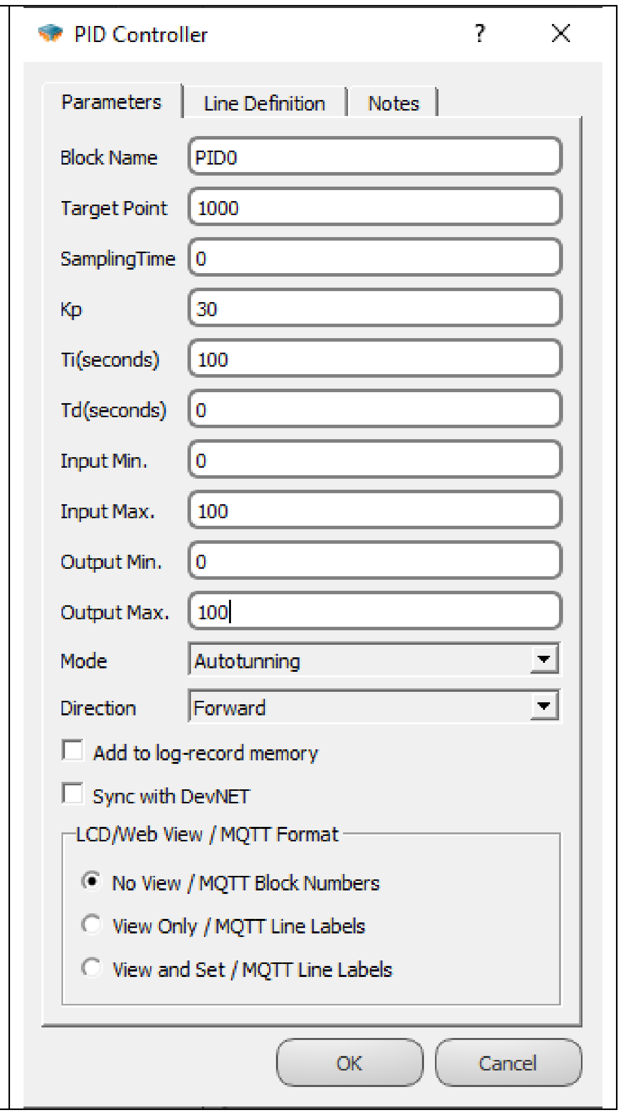
One of the most important points in PID applications is to determine the P, I, and D values that characterize the PID system. These values vary from system to system and should be optimized according to the application conditions. In order to determine these values, Mikrodev PLC has an "automatic tune" mechanism which calculates the values of P, I, D very practical and precise without the need of making any changes in the active project nor need a separate software-hardware etc.. This mechanism is activated by writing 100 values to the MOD input of the PID block.

In summary, the system prepares all components in the autotune mechanism. For the system,

the user is expected to select a target value for oscillating and a correct sampling time. The PID autotune mechanism will swing the system until it creates 8 peaks. Then it computes the system parameters according to these peak points and reports to the programmer from the USB port of the device.

16.2.5 Sample Application



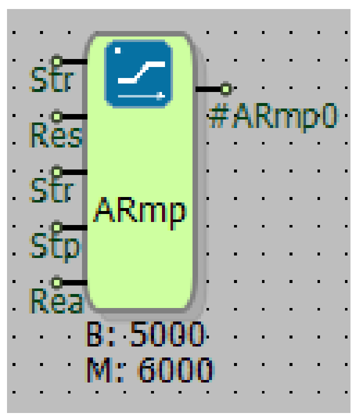


For example, if you want to use 0-10 V controller at PID output, you should enter PID “Output Min.” value as “0” and “Output Max.” value as “10”. If you want to get a current of 4-20mA, you need to write “4” as the “Output Min.” value, 20 as the “Output Max.” value. The PID controller has 12 inputs. Only the first input "Pro" input from these inputs is the input value to be processed as the reference value of the PID controller. This input cannot be left blank. Other entries can be left blank to set block options or allow values to be changed from the outside of block.

The proportional bandwidth set in the PID controller operates as “on-off logic” outside the limits of Kp. When the proportional band is activated, the PID controller starts to operate. The integral effect will give the system an energy up to the target point and as soon as the target point is reached and this energy is reduced, the derivative effect will also come into play and the system will try to keep the set value constant.

16.3 ANALOG RAMP

16.3.1 Connections

<table>
<tr><td>Str: Start/Stop</td><td rowspan="5">
</td><td rowspan="5">#ARmp0: Analog ramp block output</td></tr>
<tr><td>Res: Reset to start</td></tr>
<tr><td>Str: Initial value input</td></tr>
<tr><td>Stp: End value input</td></tr>
<tr><td>Rea: Time to finish value (ms)</td></tr>
</table>

16.3.2 Connection Explanations

<u>Str: Start/Stop</u>

The ramp block Start / Stop input.

<u>Res: Reset to start</u>

Sets the ramp block output to its initial value.

<u>Str: Inıtıal value input</u>

The ramp block initial value is entered.

Stp: End value input

The ramp block end value is entered.

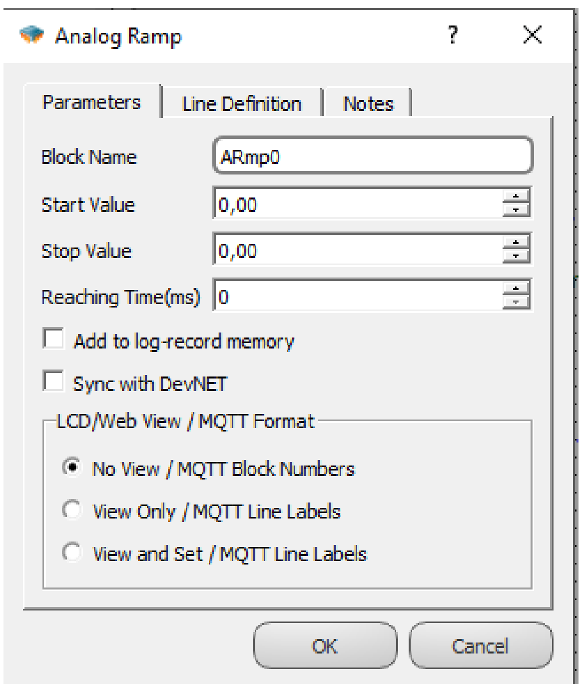
Rea: Time to finish value(ms)

Time to reach end value (ms) input.

#ARmp0: Analog ramp block output

Analog ramp block output.

16.3.3 Block Settings

	Initial Value: The initial value can be set from within the block.
	End Value: The end value can be set from within the block.
	Time of Arrival (ms): The time of arrival within the block can be adjusted.

16.3.4 Block Explanation

The analog ramp block is used in applications where it is necessary to reach a fixed value from a specified value with a constant acceleration within a certain time period.

“#ARmp0” block output value is reached with constant acceleration as soon as the logic input high(1) is applied to the input “Str” and the time to reach the stop value is reached.

The “#ARmp0” output reaching the stop value at the end of the reaching time preserves the stop value regardless of the position of the "Str" input.

If the “Str” input returns to logic low(0) position before the reaching time is completed, the “#ARmp” block output ramping stops. When the “Str” input is again logic high(1), the ramping process continues from where it left off.

The analog value between the start and end values can be measured on the “#ARmp0” output.

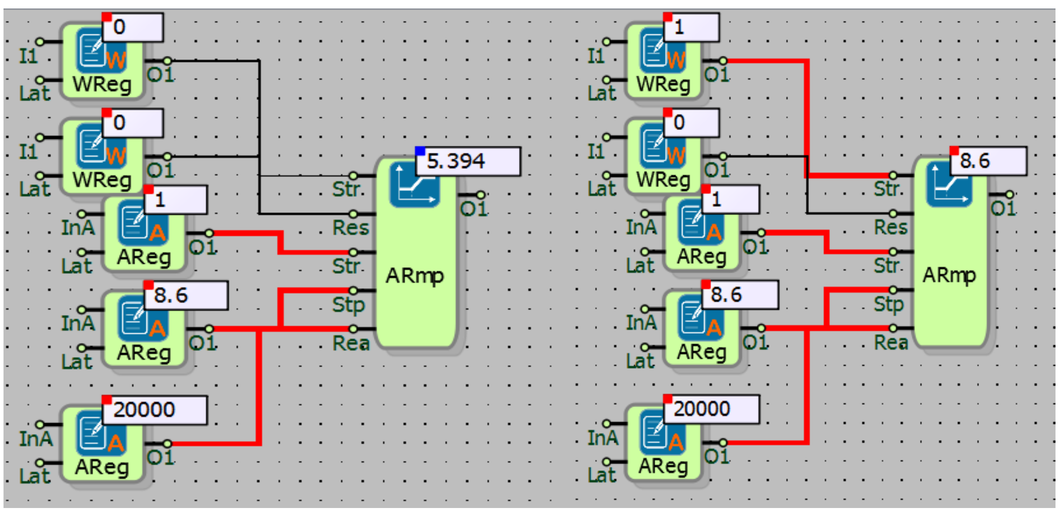
Start value, stop value and reaching time can be entered from the block object properties and from outside the block.

The logic must be applied logic high(1) to start from the "Str" input of the block and logic low(0) to stop.

The ramping operation is reset and the output “#ARmp” is fixed to the initial value when the rising edge trigger is applied to the block "Res" input.

The word, analog or long registers can be entered in the "Str", "Stp" and “Rea” inputs.

16.3.5 Sample Application

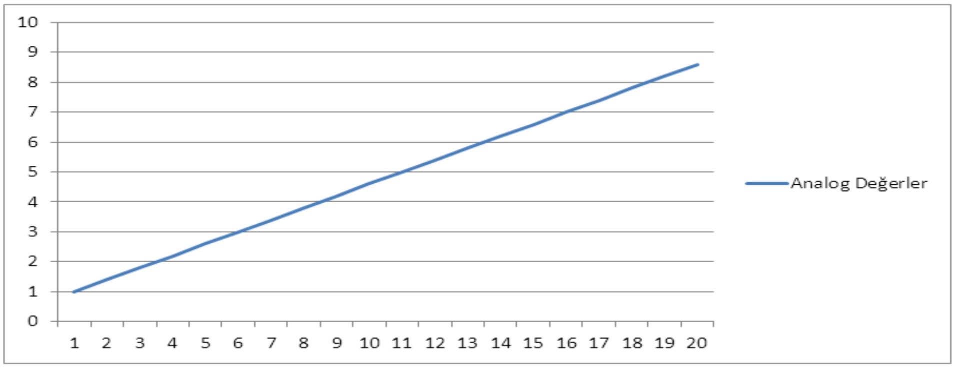


In the example; the blocks have been set starting value 1, ending value 8.6, and reaching time 20 seconds.

Initially Str input has been toggled to logic low(0) after a certain period of logic high(1) and O1 output stayed at 5.3943 because it did not reach the ramp end time.

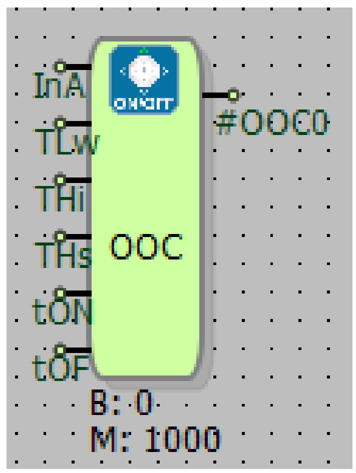
Then the Str input is again logic high(1), the ramp completes the remaining reach time and reaches the end value of 8.6.

The start value and end value graph on the time axis of reaching time are as follows.



16.4 ON/OFF CONTROLLER

16.4.1 Connections

InA: Controller block input		#OOC0: Block output
TLw: Bottom threshold		
THi: Upper threshold		
THs: Threshold hysteresis		
tON: ON standby time (ms)		
tOFF: OFF standby time (ms)		

16.4.2 Connection Explanations

InA: Controller block input

The controller block is the input. Can not be left blank.

TLw: Bottom threshold

The lower threshold input value

THi: Upper threshold

The upper threshold input value

THs: Threshold hysteresis

Threshold hysteresis input value. Hysteresis can also be added in control comparison.

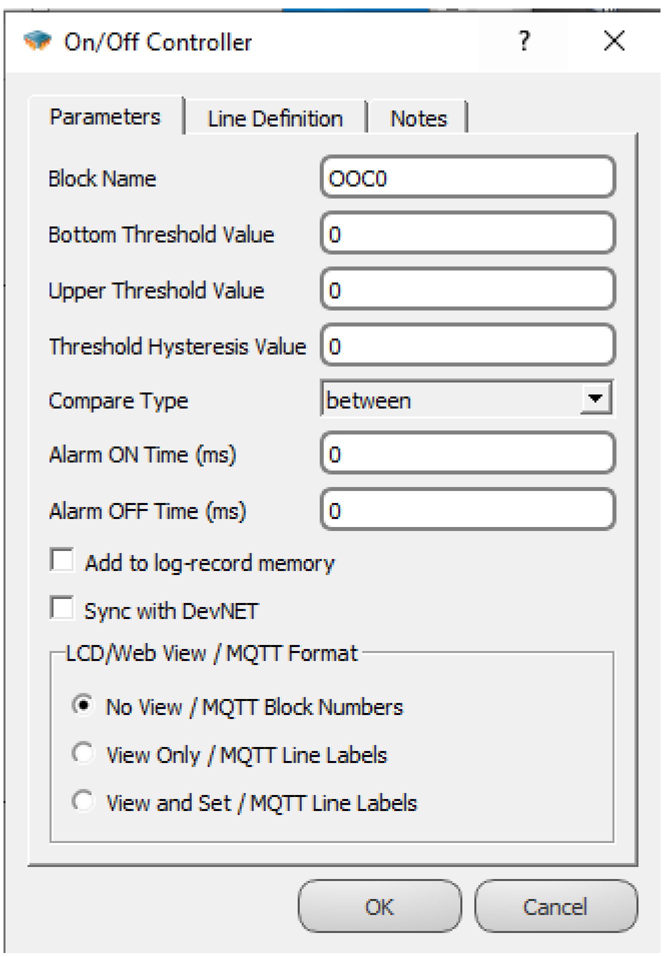
tON: ON standby time (ms)

When “#OOC0” output is in OFF state, if the block input compare condition becomes logical high(1) position and this condition is satisfied for tON duration, “#OOC0” output turns ON

tOF: OFF standby time (ms)
When “#OOC0” output is in ON state, if the block input compare condition becomes logical high(1) position and this condition is satisfied for tOFF duration, “#OOC0” output turns OFF

#OOC0: Block output
It is binary block output.

16.4.3 Block Settings

<table>
<tr><td rowspan="6">
</td><td>Bottom Threshold Value: Bottom threshold value can be adjusted from within the block.</td></tr>
<tr><td>Upper Threshold Value: Upper threshold value can be adjusted from within the block.</td></tr>
<tr><td>Threshold Hysteresis Value: Threshold hysteresis value can be entered from within the block.</td></tr>
<tr><td>Compare Type: Comparison method for ON / OFF control selected.</td></tr>
<tr><td>Alarm On Time(ms): Alarm on time value can be adjusted from within the block.</td></tr>
<tr><td>Alarm Off Time(ms): Alarm off time value can be adjusted from within the block.</td></tr>
</table>

16.4.4 Block Explanation

The process value controlled in the ON - OFF method, which is one of the most basic control methods, is operated by OFF or ON states. If the input value of the process meets the defined conditions, output status is ON, otherwise output status is OFF.
Mikrodev ON/OFF control function block fulfills this basic ON-OFF control method with a number of superior features. The following comparison types are used to check the process input value.

Comparison Type	Lower Threshold Value	Upper Threshold Value
Between	Active	Active
Greater	Active	-
Smaller	Active	-
Out of Range	Active	Active
Equal	Active	-
Smaller or Equal	Active	-
Greater or equal	Active	-
Not equal	Active	-

tON or tOFF times are entered to prevent the output from fluctuating due to the instantaneous faulty data and to add only the delay, even if the comparison operation requires state change.

When block output is OFF, block time counter is started if ON condition occurs at block input. Block output is toggled to ON if ON condition does not change until tON time is reached. Similarly, when block output is ON, block time starts when the OFF condition occurs at the block input, and block output is OFF when the OFF condition does not change until tOFF time is reached.

Both values must be set to 0 to cancel tON and tOFF operations.

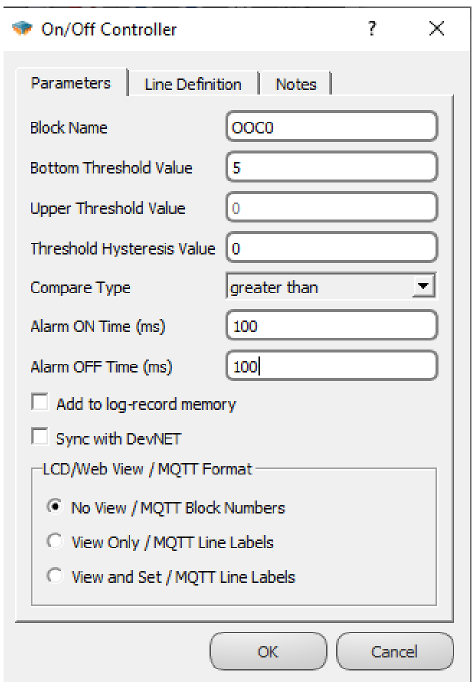
Hysteresis can be used in addition to the tON-tOFF mechanism if it is desired that the block output does not make any sudden changes in particularly slowly changing signals relative to the process input value and ambient noise.

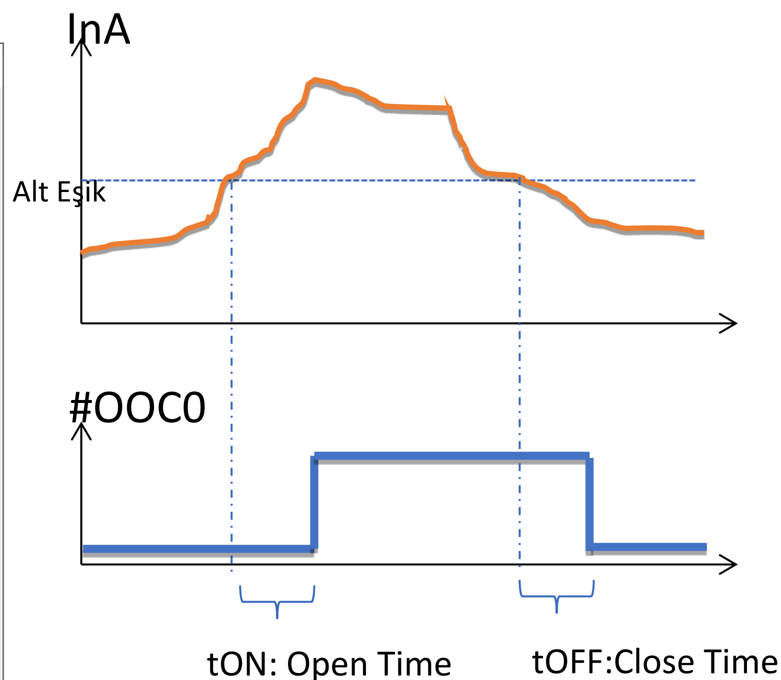
In Hysteresis feature; when the output “#OOC0” changes from ON to OFF state and from OFF to ON state , even if the input condition changes, if the hysteresis threshold is not exceeded , the output state does not change. The output state changes when the hysteresis threshold is exceeded.

16.4.5 Sample Applications

16.4.5.1 Example 1

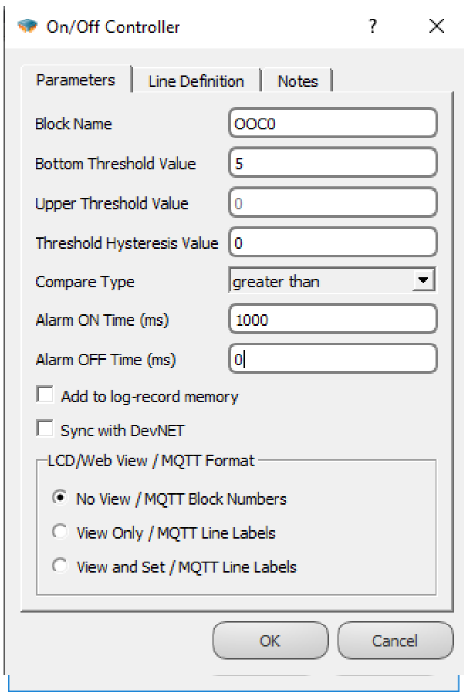
Block process input value has been controlled with ON-OFF control according to the compare type Greater Than. The change in block outputs is delayed as much as the tON and tOFF time values.

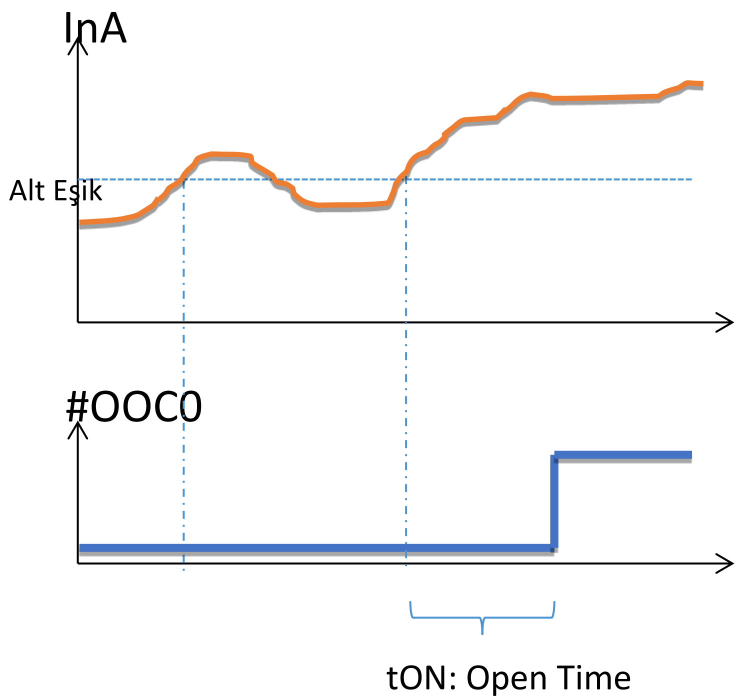




16.4.5.2 Example 2

The block process input value is controlled with ON-OFF control according to the compare type Greater Than. After the instance, the input value has satisfied the condition, the “#OOC0” output is ON-delayed as long as the tON, then the “#OOC0” output is in logical high(1) position. (The output of “OOC0” is logic high(1) after 1 second after the InA value has risen over 5).





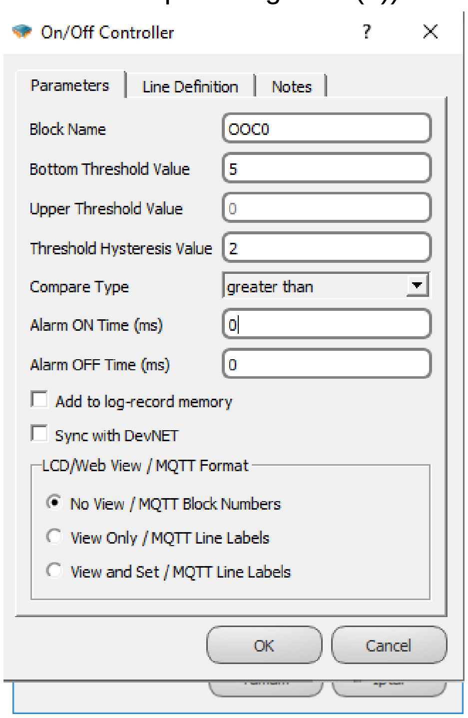
16.4.5.3 Example 3

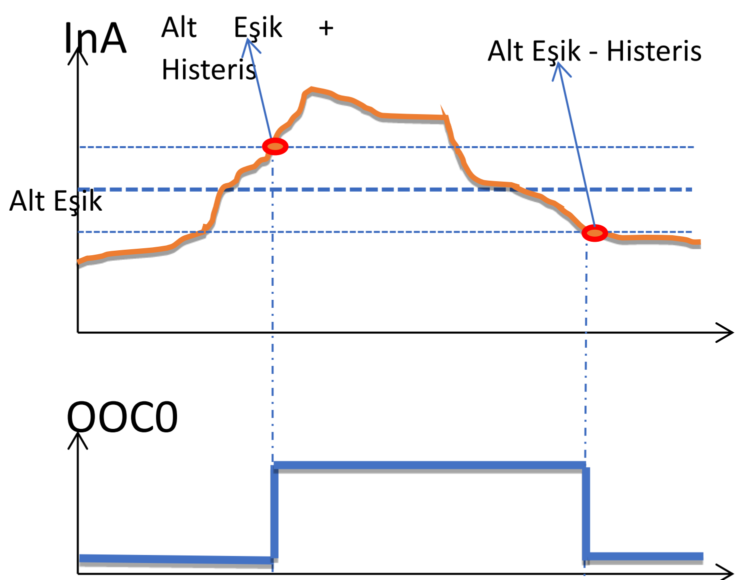
The block process input value is controlled with ON-OFF control according to the compare type Greater Than. Hysteresis value is also entered, and hysteresis is activated.

In the hysteresis comparison method:

The transition from the OFF state of the block “OOC0” to the ON state will occur if "Compare point is greater than threshold + hysteresis value." (If the value of InA is above 5 + 2 = 7, the output of “OOC0” is logic high(1).)

The transition from the ON state to the OFF state of the block “OOC0” output will occur if Compare Point is lower than Threshold-Hysteresis Value (if the value at the INA input is below 5-2 = 3, the “OOC0” output is logic low(0)).

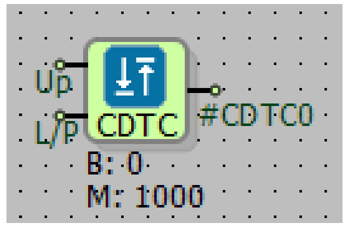




Note: The threshold hysteresis value and tON (turn on time) and tOFF (turn off time) features can be used at the same time. tON or tOFF will get activated after hysteresis threshold is exceeded.

16.5 CHANGE DETECTOR

16.5.1 Connections

<table>
<tr><td>Up: Block input</td><td rowspan="2">
</td><td rowspan="2">#CDTC0: Block output</td></tr>
<tr><td>L/P: Change value</td></tr>
</table>

16.5.2 Connection Explanations

Up: Block input

It is the block input value from which to determine whether there is a change or not.

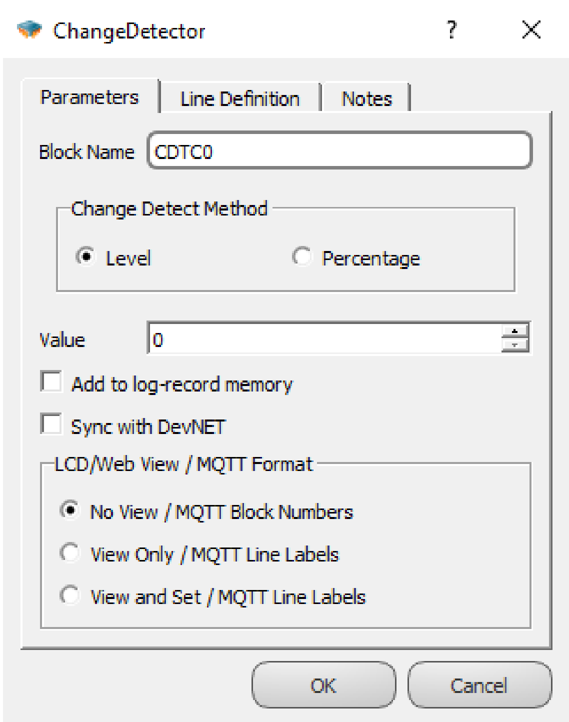
L/P: Change value

The change values can be selected from the L/P input from outside the block or from within the block for the "percentage or level" options selected from within the block.

#CDTC0: Block output

This is the output for a one cycle pulse when there is a change over the change value determined at the input of “Up”.

16.5.3 Block Settings

<table>
<tr><td rowspan="3">
</td><td>Level: If selected, pulse occurs at #CDTC0 according to level change at Up input.</td></tr>
<tr><td>Percentage: When selected, a pulse occurs at #CDTC0 according to the percentage change at the Up input.</td></tr>
<tr><td>Value: At an output change (percentage or level) greater than the input value, a pulse occurs at the output.</td></tr>
</table>

16.5.4 Block Explanation

The Change Detector block is used when changes to any block value need to be monitored.

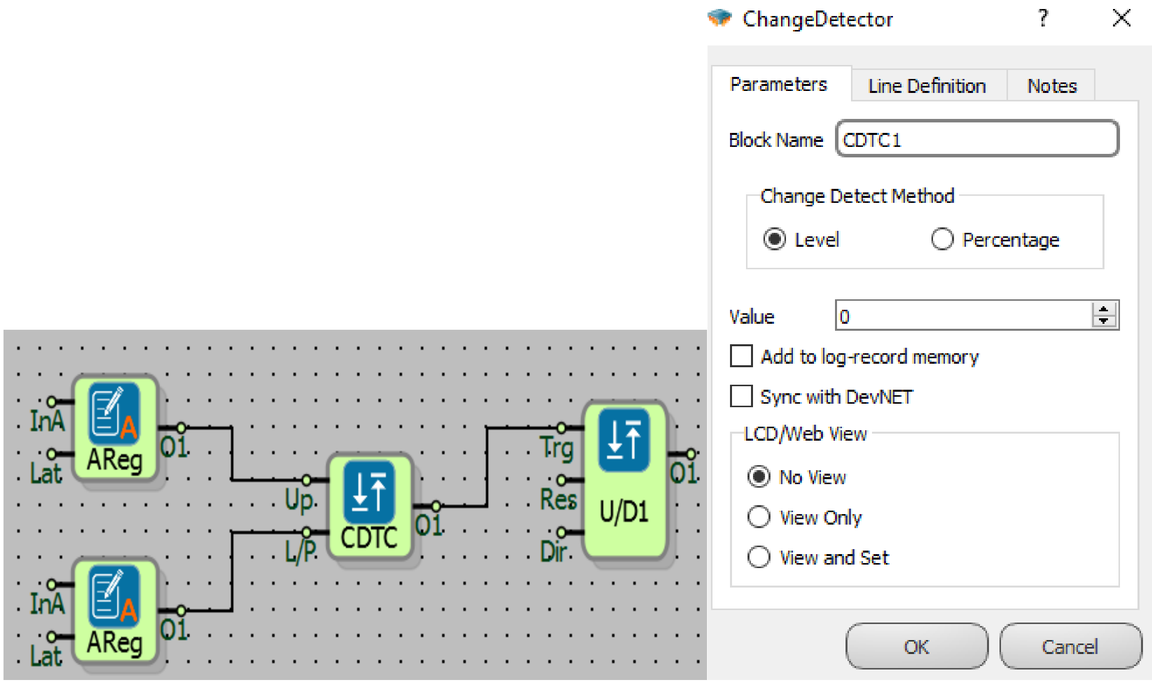
If the difference between the present value of the value at the “Up” entry and the next value is greater than the specified percentage or level change value, a momentary pulse is generated at the “#CDTC0” block output.

You should connect the block, when whose value changes the “#CDTC0” output should generate a pulse, to the “Up” input .(Counter, register, etc.)

In the block options, the value change is selected as the percentage or level change.

To generate a pulse at output “#CDTC0”, the minimum change value of the “Up” input can be set from the value window in the block options or from the L/P input outside the block.

16.5.5 Sample Application



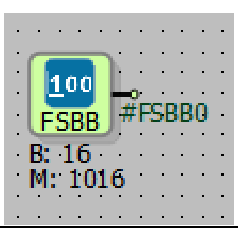
In the Example;

Level is selected as the Change Detector method, and 2 is selected as the level change value from outside the block. When the value of the analog register at the input of “Up” changes more than 2, a pulse is generated at the output of “O1”. Pulses are counted by the up counter connected to the output “O1”.

17 SYSTEM BLOCKS

17.1 FIRST SCAN BIT

17.1.1 Connections

	#FSBB0: Block output

17.1.2 Connection Explanations

Sta: Block output

It is block output.

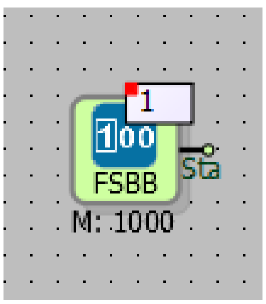
17.1.3 Block Settings

There are no block settings.

17.1.4 Block Explanation

This block generates logic(1) output when Logic Controller is activated and as long as it stays in active state. It is used to bring the Logic Controller to reference values and states.

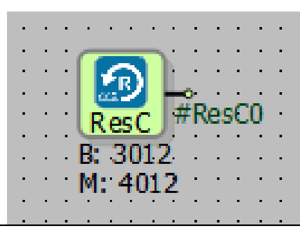
17.1.5 Sample Application



When PLC is started, the block gives logic(1) to block output.

17.2 RESET COUNTER

17.2.1 Connections

	#ResC0: Block output

17.2.2 Connection Explanations

#ResCO: Block output

It is block output.

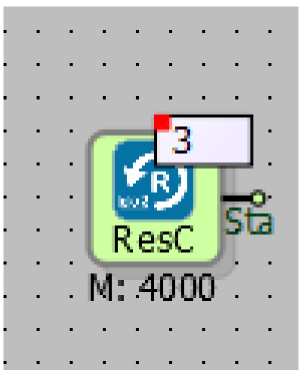
17.2.3 Block Settings

There are no block settings.

17.2.4 Block Explanation

The reset count of the device is written to the output. After every power reset operation, output block value is increased by one. If a logic project is loaded into the device, RESET counter block value is set to “1”.

17.2.5 Sample Application



It is displayed the reset count of the device.

17.3 SYSTEM RESET

17.3.1 Connections

Trg: Trigger Input	

17.3.2 Connection Explanations

Trg: Trigger input

It is block trigger input.

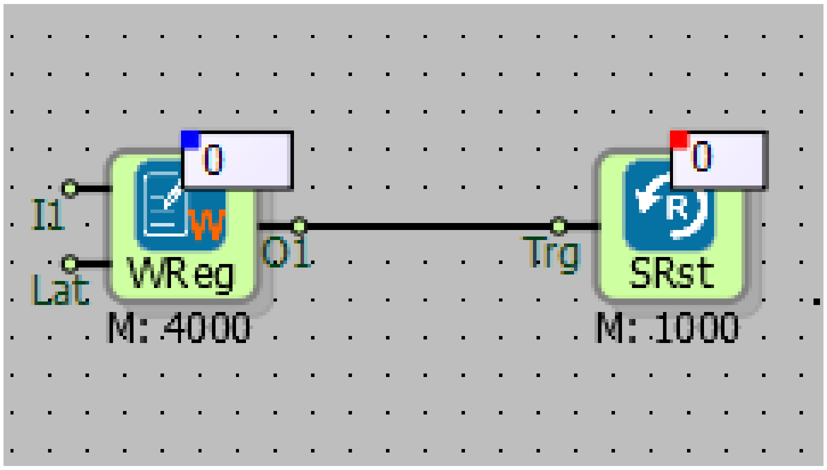
17.3.3 Block Settings

There are no block settings.

17.3.4 Block Explanation

In case of rising edge trigger signal is applied to Trg input, the device performs a soft RESET.

17.3.5 Sample Application

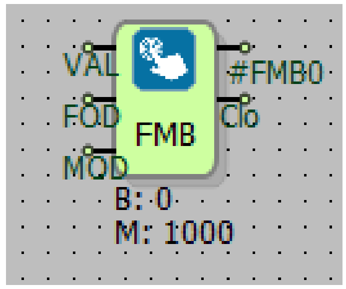


When a value different from “0” is written on the word register connected to Trg input, the device is reset.

18 HVAC BLOCKS

18.1 FLOATING MOTOR

18.1.1 Connection

<table>
<tr><td>VAL: Valve opening level input (%)</td><td rowspan="3">
</td><td>#FMB0: Opening output</td></tr>
<tr><td>FOD: Full opening duration input</td><td rowspan="2">Clo: Closing output</td></tr>
<tr><td>MOD: Minimum opening duration input</td></tr>
</table>

18.1.2 Connection Explanations

VAL: Valve opening level input (%)

It is the valve open level as a percentage (%).

FOD: Full opening duration input

It is the time duration from full closed to full opening.

MOD: Minimum opening time duration

It is the time duration for minimum opening time.

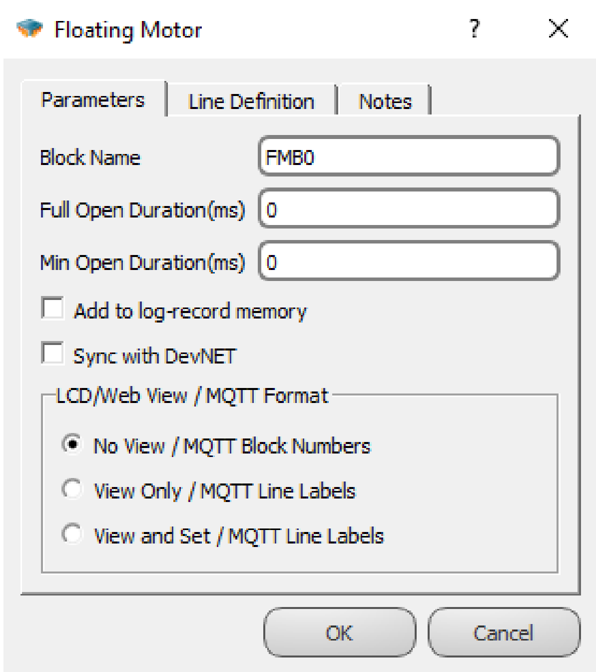
#FMB0: Opening output

It is the opening output which generates logic low(0) or logic high(1).

Clo: Closing output

It is the closing output which generates logic low(0) or logic high(1).

18.1.3 Block Settings

<table>
<tr><td rowspan="2">
</td><td>Full Open Duration(ms): The full opening time (FOD) can be entered from within the block settings.</td></tr>
<tr><td>Min Open Duration(ms): Minimum opening time (MOD) can be entered from within the block settings.</td></tr>
</table>

18.1.4 Block Explanation

It is used in Proportional or PID control applications.

Equipment connected to the output will be turned on as long as the logic (1) signal sent from the “#FMB0” output. The equipment connected to the output will shut down as long as the logic (1) signal sent from the “Clo” output.

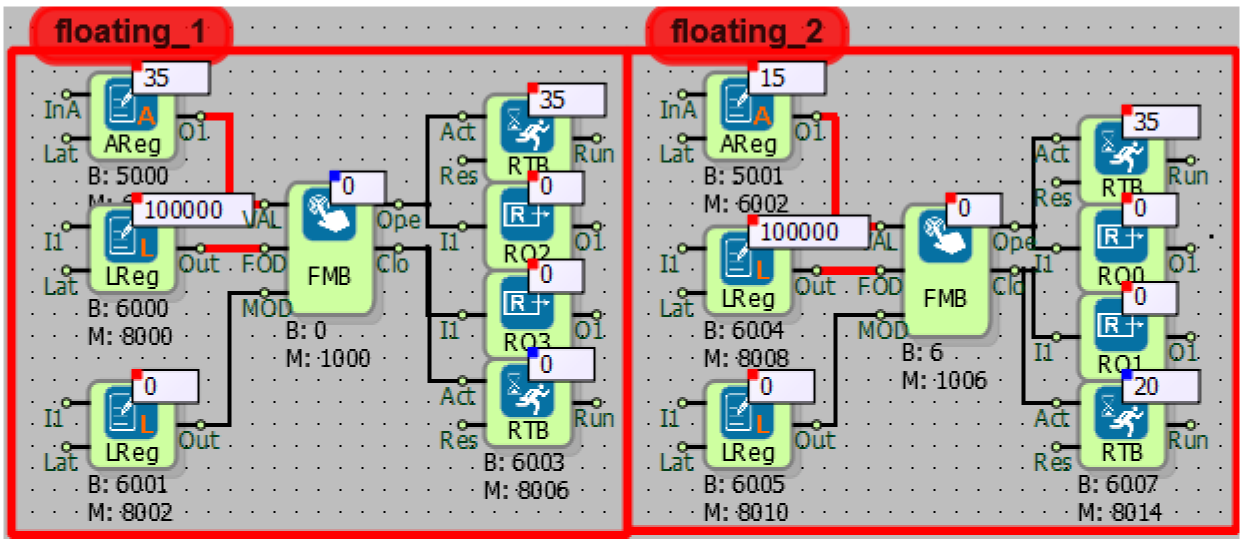
The "#FMB0" and "Clo" outputs of the block generate a logic low(0) or logic high(1) signal according to their control status. The two outputs does not produce a logic high(1) signal at the same time.

The time to one hundred percent opening time is specified in milliseconds in the Full Open Duration (FOD). The minimum running time of the equipment is specified in Minimum Opening Duration (MOD) in milliseconds. If the percentage change rate at the "VAL" entry corresponds to a smaller value than the minimum opening duration "MOD", the run signal will not be sent to the output. (If MOD: 1 sec, FOD: 100 sec. and the VAL change is greater than %1, the equipment moves.)

The "VAN" input specifies how much of the equipment should be opened in percent. Precise data input can be achieved by connecting an analog register to this input.

32-bit long value can be entered for full opening and minimum opening values.

18.1.5 Sample Applications



In the examples;

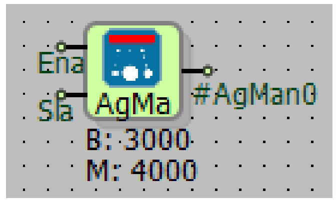
The full opening time (FOD) was entered as 100 seconds. Minimum opening time (MOD) value is 0. This means that the smallest change in the VAL input will also cause a change in the outputs.

In the case of floating 1; The VAL input is entered 35 for the 35% opening of the initially closed valve. The open output has been logic low(0) after becoming logic high(1) for 35 seconds. Thus, Floating 1 valve was opened by 35%.

In the case of floating 2; The valve is initially opened at 35%. Then the opening of the valve was reduced to 15%. The “Clo” output has been logic low(0) after becoming logic high(1) for 20 seconds. Thus, the Floating 2 valve open rate is reduced from 35% to 15%.

18.2 AGING MANAGER

18.2.1 Connection

Ena: Enable input		#AgMan0: Working slave no
Sla: Slave count		

18.2.2 Connection Explanations

Ena: Enable input

It is block activation input.

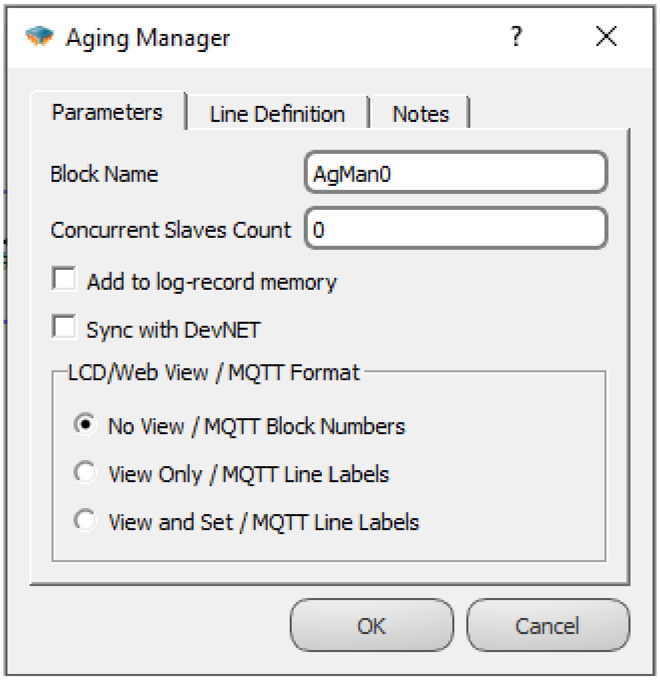
Sla: Slave together count

The number of slaves to be activated at the same time.

#AgMan0: Working slave no

It is the output of the block which shows the number of the running slave and connected to the input of the "Mas" of the aging member blocks.

18.2.3 Block Settings

	Concurrent Slaves Count: The number of slaves (member) connected to the block output which will be active at the same time can be identified from within the block.

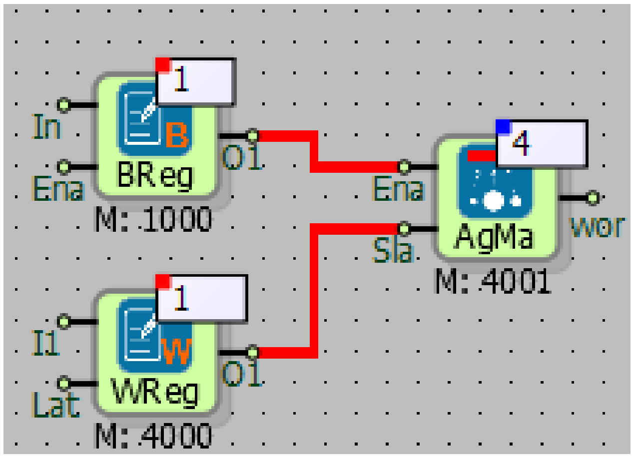
18.2.4 Block Explanations

“#AgMan0” output connected to “Mas” input of aging member blocks, NOT used alone or with other blocks.

As long as the logic high(1) signal is input to “Enb”, the block becomes active and activates the connected aging members. Up to 10 aging members can be connected to the block output.

The number of aging member blocks which are active at the same time can be set from within the block or from the block "Sla" entry. (Eg: if this value is set to 3 and 7 members are connected to the “#AgMan0” output, 7 members will be active in groups of 3.)

18.2.5 Sample Applications



The binary register is connected to the “Ena” input to activate the block.

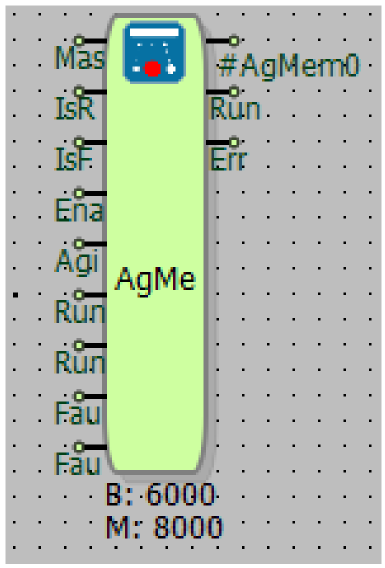
The "Sla" input is linked to the word register to determine how many Aging Members will be active at the same time.

On the block “#AgMan0” output there is information about which of the connected members is running. This information is given as bits of the output value. For example, in the above example, the output value equals 4, binary equals to “0100”, which means that the second slave is active.

The output of this block in the ready state must be connected to the "Mas" input of the Aging Member blocks.

18.3 AGING MEMBER

18.3.1 Connections

<table>
<tr><td>Mas: Aging manager input</td><td rowspan="9">
</td><td rowspan="3">#AgMem0: Run time output</td></tr>
<tr><td>IsR: Running info input</td></tr>
<tr><td>IsF: Error info input</td></tr>
<tr><td>Ena: Block activation input</td><td rowspan="3">Run: On/off output</td></tr>
<tr><td>Agi: Aging time input</td></tr>
<tr><td>Run: Run time reset input</td></tr>
<tr><td>Run: Current Aging age input</td><td rowspan="3">Err: Block error output</td></tr>
<tr><td>Fau: Error reset input</td></tr>
<tr><td>Fau: Timeout input for error</td></tr>
</table>

18.3.2 Connection Explanations

Mas: Aging manager input

The "#AgMan0" output of the Aging Manager block is connected. Another type of block cannot be connected.

IsR: Running info input

Equipment run information is entered.

IsF: Error info input

Equipment information such as thermal, fault, error is entered.

Ena: Block activation input

The block is activated by the logic high(1) signal.

Agi: Aging time input

The aging time in minutes.

Run: Run time reset input

With the rising edge trigger, the run time information on the block is reset.

Run: Aging age input

Aging members are the input of current working time information.

Fau: Error reset input

With the rising edge trigger, the error information at the block output is reset.

Fau: Timeout error input

It is the timeout error input for waiting time of error information from block output.

#AgMem0: Run time output

It is run time output for equipment running time in minutes.

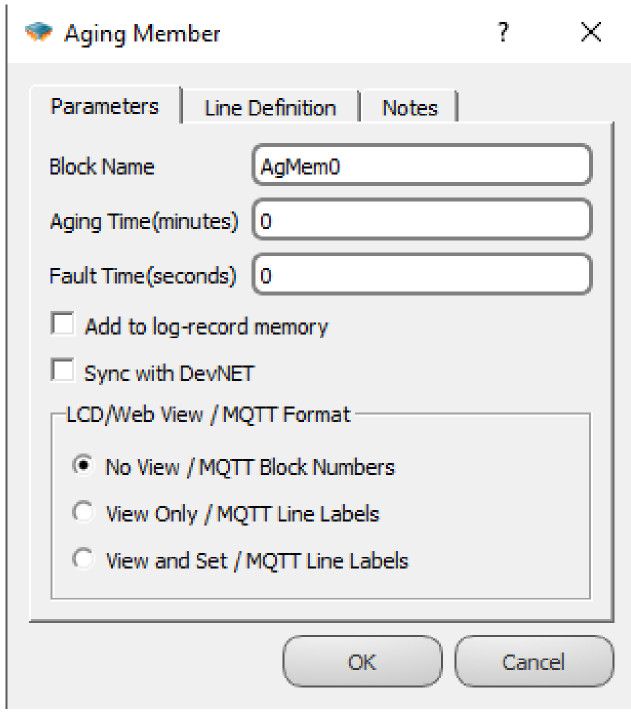
Run: On/off output

It is on/off output for the equipment which is logic low(0) or logic high(1).

Err: Block err output

When block err input is logic high(1) or the timeout for the error exceeded, block err output is logic high(1).

18.3.3 Block Settings

<table>
<tr>
<td rowspan="2">
</td>
<td>Aging Time(minutes): The Aging Time in minutes can be entered from within the block.</td>
</tr>
<tr>
<td>Fault Time(second): The Fault Time in minutes can be entered from within the block.</td>
</tr>
</table>

18.3.4 Block Explanations

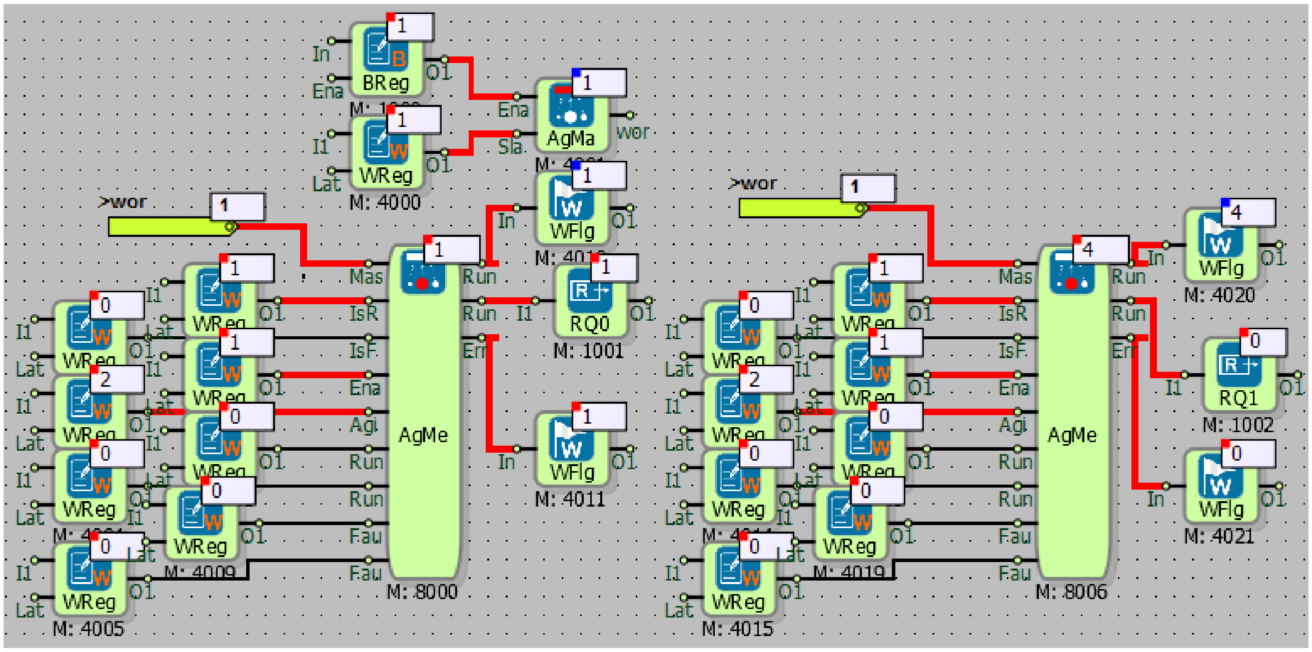
It is used in applications where several equipment must be started and stopped in sequence for a certain period of time. It is also called aging.

This block is used when the pumps in a pump station are operated in the determined sequence and durations. After a pump completes the aging period, it is stopped and another pump which is included in the aging scenario period is started and the system enters into a cyclic loop and the same pumps are used.

This block used with the Aging Manager block.

Block Inputs	Explanations
Mas	The "#AgMan0" output of the Aging Manager block is connected to the "Mas" input.
IsR	Operating information(On / Off) of the equipment is connected, logic low(0) or logic high(1)
IsF	A thermal fault or other fault information can be entered to prevent the system from being forced into operation. When the logic input (1) signal is input to the error input, the output of the "IsF" at the block output becomes logic (1), and the equipment run output "#AgMem0" at the block output changes to the logic (0) state to prevent further malfunctions in the system. Even if the error at block error input returns to logic (0), the error output at block “IsF" will not turn to logic (0). The block error output is reset when the rising edge trigger is applied to the block error reset input.
Ena	Logic (1) must be applied for the block to be active. If you do not want to operate due to the maintenance, malfunction, etc. of the equipment connected to the block, the "Ena" input is disabled (0) and the equipment is deactivated. Other equipment in aging continues to work in sequential order.
Agi	The aging time is entered in minutes. The equipment connected to the block runs as long as the aging time, then stops, the operating turn comes to the other equipment. It can be set from inside and outside the block.
Run	The runtime at the block output is reset at each rising edge trigger.
Run	The current operating times of the equipment in the system are entered. Those with higher run times are run less, balancing their run times, thus establishing standard run time periods. The maintenance and replacement periods of the equipment are standardized.
Fau	The rising edge trigger must be applied in this input to reset the fault when the block fault output is at logical (1) state. (If the block has thermal, fault, error, etc. at the fault input, it must be cleared before resetting.)
Fau	There are two factors that cause the block error output to be logic (1). The first is the information of the thermal, fault, error etc. coming to the fault input. 2nd is; If no operation fault, or thermal information from the equipment is received even though the block output is switched on, the error timeout period is taken into account. When the error timeout period is exceeded, the block error output becomes logic (1).
Block Output	**Explanations**
#AgMem0	It is the block runtime information in minutes. It can be reset at the rising edge trigger applied to the reset runtime input “Run”
Run	It is the connection output to the equipment to be operated. Since equipment On/Off control with Mikrodev PLC products are made with digital output (DQ) or relay output (RQ); digital output (DQ) or relay output (RQ) must be connected to the block output. If the equipment is connected to the output, the digital output (DQ) or the relay output (RQ) block must be selected.
Err	It is the error output. 1- When any thermal, fault, error, etc. occurs at the block fault input, the error output becomes logic (1). 2- If the running information does not seem to appear at “IsR” input even though On/Off control logic is output (1), error output becomes logic (1) after the time out duration. In order to reset the error output, the rising edge trigger must be applied on the error reset input.

18.3.5 Sample Applications



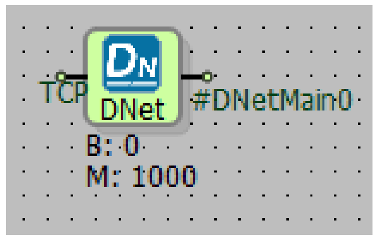
In the example;

2 Aging Member blocks have been added to an aging manager. The Aging Manager block's "Sla" input is entered as "1" which indicates that the members will be run "one by one". Two minutes were selected for both members as the aging period.

The relay output connected to the first member became logic (1) for 2 minutes, after 2 minutes the first member output became logic (0) and the second member output became logic (1). After 2 minutes, the second member becomes logical (0) and the first member becomes logical (1) again. The system has thus entered the periodic operation cycle.

18.4 DEVNET MAIN

18.4.1 Connection

TCP: Connection parameters input		#DNetMain0: Block connection output

18.4.2 Connection Explanations

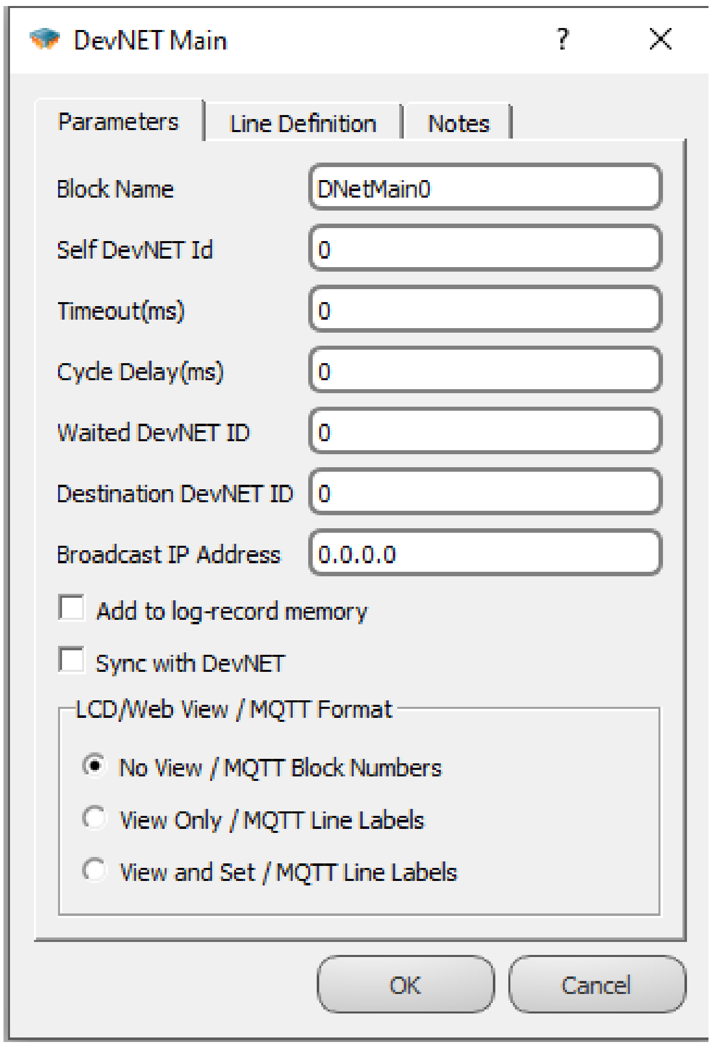
TCP: Connection parameters input

It is the input connection for parameters.

#DNetMain0: Block connection output

It is the output connection of the block

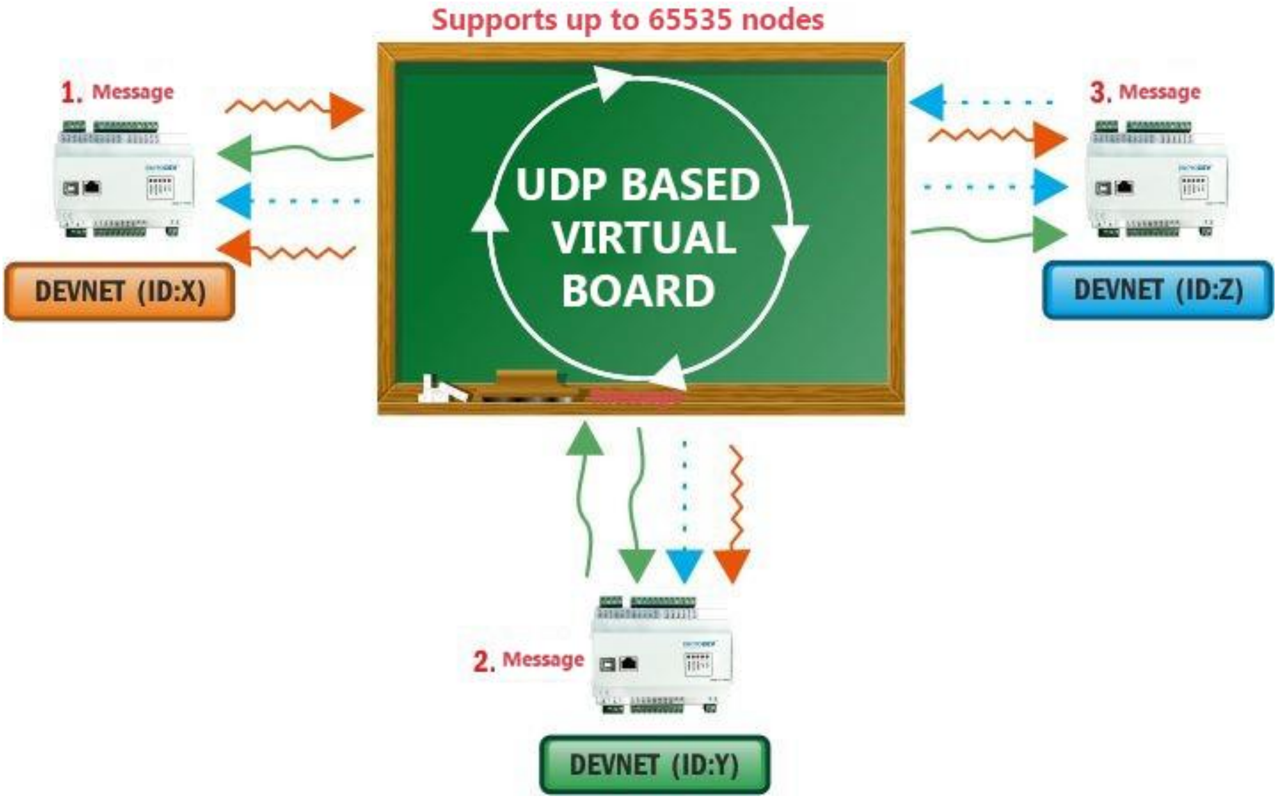
18.4.3 Block Settings



Self DevNET ID: The device's ID
Timeout(ms): Timeout in milliseconds
Cycle Delay (ms): The cycle delay in milliseconds
Waited DevNET ID: Waited DevNET (Connected Device) ID
Destination DevNET ID: Destination DevNET (Device to be connected) ID
Broadcast IP Adress: The IP address to which the devices are connected

18.4.4 Block Explanations

DevNET is a system that reads and writes data from one device selected via ethernet and transfers the data to the DevNET register. This system can be thought of as a circular queue structure.



It is a UDP-based protocol repetitive and cyclical package.

It is a multi-drop protocol and can be added to a single DEV-NET network with up to 65535 PLCs.

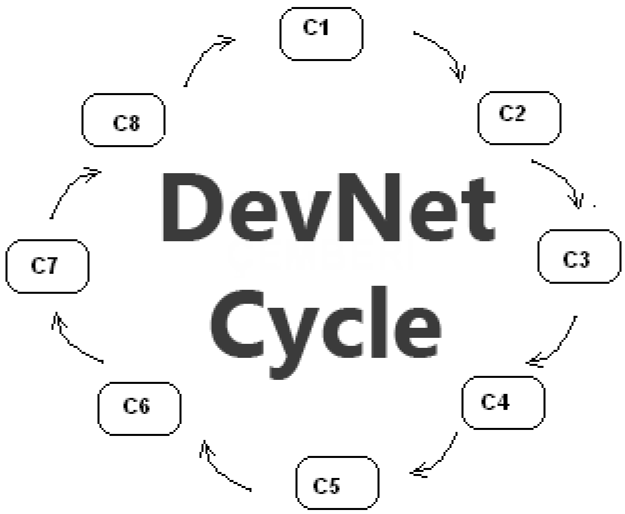
All nodes are on the same level and there is no Master / Slave structure.

The Ethernet port used in the PLC also supports other protocols.

Points are automatically synchronized according to their node addresses and messages they receive.

Thanks to the wait time and timeout mechanisms, network changes can be adaptable. This provides a superior level of robustness.

Excellent compatibility with Mikrodev PLCs is ensured.



C1-C2 ... represents Device 1-Device 2.

Self DevNET Id found in the window is the DevNET Id of the device used.

The timeout period found in the window is the time that one of the devices in this DevNet network is waiting for data from the previous device.

Cycle Delay, located in the window, is the "how long the device will wait for a cycle". A value can be entered by subtracting 1 from the number of devices and multiplying the timeout value by this value.

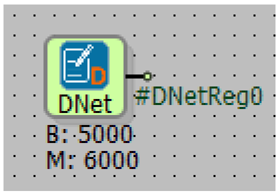
The Waited DevNET Id located in the window is the Id of the device to be read.

The TargetDevNET Id in the window is the Id of the device to which the data will be written.

The "Broadcast IP Address" located in the window is the address of the internet network to which the devices are connected. (An example is 192.168.2.255, where the last 255 is entered to allow access to all devices connected to this network.)

18.5 DEVNET REGISTER

18.5.1 Connection

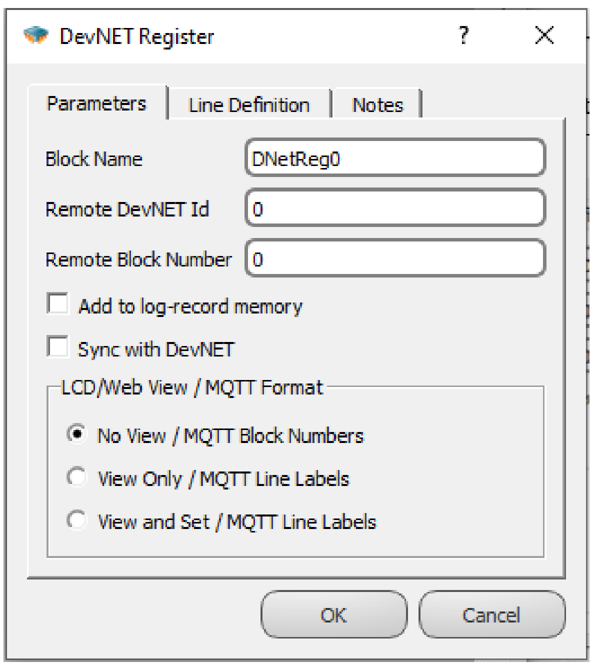
	#DNetReg0: Block connection output

18.5.2 Connection Explanations

Val: Block connection output

It is the block output connection.

18.5.3 Block Settings

	Remote DevNET ID: The Id of the device from which the data will be read
	Remote Block Number: It is the Modbus address of the block at the device from which the data will be read

18.5.4 Block Explanations

The data of the registers in the device connected via the Ethernet network is transferred into this block.

On the remote DevNET Id field of the window, the ID of the device to be read the data is written.

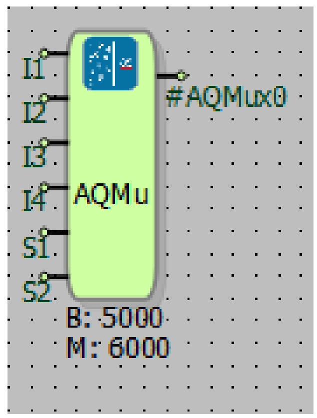
The field which is labeled as Remote Block Number in the window is the Modbus address of the device to be read.

Warning!! To read with this block, "Sync with DevNET" option also must be checked on the other device.

19 MULTIPLEXER BLOCKS

19.1 ANALOG QUART MULTIPLEXER

19.1.1 Connections

I1: Analog quart multiplexer input		#AQMux0: Analog quart multiplexer output
I2: Analog quart multiplexer input		
I3: Analog quart multiplexer input		
I4: Analog quart multiplexer input		
S1: Analog quart multiplexer select input 1		
S2: Analog quart multiplexer select input 2		

19.1.2 Connection Explanations

<u>I1: Analog quart multiplexer input</u>

is an Analog value input that can be written to the output depending on the value of the "S1" and "S2" block inputs. Analog Register block can be connected

<u>I2: Analog quart multiplexer input</u>

It is an Analog value input that can be written to the output depending on the value of the "S1" and "S2" block inputs. Analog Register block can be connected

<u>I3: Analog quart multiplexer input</u>

It is an Analog value input that can be written to the output depending on the value of the "S1" and "S2" block inputs. Analog Register block can be connected

<u>I4: Analog quart multiplexer input</u>

It is an Analog value input that can be written to the output depending on the value of the “S1” and “S2” block inputs. Analog Register block can be connected

S1: Analog quart multiplexer select input 1

It is the input in which which of the “I1”, “I2”, “I3” and “I4” block inputs to output is determined according to the truth table. The Binary Register block can be linked.

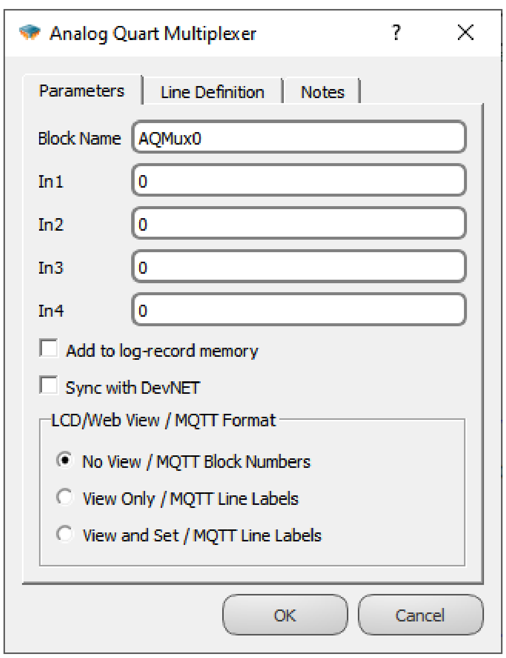
S2: Analog quart multiplexer select input 1

It is the input in which which of the “I1”, “I2”, “I3” and “I4” block inputs to output is determined according to the truth table. The Binary Register block can be linked.

#AQMux0: Analog quart multiplexer output

It is the output connection where one of the “I1”, “I2”, “I3” and “I4 block inputs” is written, which is determined according to the truth table from the “S1” and “S2” block inputs.

19.1.3 Block Settings

	In1: First value can be chosen from inside of the block.
	In2: Second value can be chosen from inside of the block.
	In3:Third value can be chosen from inside of the block.
	In4: Fourth value can be chosen from inside of the block.

19.1.4 Block Explanations

One of the inputs is selected among the four inputs and transferred to the block output. The input which will be transferred to the block output is determined with S1 and S2 selection inputs.

In order to transfer the I1 input to the block output; S1:must be logic(0), S2:must be logic(0)

In order to transfer the I2 input to the block output; S1:must be logic(1), S2: must be logic(0)

In order to transfer the I3 input to the block output; S1:must be logic(0), S2: must be logic(1)

In order to transfer the I4 input to the block output; S1:must be logic(1), S2:must be logic(1)

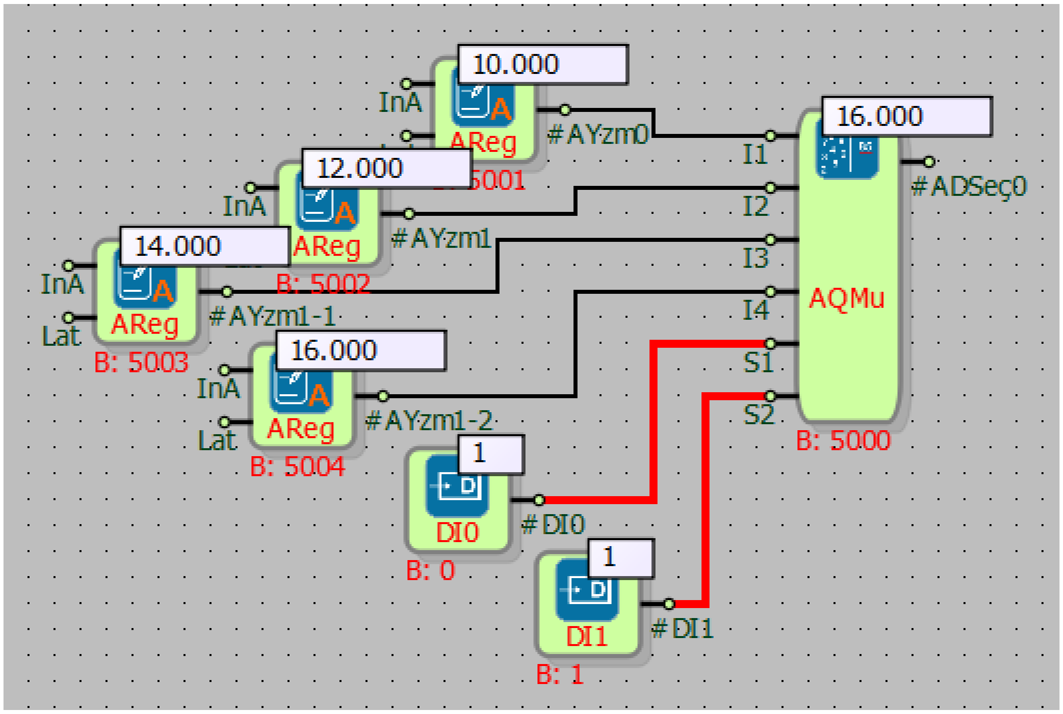
The input value is transfered to the block output as a 32 bit analog value

19.1.4.1 Truth Table

According to the “S1” and “S2” inputs of the Analog Quad Selector block, which input will be written to the output is specified in the following truth table.

S1	S2	#AQMux0
1	1	I4
0	1	I3
1	0	I2
0	0	I1

19.1.5 Sample Application

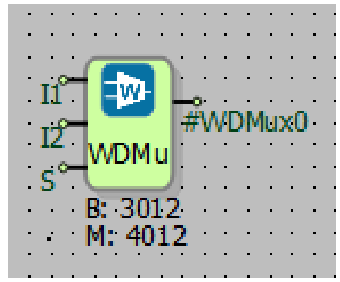


In the example;

According to logic states of the Analog Quart Multiplexer's selection inputs (S) , the values in the inputs and O1 output are showed. In the example I4 is selected by setting both S1 and S2 to logic(1). (For logic (0) to the S choosing input shold be 0; for logic (1) any value which is different from zero is valid.

19.2 WORD DUAL MULTIPLEXER

19.2.1 Connections

<table>
<tr><td>I1: Word dual multiplexer input</td><td rowspan="3">
</td><td rowspan="3">#WDMux0: Word dual multiplexer output</td></tr>
<tr><td>I2: Word dual multiplexer input</td></tr>
<tr><td>S: Word dual multiplexer selection input</td></tr>
</table>

19.2.2 Connection Explanations

<u>I1: Word dual multiplexer input</u>

Depending on the value of the “S” block input, it is the word value input that is likely to be written to the output. Word Writer block can be connected.

<u>I2: Word dual multiplexer input</u>

Depending on the value of the “S” block input, it is the word value input that is likely to be written to the output. Word Writer block can be connected.

<u>S: Word dual multiplexer selection input</u>

It is the input that determines which of the “I1” or “I2” block inputs will be output, according to the truth table. Binary Register block can be connected.

<u>#WDMux0: Word dual multiplexer output</u>

It is the block output that writes one of the word register block values connected to the "I1" or "I2" block input according to the truth table in line with the value of the "S" block input.

19.2.3 Block Setting

There are no block settings.

19.2.4 Block Explanation

One of the inputs is selected among the two inputs and transferred to the block output. The input which will be transferred to the block output is determined with S selection input

In order to transfer the I1 input to the block output; S:must be logic(0)

In order to transfer the I2 input to the block output; S:must be logic(1)

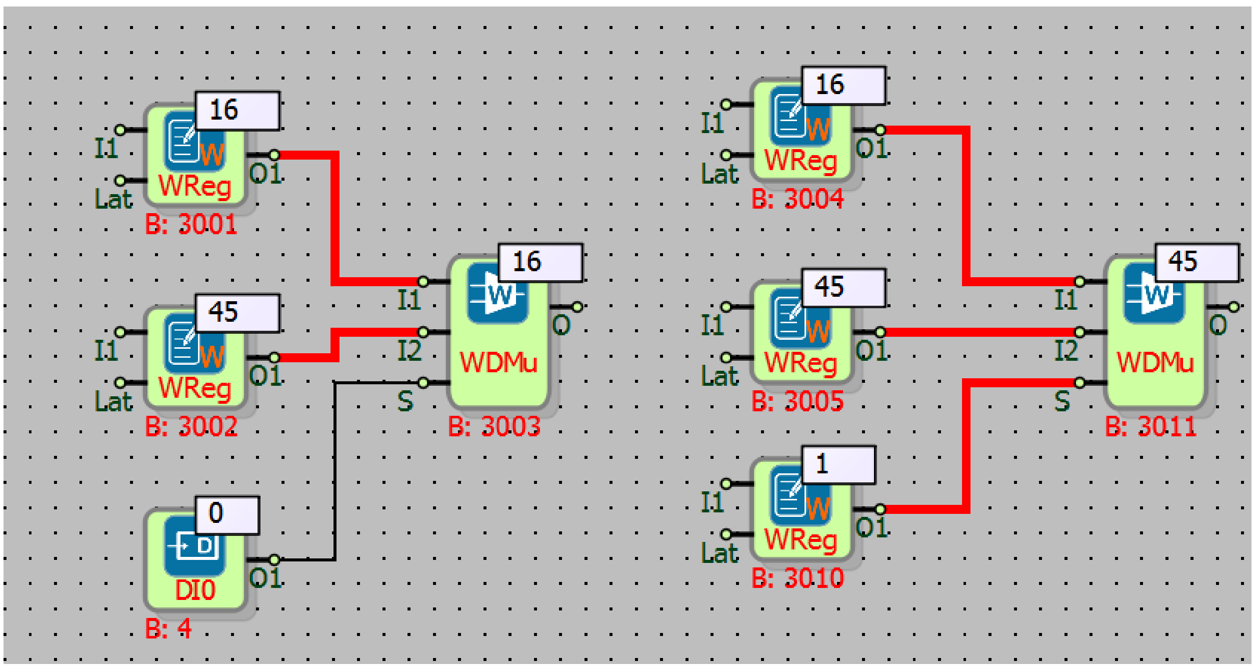
The input value is transfered to the block output as a 16 bit word value

19.2.4.1 Truth Table

According to the “S” block input value of the Word Binary Selector block, which input will be written to the output is specified in the truth table below.

S	#WDMux0
0	I1
1	I2

19.2.5 Sample Application

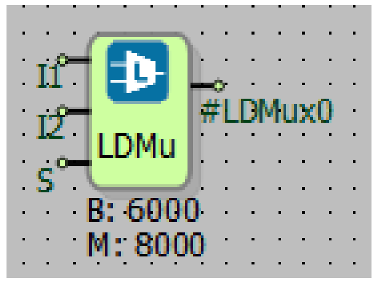


In the example;

According to logic states of the Word Dual Multiplexer’s selection input (S) , the values in the inputs and O1 output are showed. In the example different inputs are selected by setting both S to logic(1) or logic(0). (For logic (0) to the S choosing input shold be 0; for logic (1) any value which is different from zero is valid.

19.3 LONG DUAL MULTIPLEXER

19.3.1 Connections

I1: It is input which is long dual multiplexer.		#LDMux0: It is output which is long dual multiplexer
I2: It is input which is long dual multiplexer.		
S: It is input which is long dual multiplexer choice input		

19.3.2 Connection Explanations

<u>I1: It is input which is long dual multiplexer</u>

It is the Long value input that is likely to be written to the output depending on the value of the "S" input. Long Register block can be connected.

<u>I2: It is input which is long dual multiplexer</u>

It is the Long value input that is likely to be written to the output depending on the value of the "S" input. Long Register block can be connected.

<u>S: It is input which is long dual multiplexer choice input</u>

It is the input that determines which of the "I1" or "I2" block input values will be given to the output, according to the truth table. Binary Register block can be connected.

<u>#LDMux0: It is output which is long dual multiplexer</u>

It is the block output that writes one of the block values of the Long Register connected to the "I1" or "I2" block input according to the truth table in line with the value of the "S" block input.

19.3.3 Block Settings

There is no block settings.

19.3.4 Block Explanation

One of the inputs is selected among the two inputs and transferred to the block output. The input which will be transferred to the block output is determined with S selection input

In order to transfer the I1 input to the block output; S:must be logic(0)

In order to transfer the I2 input to the block output; S:must be logic(1)

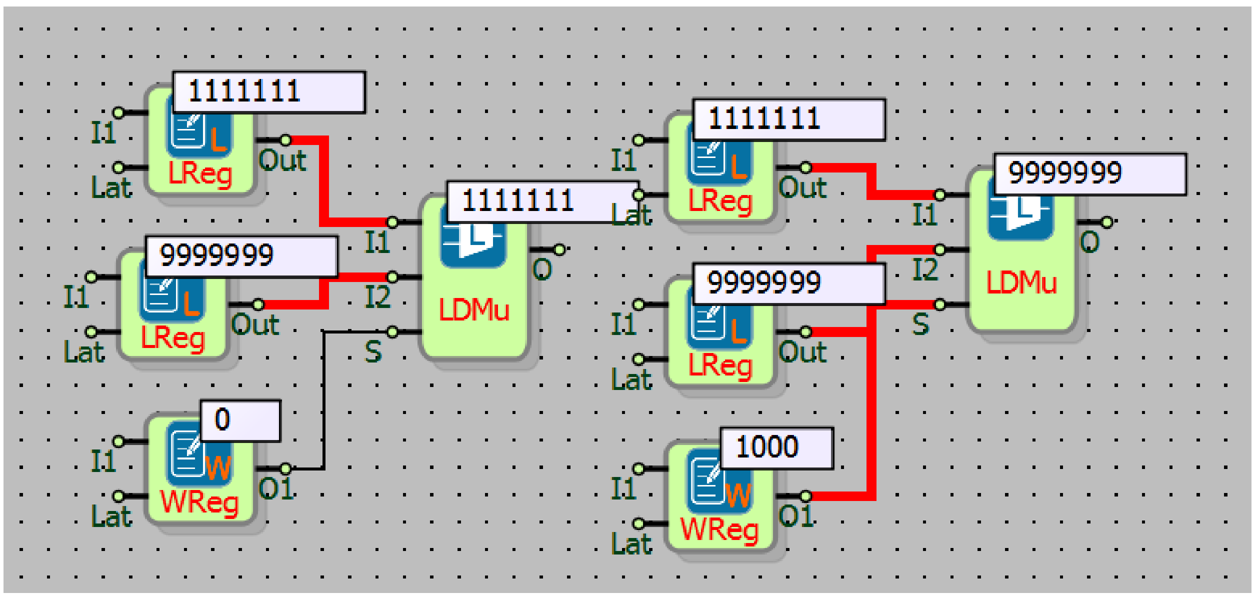
The input value is transfered to the block output as a 32 bit long value

19.3.4.1 Truth Table

According to the “S” block input value of the Lord Binary Selector block, which input will be written to the output is specified in the truth table below.

S	#LDMux0
0	I1
1	I2

19.3.5 Sample Application

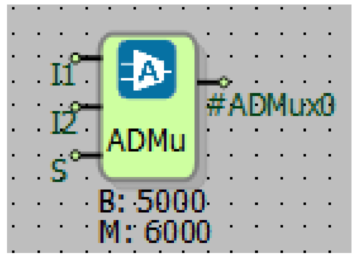


In the example ;

According to logic states of the Long Dual Multiplexer’s selection input (S) , the values in the inputs and O1 output are showed. In the example different inputs are selected by setting both S to logic(1) or logic(0). (For logic (0) to the S choosing input shold be 0; for logic (1) any value which is different from zero is valid.

19.4 ANALOG Dual Multiplexer

19.4.1 Connections

<table>
<tr><td>I1: Analog dual multiplexer input</td><td rowspan="3">
</td><td rowspan="3">#ADMux0: Analog dual multiplexer output</td></tr>
<tr><td>I2: Analog dual multiplexer input</td></tr>
<tr><td>S: Analog dual multiplexer selection input</td></tr>
</table>

19.4.2 Connection Explanations

<u>I1: Analog dual multiplexer input</u>

It is an Analog value input that can be written to the output depending on the value of the “S” input. Analog Register block can be connected.

<u>I2: Analog dual multiplexer input</u>

It is an Analog value input that can be written to the output depending on the value of the “S” input. Analog Register block can be connected.

<u>S: Analog dual multiplexer selection input</u>

It is the input that determines which of the “I1” or “I2” block input values will be given to the output, according to the truth table. Binary Register block can be connected.

<u>#ADMux0: Analog dual multiplexer output</u>

It is the output of the analog dual multiplexer block which is 32 bit. It is the block output that writes one of the Analog Register block values connected to the “I1” or “I2” block input according to the truth table in line with the value of the “S” block input.

19.4.3 Block Settings

There are no block settings.

19.4.4 Block Explanation

One of the inputs is selected among the two inputs and transferred to the block output. The input which will be transferred to the block output is determined with S selection input

In order to transfer the I1 input to the block output; S:must be logic(0)

In order to transfer the I2 input to the block output; S:must be logic(1)

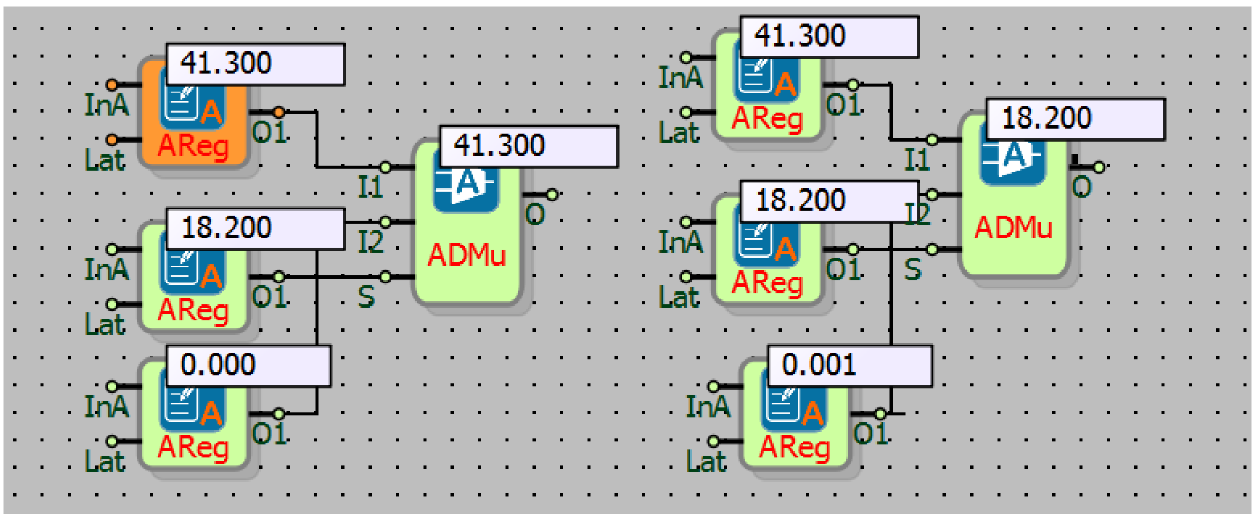
The input value is transfered to the block output as a 32 bit analog value

19.4.4.1 Truth Table

According to the “S” block input value of the Analog Dual Multiplexer block, which input will be written to the output is specified in the truth table below.

S	#ADMux0
0	I1
1	I2

19.4.5 Sample Application



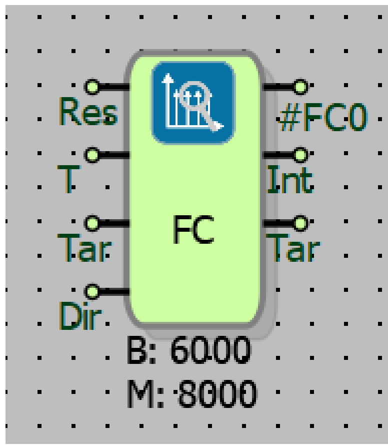
In the example;

According to logic states of the Analog Dual Multiplexer’s selection input (S) , the values in the inputs and O1 output are showed. In the example different inputs are selected by setting both S to logic(1) or logic(0). (For logic (0) to the S choosing input shold be 0; for logic (1) any value which is different from zero is valid.

20 MOTION CONTROL BLOCKS

20.1 FAST COUNTER INPUT

20.1.1 Connections

<table>
<tr><td>Res: Reset input</td><td rowspan="4">
</td><td rowspan="1">#FC0: Total pulse count</td></tr>
<tr><td>T: Time Period for Frequency Calculation</td><td rowspan="2">Int: Pulse count within a period cycle</td></tr>
<tr><td>Tar: Target pulse count</td></tr>
<tr><td>Dir: Counting direction Up or Down</td><td>Tar: Pulse number target reached</td></tr>
</table>

20.1.2 Connection Explanations

Res: Reset input

Reset the total pulse count kept inside the block.

T: Period input

Specifies the period in units of milliseconds.

Tar: Target pulse count input

Specifies the target pulse count

Dir: Count direction input

“1: Up, 0: Down. Specifies the counting direction.

#FC0: Total pulse count

Total pulse count after Reset

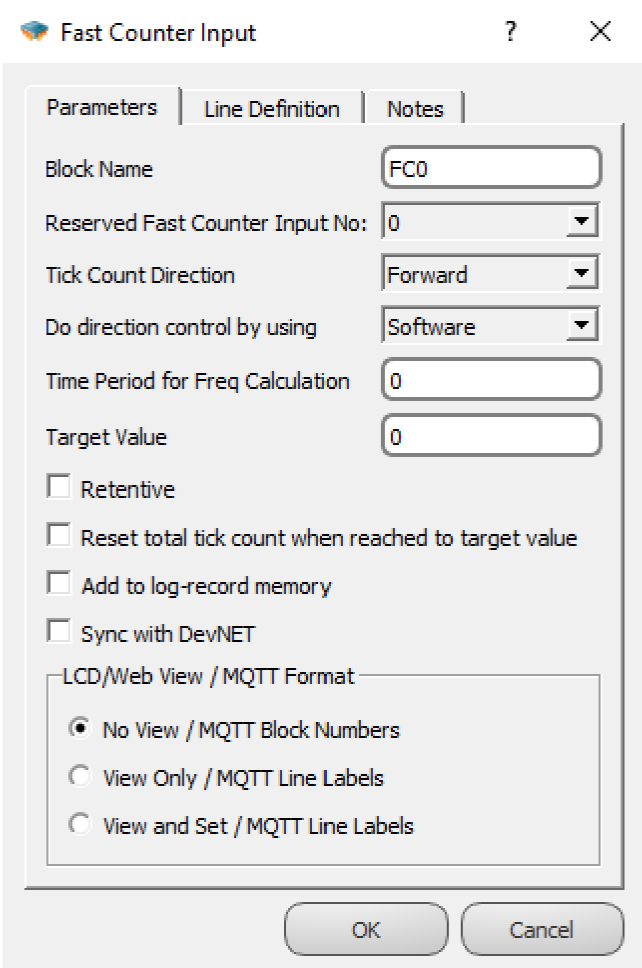
Int: Pulse count in a period cycle

Output that gives the counted pulses within a period cycle.

Tar: Target Value Reached

Indicates whether the target value is reached or not.

20.1.3 Block Settings



Reserved Fast Counter Input No: Specifies which fast counter channel will be used by the block.
Tick Count Direction: Specifies the counting direction.
Do direction control by using: Specifies whether hardware or software is controlling the counting direction. If Hardware is selected, fast counter channel work as encoder input.
Time Period: Millisecond based time period for frequency calculation
Target Value: Target count value is entered here
Reset total tick count when reached to target value: Makes the block reset when the count reaches to the target value.

20.1.4 Block Explanation

Fast counter blocks are used to count digital pulse input signals. Fast Counter Blocks are different from other up/down counters because of using of hardware fast counter channels. Therefore, Fast counter blocks can be able to count much more faster signal than software counter.

First output of the block (#FC0) indicates the total ticks counted. This value is reset when the device is reset or when a rising edge is detected on reset input of the block. It counts up or down with each incoming pulse signal.

“T” value indicates the time period which the pulses are counted in. Number of pulse count in defined period “T” is written to “Int” output of the block. “T” has the units of milliseconds. For example, if T=1000 (means 1000 millisecond), than the “Int” output of the block will show he frequency of the input digital signal connected to related fast counter input.

Note: If the period value of the “T” block input time is 0, the total number value is read at the second output.

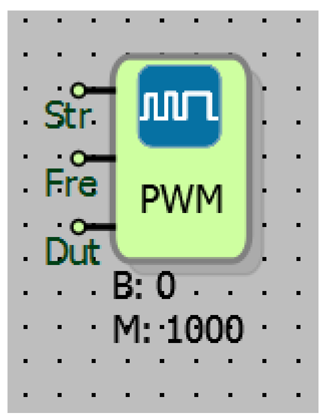
With “Target Val” input, a target value is specified and when the target value is reached a pulse is sent out from the third output (“Target Reached”) of the block. If the “Target Val” input is equal to 0, then the mechanism is disabled. If the “reset when target value is reached” option is activated, total count and the “Target Reached” output of the block is reset when the target value is reached. A target value can be specified only when counting up.

If a high signal is applied to the “Res” input of the block, total count value will be reset.

Fast counters counts in the signed 32-bit format. Count value can vary between the values -2147483648 and 2147483648.

20.2 PULSE WIDTH MODULATION (PWM)

20.2.1 Connections

Str: Start / Stop input	
Fre: Frequency input	
Dut: Duty Cycle input	

20.2.2 Connection Explanations

<u>Str: Start/stop input</u>

Input for Start/Stop signal. If it is 0, PWM signal is shut down and related PWM channel becomes a normal digital output. If it is 1, PWM signal is activated.

<u>Fre: Frequency input</u>

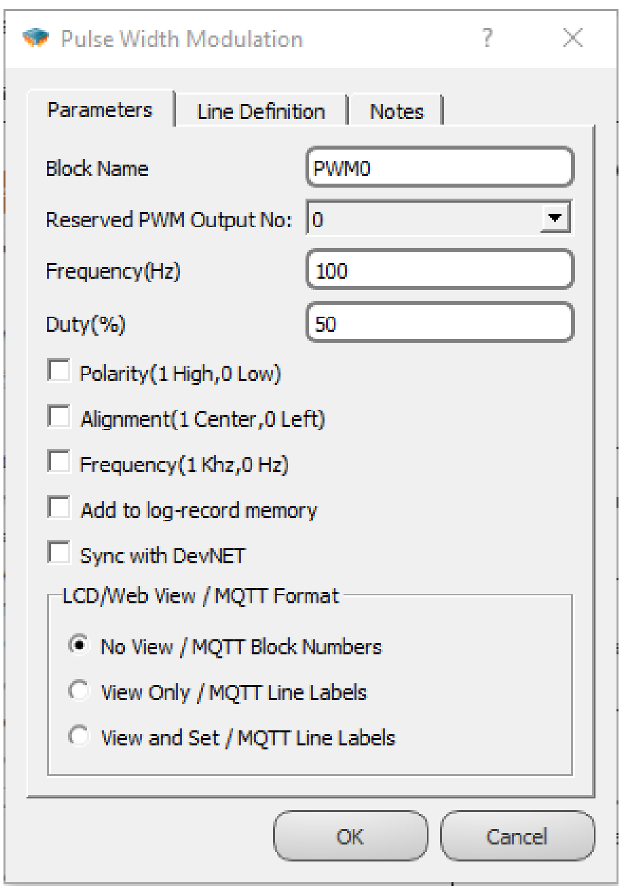
Input for the frequency. Any value between 2Hz – 60000Hz(60 kHz) can be entered.

<u>Dut: Duty cycle input</u>

Input for duty cycle. Duty cycle percentage is entered as an number between 0-100.

Note: In order to activate the block, the “AT+PT0=1” command should be sent from the Mikroterminal and reset to the device to be used

20.2.3 Block Settings

	Reserved PWM Output No: If 0, DQ0 channel is used as PWM If 1 DQ1 channel is used as PWM If 2 DQ2 channel is used as PWM If 3 DQ3 channel is used as PWM
	Frequency(Hz): Frequency is specified here.
	Duty(%): A percentage value is entered here.
	Can be used with default Polarity, Alignment and Frequency settings.

20.2.4 Block Explanation

PWM block is used to control the PWM outputs of the device.

“Dut” input of the block specifies the duration of the high and low parts of the signal with a specified frequency. According to the “Duty” value determined from the PWM block input or block properties, it sets how many percent of the pulse width at the desired frequency should be a high signal; how many percent should be a low signal.

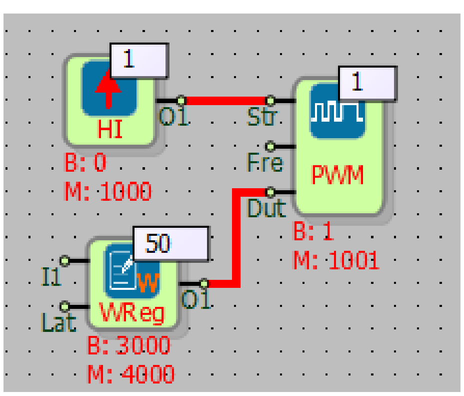
“Str” input of the block stands for “Start/Stop”. When a high signal applied to the “Str” input, PWM Block is activated and starts to generate PWM signal. When a low signal is applied to the “Str” input, the block is deactivated and PWM output serves as a normal digital output. If this input is low-level (0), the PWM will be passive, and the corresponding PWM output will serve as the normal digital output.

“Fre” input is used to change the frequency externally. It can be left blank and can be set in the Block Settings menu. Since all the PWM channels in the device are using the same timing source, frequency value is the same for all the PWM channels. Whichever block’s frequency is changed most recently, all the other blocks will have the same frequency.

“Dut” input of the block can be set externally or can be set in Block Settings. Different duty cycle values can be assigned to the different blocks, independent from each other.

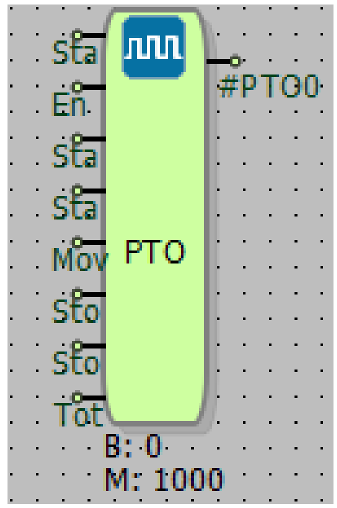
Block serves as a PWM signal generator when the PWM channel is active, and serves as a normal digital output when the PWM channel is passive.

20.2.5 Sample Application



20.3 PULSE TRAIN OUTPUT

20.3.1 Connections

<table>
<tr><td>Sta: Start input</td><td rowspan="8">
</td><td rowspan="8">#PTO0: Signal output</td></tr>
<tr><td>En: Enable input</td></tr>
<tr><td>Sta: Start speed input</td></tr>
<tr><td>Sta: Start pulse number input</td></tr>
<tr><td>Mov: Operating speed input</td></tr>
<tr><td>Sto Stop speed input</td></tr>
<tr><td>Sto: Stop pulse number input</td></tr>
<tr><td>Tot: Total pulse number input</td></tr>
</table>

20.3.2 Connection Explanations

Sta: Start input

It is the input for starting PTO. It is the trigger input to start the pulse train output according to the updated settings.

En: Enable input

It is the input for activation PTO. The PTO can also be used as an emergency stop input, the pulse train stops when this input is 0.

Sta: Start Speed input

It is input to set starting speed. “Start Speed” specifies starting speed of the PTO during acceleration phase. The value entered here is width of the pulse and considered as 10 μs multiplier.

Sta: Start pulse count input

It is the input to set starting pulse count. Specifies the number of pulses in acceleration phase. PTO automatically performs acceleration using start speed, normal speed and start pulse count parameters.

Mov: Operating speed input

It is an operating speed input that specifies normal operating speed. The value entered here is the width of the pulse and considered as 10 μs multiplier.

Sto: Stop speed input

It is the input for stop speed. If there is going to be a deceleration during the stop process, the speed just before the stop moment must be entered. The value entered here is the width of the pulse and considered as 10 μs multiplier.

Tot: Total pulse input

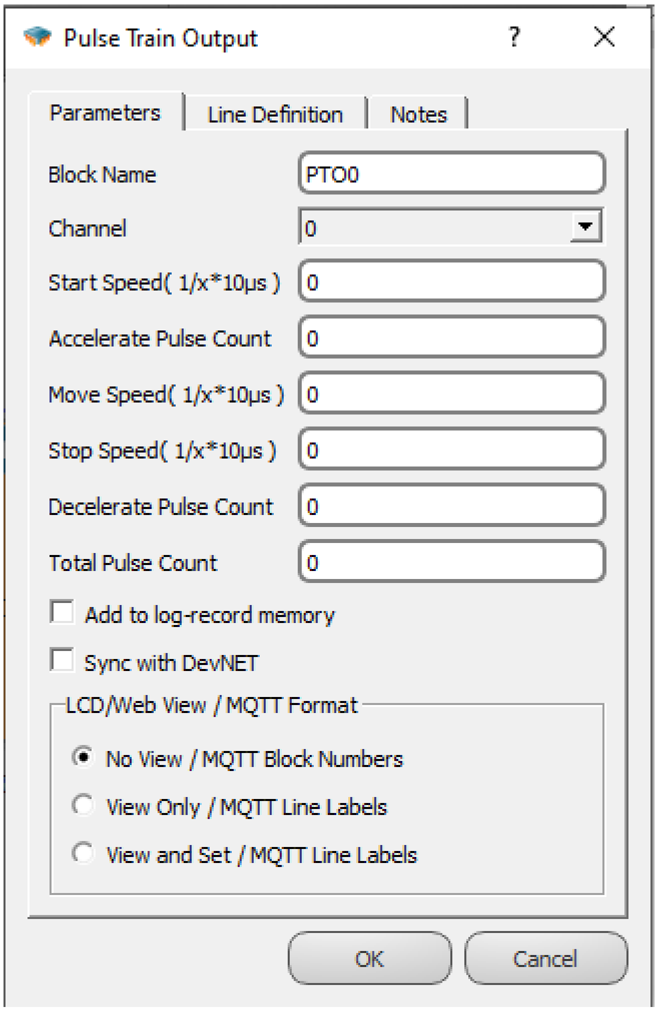
It is the input for total pulse number that specifies the total number of pulses, including the acceleration, deceleration and stop processes.

#PTO0: Signal output

It is the output for the signal. PTO generates high output after total pulse count reached.

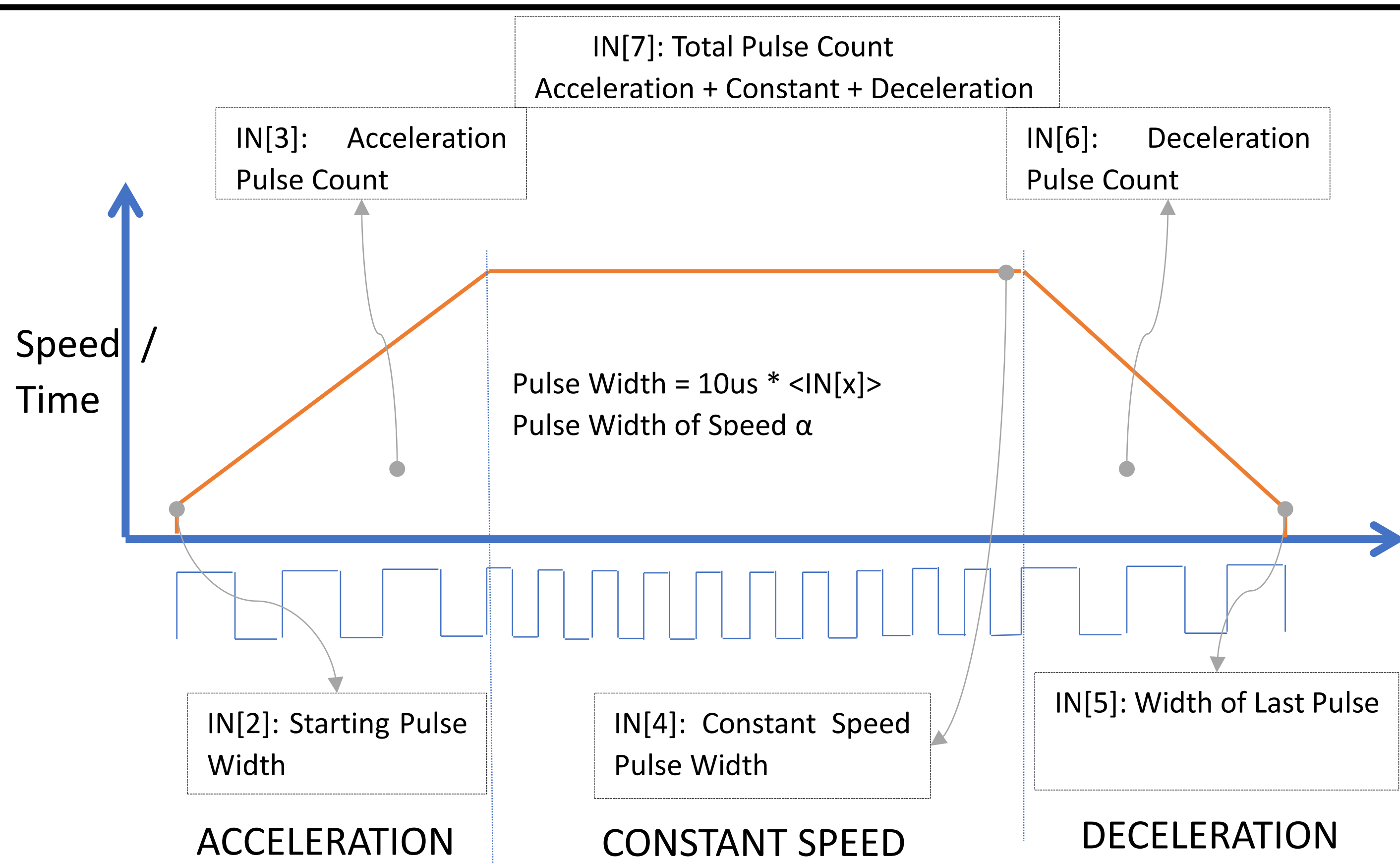
Note: In order to activate the block, the command “AT+PT0=1” must be sent from the Mikroterminal and the device to be used must be reset.

20.3.3 Block Settings

	Reserved PTO Output No If 0, DQ0 channel is used as PTO If 1 DQ1 channel is used as PTO If 2 DQ2 channel is used as PTO If 3 DQ3 channel is used as PTO
	Start Speed: Start speed to begin acceleration
	Accelerate Pulse Count: Number of pulses during the acceleration process
	Move Speed: Normal operating speed
	Stop Speed: In deceleration process, the speed just before the moment PTO stops.
	Decelerate Pulse Count: Number of pulses during the deceleration process.
	Total Pulse Count: Total number of pulses

20.3.4 Block Explanation

It is used to make a controlled step input motion by sending a certain number of pulses. Acceleration, constant motion and deceleration functions are performed by controlling the pulse width and the number of pulses.



If Acceleration and Deceleration functions are not required, corresponding pulse count parameters must be set to 0.

Total pulse count parameter is sum of pulse count of Accelaration, Constant Speed, Deceleration phases.

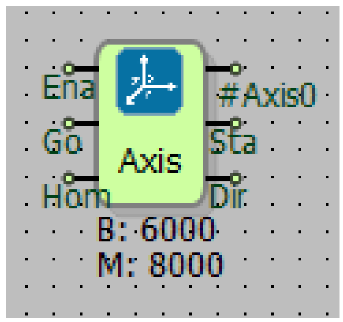
Acceleration phase; Acceleration is start from "Initial pulse width" to "Constant speed pulse width" and acceleration speed is controlled by acceleration pulse count parameter. The same applies to the deceleration cycle.

The value entered as pulse width information is evaluated as 10 µs on the device side. For example, for input pulse width of 120 us, input must be written 12.

After the PTO block starts to work, the output signal of the block goes to Logic(0) . After the total number of pulses is completed, the "#PTO0" output goes to Logic(1). Connecting to “#PTO0” output to another PTO block inputs result in controlling multiple PTO blocks together.

20.4 AXIS DEFINITON

20.4.1 Connections

Ena: Enable input of the block		#Axis0: Absolute position at axis
Go: Go home command input		Sta: Status
Hom: Home indicator		Dir: Direction

20.4.2 Connection Explanations

Ena: Enable input of the block

It can be used as an emergency stop or to enable axis movement.

Go: Go home command input

Homing command input

Hom: Home indicator

It reads whether it is in HOME position through this input.

#Axis0: Absolute Position

It is the output of the block that gives the absolute position at the axis

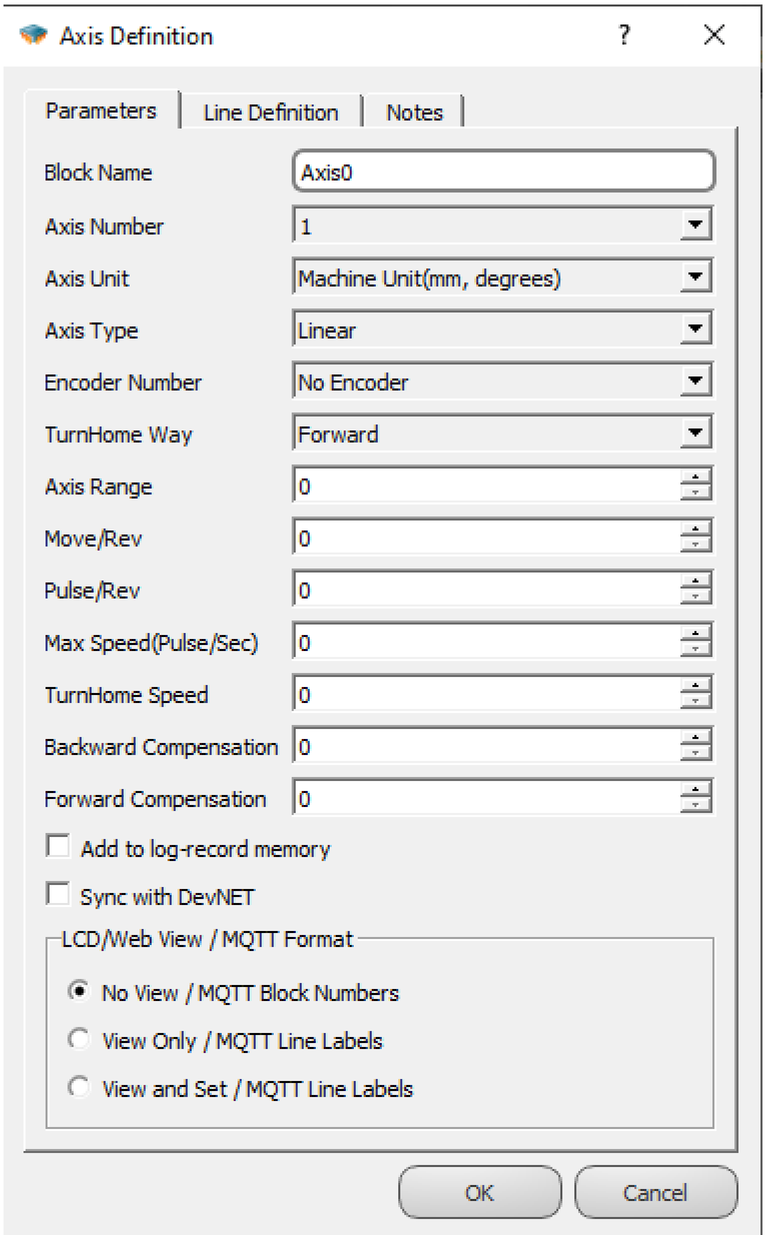
Sta: Binary output

It is the output of the block that indicates the state of the motor.

Dir: Binary output

It is the output of the block that indicates the direction of the motor.

20.4.3 Block Settings



Axis Number: Specifies the output which the axis block will be using
Axis Unit: Machine Unit: Moves in units of millimeters. Motor Unit: Moves in units of pulses.
Axis Type: Specifies how to reach the deisred position.
Encoder Number: Specifies the encoder number.
TurnHome Way: Specifies the direction when going to home position.
Axis Range: Specifies the required number of pulses during one complete lap of motor.
Move/Rev: Specifies the speed when in Machine unit mode.
Pulse/Rev: Specifies the speed when in Motor Unit mode.
Max Speed(Pulse/Second): Specifies the maximum speed when operating.
TurnHome Speed: Specifies the speed when motor is returning to home position.
Bacward Compensation: Ramp amount when moving backwards.
Forward Compensation: Ramp amount when moving forward.

20.4.4 Block Explanation

This block is used to control the position of the system on the axis. The block keeps the last position of the system and using this info control the Pulse Train Outputs to realize position aware movement.

When a Logic(1) signal is applied to “Go Home” input of the block, PLC starts to drive the motion system until Logic(1) signal appears at “Home” input of the block. If homing process is started while it is already at home position, PLC starts homing process to calculate axis length per pulse count.

“Axis Number” specifies PTO channel of the Axis block.

- If Axis Number is 1, output will be at PTO channel 0 - DQ0
- If Axis Number is 2, output will be at PTO channel 1 – DQ1
- If Axis Number is 3, output will be at PTO channel 2 – DQ2
- If Axis Number is 4, output will be at PTO channel 3 – DQ3

If an axis block is used in the project, PTO blocks cannot be used anymore for this channel.

To use the axis block, AT+PTO=1 command must be sent to the device using Mikroterminal to make digital outputs pulse outputs. DO0, DO1, DO2, DO3 cannot be used for any other purposes anymore.

Axis and machine moves in units of millimeters: Machine Unit: Moves in units of millimeters. Motor unit moves in the units of pulses.

Axis type: Determines how the motor reaches the desired position. If “circular” is selected, desired position will be reached by the shortest path. If “linear” is selected, when the starting point is reached, motor starts to move at the opposite direction, and then reaches to the desired position. This way, if there is a cable attached to the motor it will not be damaged.

Turn home way: specifies the direction of the motor when going back to starting point.

Axis range: Required pulse amount for a lap is entered here.

Move/Rev: In Machine Unit option, defines the speed. Speed of the motor is reversely proportional with the number entered here. It has the units of milliseconds.

MaxSpeed(Pulse,sec): Defines the maximum speed of the motor when operating. The number is reversely proportional with the speed of the motor. It has units of microseconds.

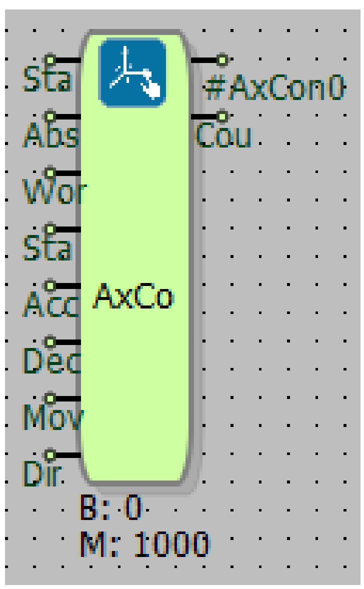
Turn Home Speed: Defines the speed when going to starting point. It is reversely proportional with the speed of the motor. It has units of microseconds.

Backward Compensation: Compensation value for turning in reverse direction.

Forward Compensation: Compensation value for turning in forward direction.

20.5 AXIS CONTROL

20.5.1 Connections

Sta: Start command		#AxCon0: Pulse output
Abs: Target Position		
Wor: Input for motor speed		
Sta: Input for starting speed.		
Acc: Input for acceleration duration		Cou: Output of the block
Dec: Input for deceleration duration		
Mov: Target position input		
Dir: Direction Input		

20.5.2 Connection Explanations

Sta: Start command binary input:

When a signal is applied to this input, block start to drive system.

Abs: Target position

Determines the target position.

Wor: Input for motor speed:

Motor speed is setting by connecting a word register to this input.

Sta: Input for starting speed:

Specifies the ramp speed before reaching to the target input.

Acc: Input for acceleration duration:

Specifies the duration of acceleration of the motor.

Dec: Input for deceleration duration:

Specifies the duration of deceleration of the motor.

Mov: Target position input:

Specifies the target position, using word or long register.

Dir: Direction input:

Specifies the direction of the motion. 1: forward, 0: reverse.

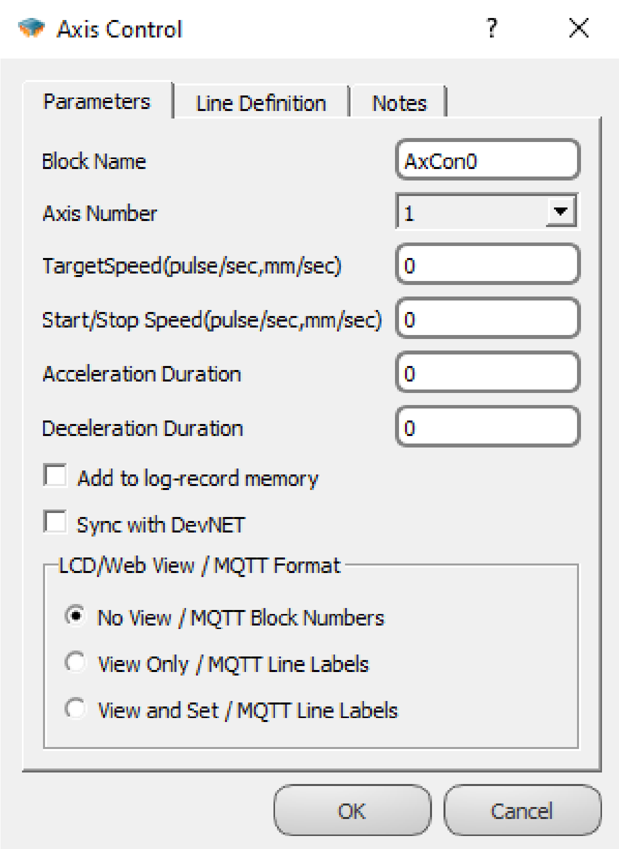
#AxCon0: Pulse output:

When the block produces a pulse, this output generates a momentary signal.

Cou: Binary output

Gives a binary output.

20.5.3 Block Settings

<table>
<tr><td rowspan="5">
</td><td>Axis Number: Specifies the axis number.</td></tr>
<tr><td>TargetSpeed(pulse/sec,mm/sec): Specifies the target speed.</td></tr>
<tr><td>Start/Stop Speed(pulse/sec,mm/sec): Specifies the start and stop speed of the motor.</td></tr>
<tr><td>Acceleration Duration: Specifies the acceleration duration.</td></tr>
<tr><td>Deceleration Duration: Specifies the deceleration duration.</td></tr>
</table>

20.5.4 Block Explanation

It is used to stabilize the position of the servo motor.

Sta: When the signal is applied to the “Sta” input, block starts to send out pulses.

Wor: A word register block is connected to this input to adjust the motor speed. It can be set in the Block Settings menu either. It is reversely proportional to the speed of the motor. It can be set in the Block Settings menu. It has units of microseconds.

Sta: Specifies the ramp speed before reaching the target speed. It is used when accelerating and decelerating. It can be set in the Block Settings menu. It has units of microseconds.

Acc: Specifies the acceleration time of the motor. Desired value can be entered in Block Settings menu either.

Dec: Specifies the deceleration time of the motor. Desired value can be entered in Block Settings menu either.

Mov: Specifies the target position. By connecting a word or long register required pulse count is indicated.

Dir: Specifies the direction of the movement. 1 means forward, 0 means reverse direction.

#AxCon0: It is a pulse output. When block produces a pulse, gives a pulse signal.

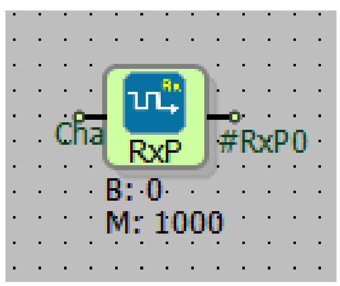
Cou: Gives a binary output.

“Axis Number” specifies the output of the Axis Control Block.

21 SERIAL COMMUNICATION BLOCKS

21.1 Rx Packet

21.1.1 Connections

Cha: Channel Input		#RxP0: Block Output

21.1.2 Connection Explanations

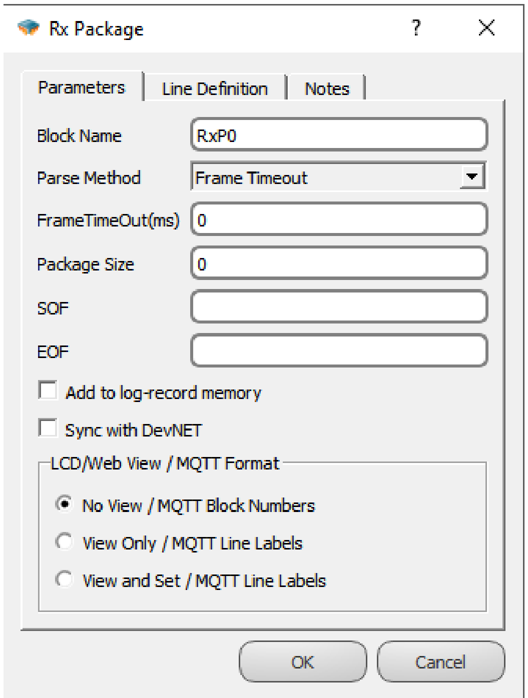
Cha: Channel Input

It is the channel input connection.

#RxP0: Block Output

The block number is the output connection.

21.1.3 Block Settings

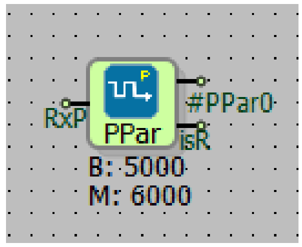
<table>
<tr><td rowspan="5">
</td><td>Parse Method: Evaluates the incoming data according to the exception set in the tab.</td></tr>
<tr><td>FrameTimeOut(ms): It reads the incoming data packet within the ms value specified here.</td></tr>
<tr><td>Paket Boyutu: It determines the size of the incoming data packet in bytes.</td></tr>
<tr><td>SOF: This is the field where the starting character of the incoming data packet is entered.</td></tr>
<tr><td>EOF: This is the field where the ending character of the incoming data packet is entered.</td></tr>
</table>

21.1.4 Block Explanation

Rx Packet block is used to define the incoming data. It checks whether the incoming data is in accordance with the rules determined in the block special settings. If appropriate, it sends it to the Packet Parser block. If it is not appropriate, it does not evaluate the incoming data.

21.2 Packet Parser

21.2.1 Connections

<table>
<tr><td rowspan="2">RxP: Rx packet input</td><td rowspan="2">
</td><td>#PPar0: Parsed result output</td></tr>
<tr><td>isR: Result valid output</td></tr>
</table>

21.2.2 Connection Explanations

<u>RxP: Rx packet input</u>

The output of the Rx Packet block is connected to the “RxP” input of the Packet Parser block.

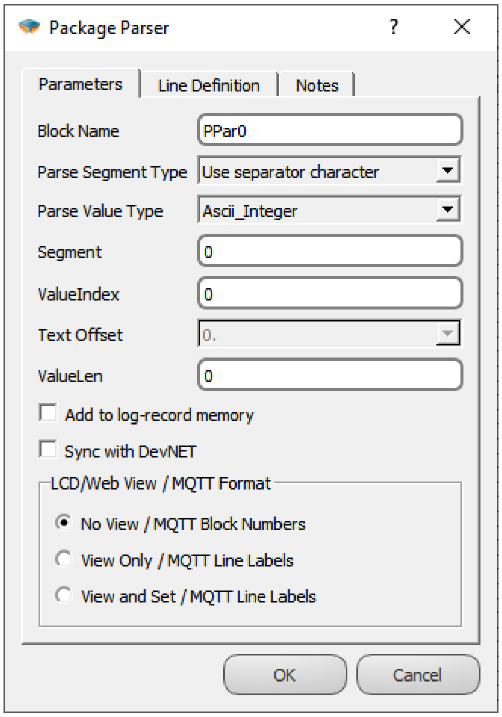
<u>#PPar0: Parsed result output</u>

Parceled data value is output

<u>isR: Result valid output</u>

Output that generates 1 rising edge trigger at the end of each successful plot.

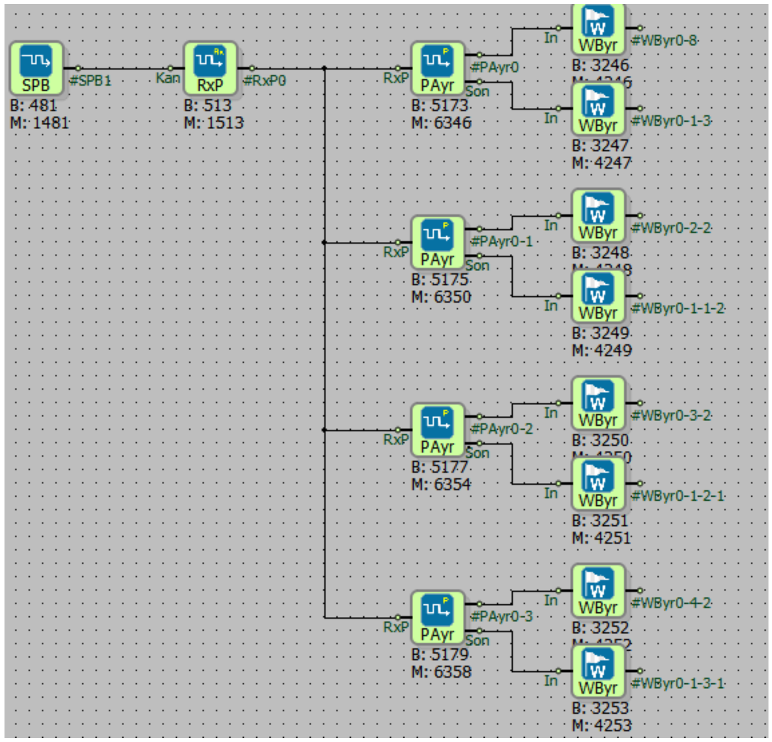
21.2.3 Block Settings

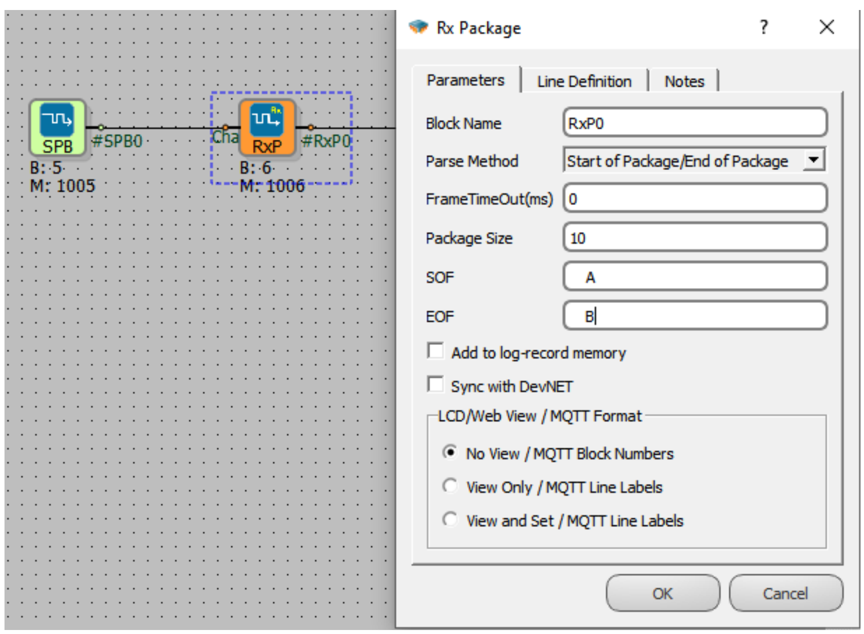
	Parse Segment Type: How to split the incoming data packet is selected under this tab.
	Parse Value Type: The value type of the data to be parsed.
	Segment:
	ValueIndex: After which index the incoming data packet should be separated is entered here.
	Text Offset:
	ValueLen: After which index the incoming data packet should be separated is entered here.

21.2.4 Block Explanation

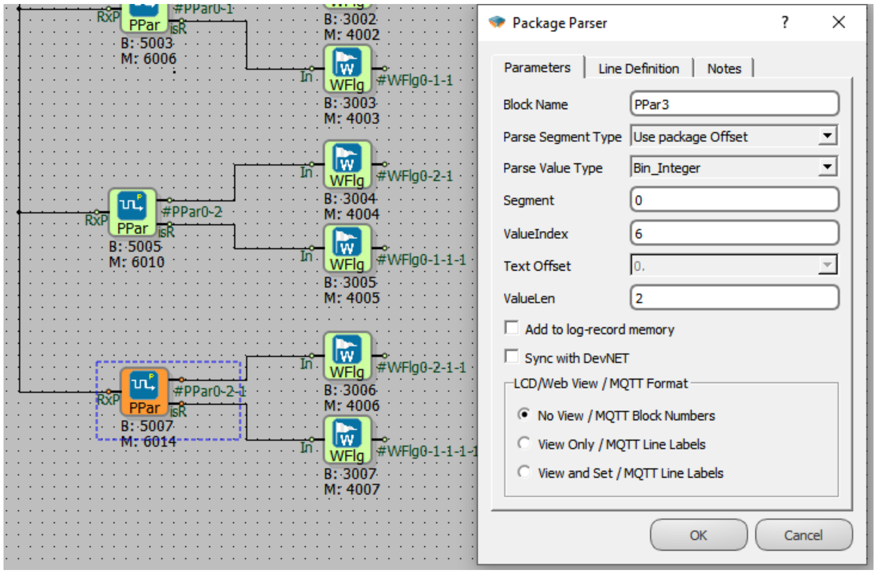
It is used to pars the incoming data. Data is transmitted in packets between devices. In order to turn these data packets into usable information, these packets need to be decomposed. The packet parser block divides the incoming data packets into parts according to the rules we have determined from the block properties.

21.2.5 Sample Application





For the example, we have chosen the decoding method of the Rx packet block as "Beginning/End of Packet". The beginning-of-package character is A and the end-of-package character is B. Index ranges are determined by connecting a packet parser block to its output.



For example, when 41 01 00 02 00 FF 00 FF 00 42 is sent in hex base, the characters at the beginning of the packet and the end of the packet match (41 : A , 42:B) and the data packets are formed if the 8 bytes are in between.

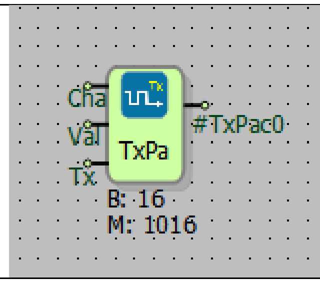
For 01 00 first data, the value 1 appears at the output of the block.

01 00 --> 00 01 becomes 1 when converted from hex to decimal.

FF 00 --> 00FF hex to decimals becomes 255.

21.3 Tx Packet

21.3.1 Connections

Cha: Block connection input		#TxPAc0: Block output
Val: Block value input		
Tx: Trigger input		

21.3.2 Connection Explanations

<u>Cha: Block connection input</u>

Serial port is the input connection to which the block output is connected.

<u>Val: Block value input</u>

The values that we send from this input form the data to be included in the data package to be sent.

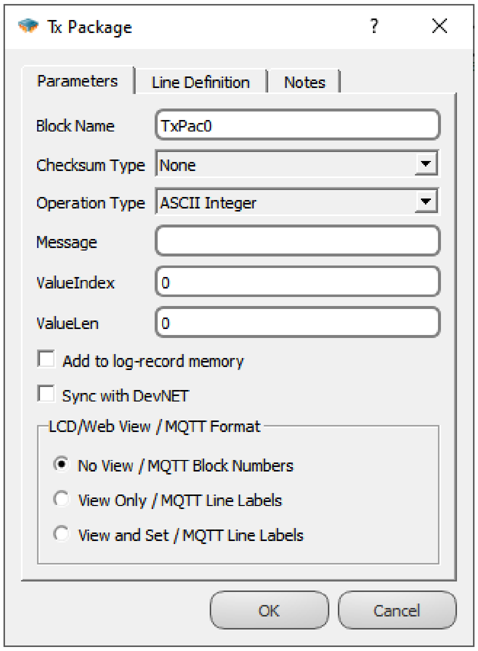
<u>Tx: Trigger input</u>

Each time a rising edge trigger comes to the “Tx” block input, it sends a data packet from the “#TxPac0” block output.

<u>#TxPAc0: Block output</u>

After the data is packaged, it is sent as packet data from the “#TxPac0” block output.

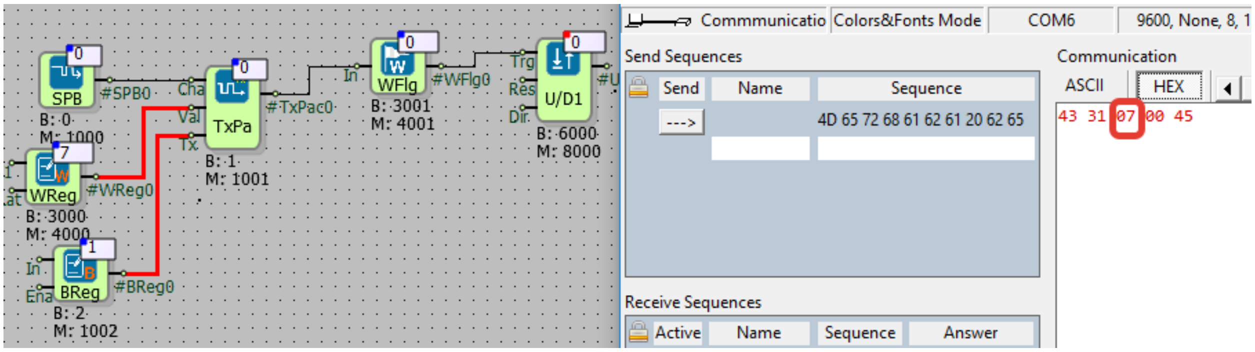
21.3.3 Block Settings

	Checksum Type:
	Operation Type: Specifies the type of data to be sent
	Message: The data package form to be sent is entered. Ex: " C1DDE" C: start E: end character
	ValueIndex: The data in the message to be sent starts from the 2nd Index. If this value is 0; Adds Data and Serial number to log record; If it is 1, it does not add.
	ValueLen: The size of the data to be sent in bytes is entered here.

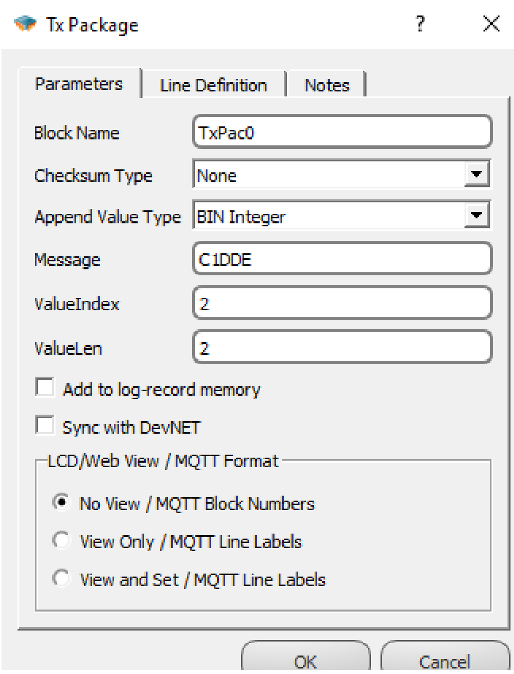
21.3.4 Block Explanation

In order for the data to be sent to be detected by other devices, it must be converted into a data packet form according to some rules and sent. Tx Packet block converts the data to be sent from the "Val" block input into a packet and sends a data packet from the "#TxPac0" block output.

21.3.5 Sample Application



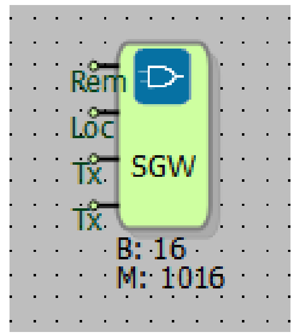
In the above example application, the value 7 is sent with the Tx Packet block and is read by the Docklight application, which is a serial port simulation.



In the picture on the side, the rules to package and send the data to be sent in the Tx Packet block are set.

21.4 Serial Gateway

21.4.1 Connections

Rem: Connection input	
Loc: Connection input	
Tx: Data size input	
Tx: Data TimeOut time input	

21.4.2 Connection Explanations

Rem: Connection input

The port input to which the remote device is connected is set.

Loc: Connection input

It is the serial port input to which the locally operating device is connected.

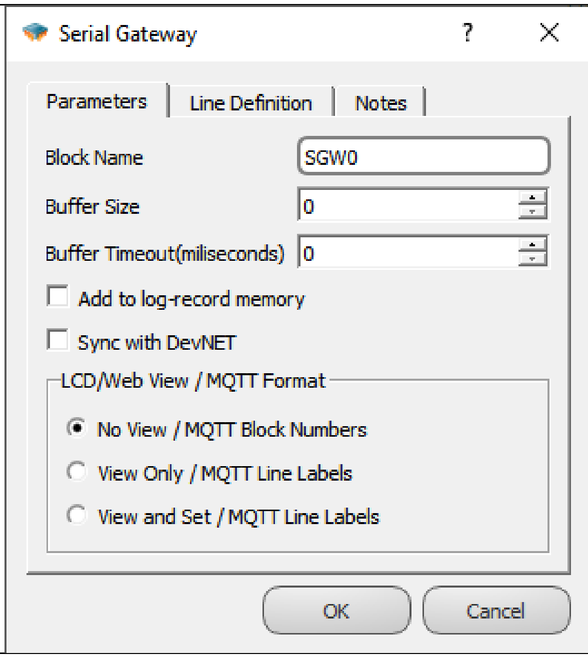
Tx: Data size input

It is the input where the size of the data to be sent is set.

#Tx: Data TimeOut time input

It is the input of the "timeout" duration of the data to be sent.

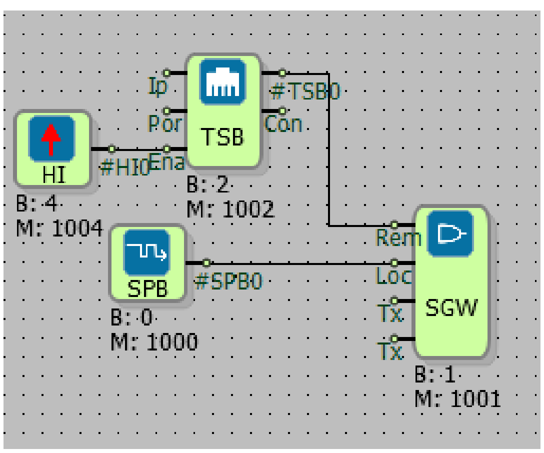
21.4.3 Block Settings

<table>
<tr><td rowspan="2">
</td><td>Buffer Size: It is the size of the value in bytes of the data to be sent in a single packet. This value can also be determined from the “Tx” block input.</td></tr>
<tr><td>Buffer Timeout (milliseconds): This is the place where the “timeout” period of the data to be sent is entered. This value can also be determined from the “Tx” block input.:</td></tr>
</table>

21.4.4 Block Explanation

It is the block used to provide transparent data transmission. It provides data transmission between the Serial Port block and the remotely connected device independent of any protocol.

21.4.5 Sample Application



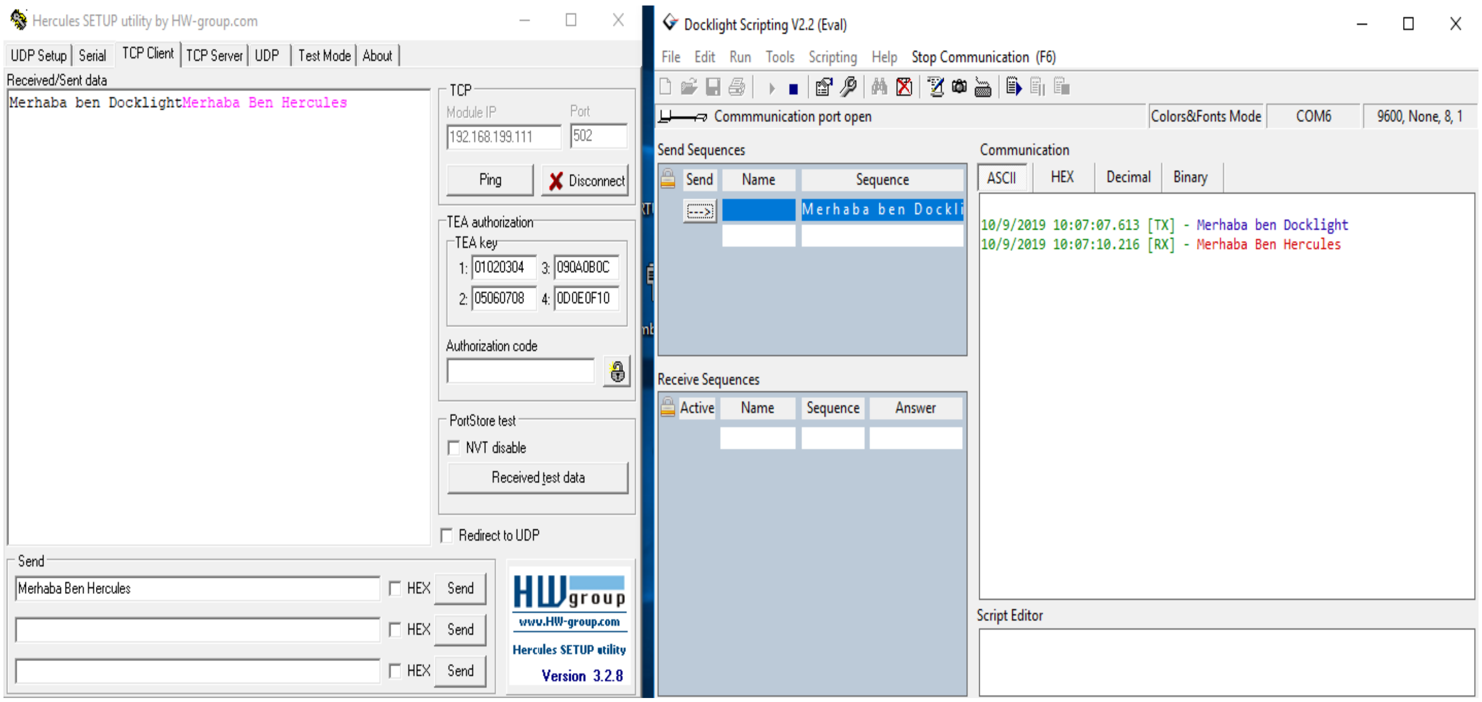
In the next picture:

Connection settings of the device connected remotely with TCP Socket block are entered.

The connection settings of the device connected to the PLC working locally with the Serial Port Block are entered via the Serial port connection. In this way, communication is provided between the device that communicates through the serial port working locally and the device that is connected remotely, using the Serial Gateway block.

Hercules app for remote device simulation,

Example using Docklight for serial connection simulation,

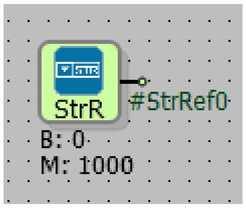


In the simulation application above, the "Hello, I'm Docklight" message sent as ASCII over the Serial Port was read by Hercules and the "Hello, I'm Hercules" message sent by Hercules was read by Docklight.

22 STRING BLOCKS

22.1 STRING REFERANCE

22.1.1 Connections

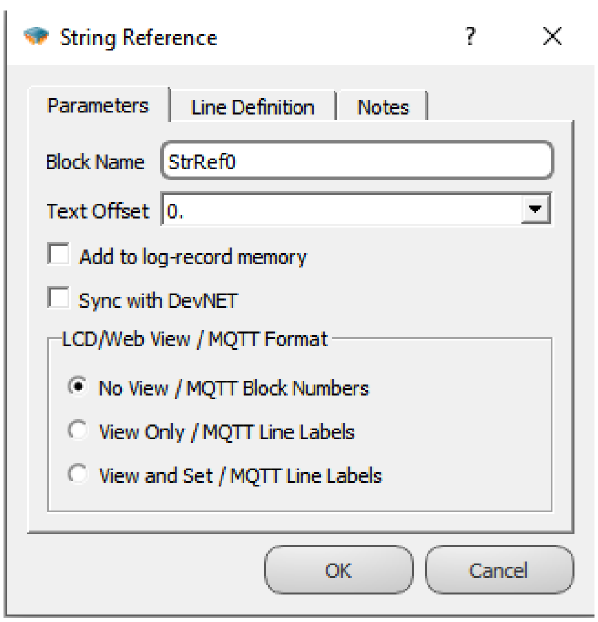
	#StrRef0: String data output

22.1.2 Connection Explanations

<u>#StrRef0: String data output</u>

String data output is a reference connection.

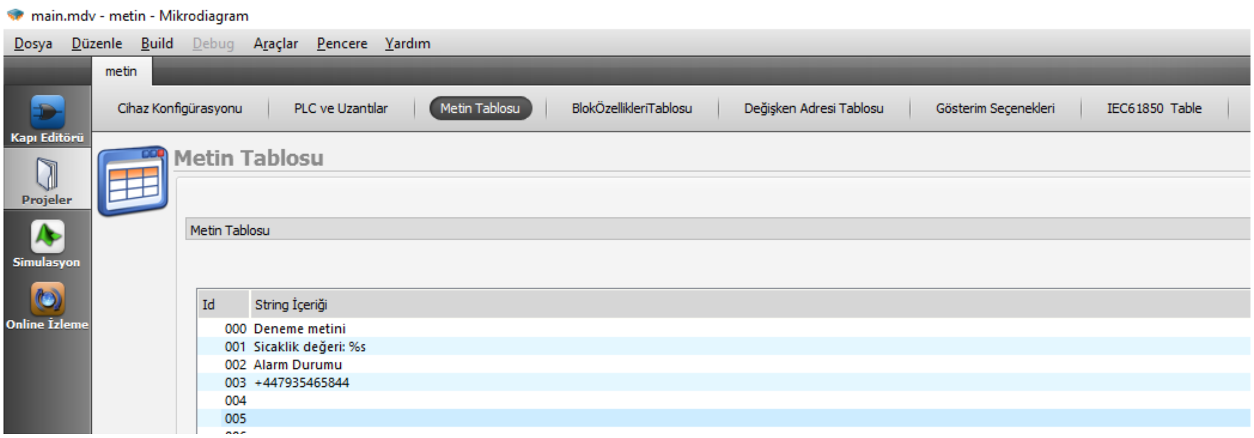
22.1.3 Block Settings

	String Offset: It is the part of the string table where the data to be used is selected.

22.1.4 Block Explanation

It is used to select and use the desired index in the String Table for blocks that process or input texts (String - Text).

”String table” part is pushed from project tab in order to reach String table on the Mikrodiagram or Telediagram

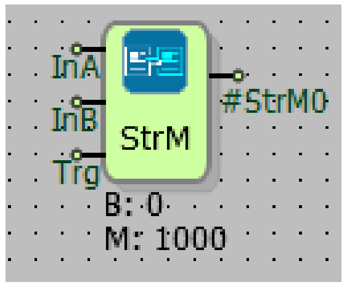


From on the String table will be used string contains such as (number, message containing etc.) can be entered in the String table. Each line can have max 63 characters on the String table.

When the do program can be used string reference in order to use values which have been recorded on the String table. Send and receiving SMS, acception calling and doing research on blocks and entered numbers and message content are identified from the String Reference block . SMS content where is on string table and choosing telephone number is done from the string reference block’s “string offset”.

22.2 STRING MANIPULATION

22.2.1 Connections

<table>
<tr><td>InA: First string value input</td><td rowspan="3">
</td><td rowspan="3">#StrM0: String change output</td></tr>
<tr><td>InB: Second string value input</td></tr>
<tr><td>Trg: Trigger input</td></tr>
</table>

22.2.2 Connection Explanations

<u>InA: First string value input</u>

<u>It is first string’s input.</u>

<u>InB: Second string value input</u>

It is second string’s input.

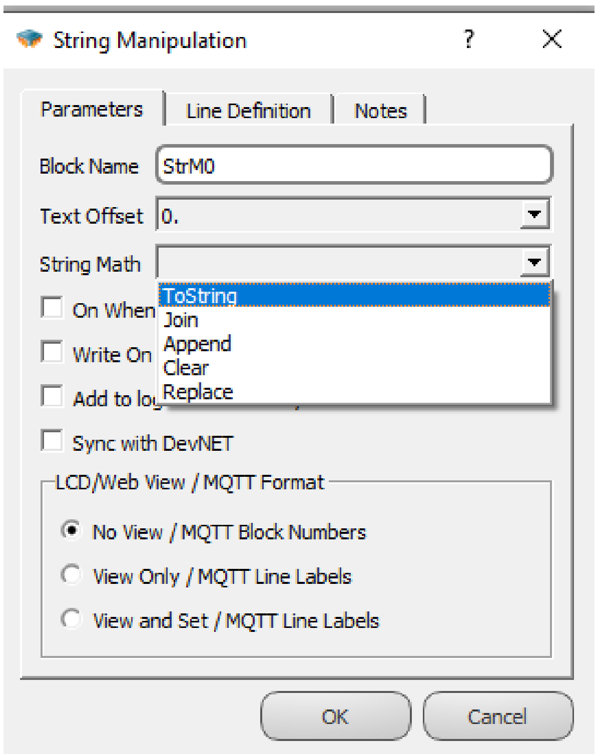
<u>Trg: Trigger input</u>

It is trigger input from block.

<u>#StrM0: String change output</u>

<u>It is string changing block connection.</u>

22.2.3 Block Setting

	Text Offset: Result of operation which is written on String table is determined the index.
	String Math: The part of the process to be done is selected.
	On When Trig is Active: If selected; When the rising edge (logic (1)) trigger comes to the block “Ttk” block input, the action is taken.
	Write On Input: If selected; The value at the “InA” block input and the value at the “InB” block input are processed, and the result is written to the “InA” block input.

22.2.4 Block Explantion

As do operation on the string reference result of operation new string is written to string offset. It which is operate type is as operate on the strings produce the string againly.

String format data to transformer, ToString, Join, Append (add to end), Clear, Replace is used for doing operation.

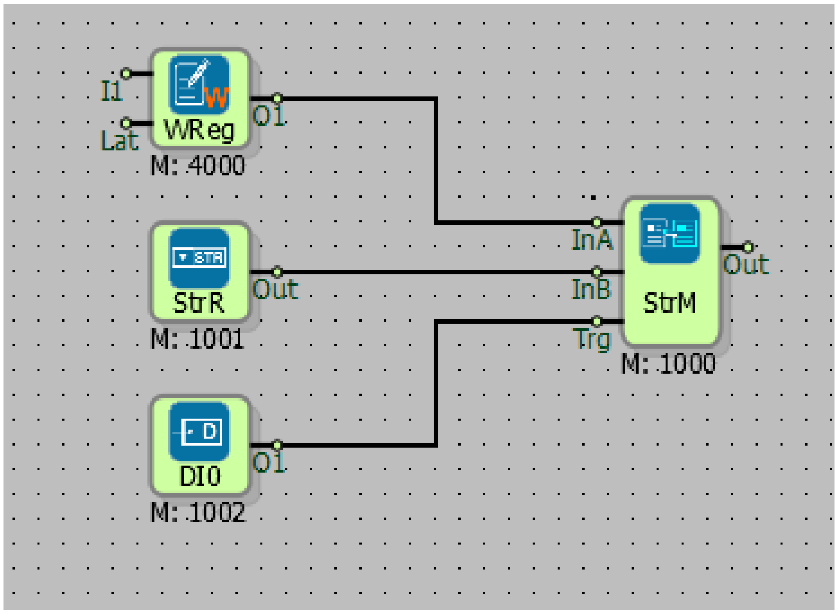
When convert to string function is selected; A word or long register is connected to the “InA” input. In the “InB” input, the text to which the value entered in the “InA” input will be written is selected with a string reference. The number of digits to be read to the part where the value entered from the “InA” input will be written should be specified with the expression “%s”. Word, Long and Analog values are converted to text with this operation. The result is written to the string table index selected from the String Offset in the block options.

When the Combine function is selected; The string reference connected to the “InA” input is combined with the string reference connected to the “InB” input. The result is written to the string table index selected from the String Offset in the block options.

When the append function is selected; The string reference linked to the “InB” input is appended to the end of the string reference linked to the “InA” input. The result is written to the string table index selected from the String Offset in the block options.

Process	**Entries Used**	**Explanation**
ToString	InA, InB	The value to be converted into text is entered from the long or word register value connected to the “InA” input. If the text to be connected to the “InB” input is from the reference block, the number of digits to be read from the “InA” input value should be specified with the expression “%s”. The value to be converted to text is saved in the string table index selected from the text offset part of the block options. (For example: If InA= 539 , InB= %03s, the number 539 is saved in the table index, the text selected from the text offset part of the text change block options.)
Join	InA, InB	When the Join function is selected; The string reference connected to the “InA” input is combined with the string reference connected to the “InB” input. The result is written to the string table index selected from the String Offset in the block options. (Ex: InA=micro, InB=dev Result=mikrodev)
Append	InA, InB	When the append function is selected; The string reference linked to the “InB” input is appended to the end of the string reference linked to the “InA” input. The result is written to the string table index selected from the String Offset in the block options. (Ex: InA=micro, InB=dev Result=mikrodev)

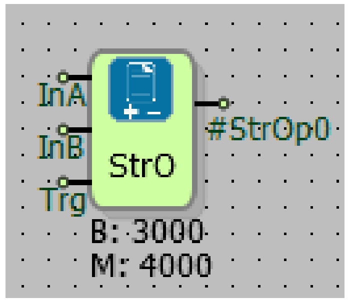
22.2.5 Sample Application



The data from the word register connected to the inA input will be converted to string value with the expression "% s" at the inB input. When the trigger is active is signed when only DI0 logic high(1) signal comes will operate.

22.3 STRING OPERATION

22.3.1 Connections

InA: First string value input		#StrOp0: String processing output
InB: Second string value input		
Trg: Trigger input		

22.3.2 Connection Explanations

InA: First string value input

It is first string value input

InB: Second string value input

It is second text value input.

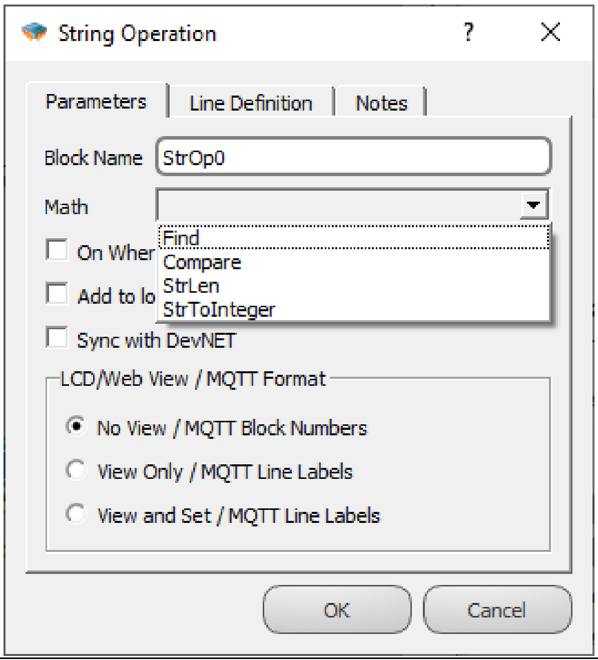
Trg: Trigger input

It is block trigger input.

#StrOp0: String operation output

Text processing is output connection

22.3.3 Block Settings

<table>
<tr>
<td rowspan="2">
</td>
<td>Math: Strings process steps that are part of the selection.</td>
</tr>
<tr>
<td>On When Trig is Active: If selected; When the rising edge (logic (1)) trigger comes to the block “Ttk” block input, the action is taken.</td>
</tr>
</table>

22.3.4 Block Explanation

As doing operation on the string reference ,result of operation composed the integer value is written blocks output.

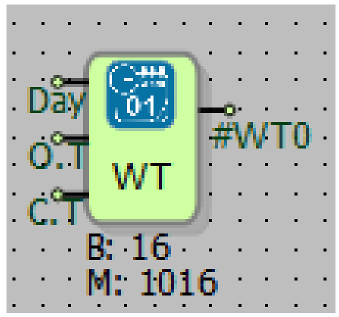
Performed operations and expectation are given below:

Find	If the InA string reference includes the InB string reference, it outputs which index of the InB string reference the expression in InB starts from.
Compare	Compares the indices of InA and InB string references and outputs the ASCII equivalent of the different value. **Note:** The InA string reference must contain the InB text reference. **Note:** If the InB string reference includes the InA text reference, it subtracts the ASCII equivalent of the different value from 65.356 and writes it to the output.
StrLen	Writes the number of characters of the text in the InA reference to the output.
StrToInteger	The text in the InA reference converts the content to integers and writes it to the output

23 CALENDER BLOCKS

23.1 WEEKLY TIMER

23.1.1 Connections

Day: Day selection input		#WT0: Block Output
O.T: Opening time input		
C.T: Closing time input		

23.1.2 Connection Explanations

Day: Day selection input

It is day selection input.

O.T: Opening time input

It is the input which determine opening time.

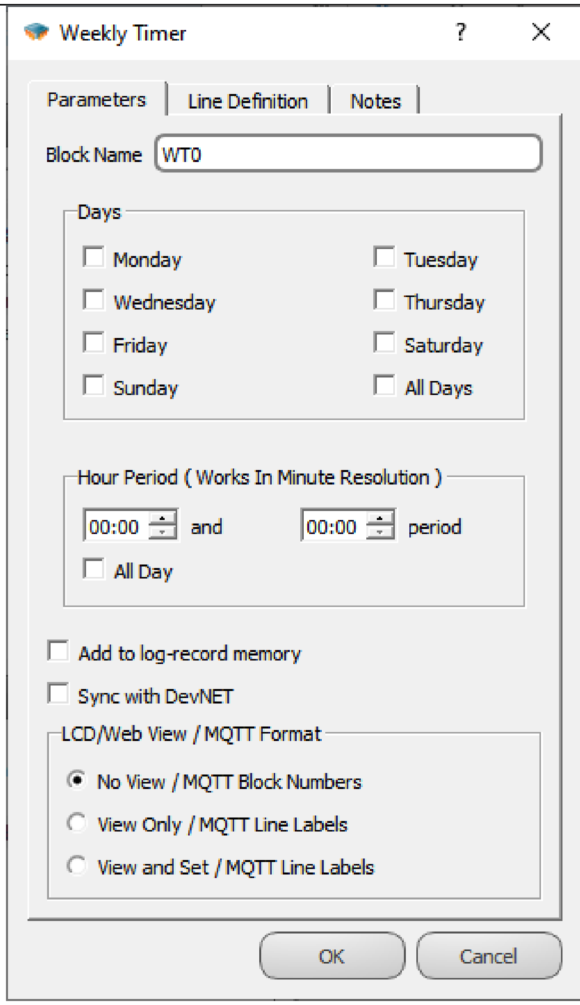
C.T: Closing time input

It is the input which determine closing time.

#WT0: Block output

It is block output which is producing logic(0) and logic(1) signal.

23.1.3 Block Settings

	Days: Determines the operating days for weekly timer.
	Hour period: It determines weekly timer's operating interval.
	All Day: If it is chosen the hour period becomes passive; block output is activated during 24 hours for the selected days

23.1.4 Block Explanation

O1 output becomes logic(1) for the selected day and time intervals.

Provides simple and excellent programming ease in the control of the systems which are to be operated at the determined days and time intervals of the week.

When week's day is desired to choose from out of block, every day is represented by one bit.

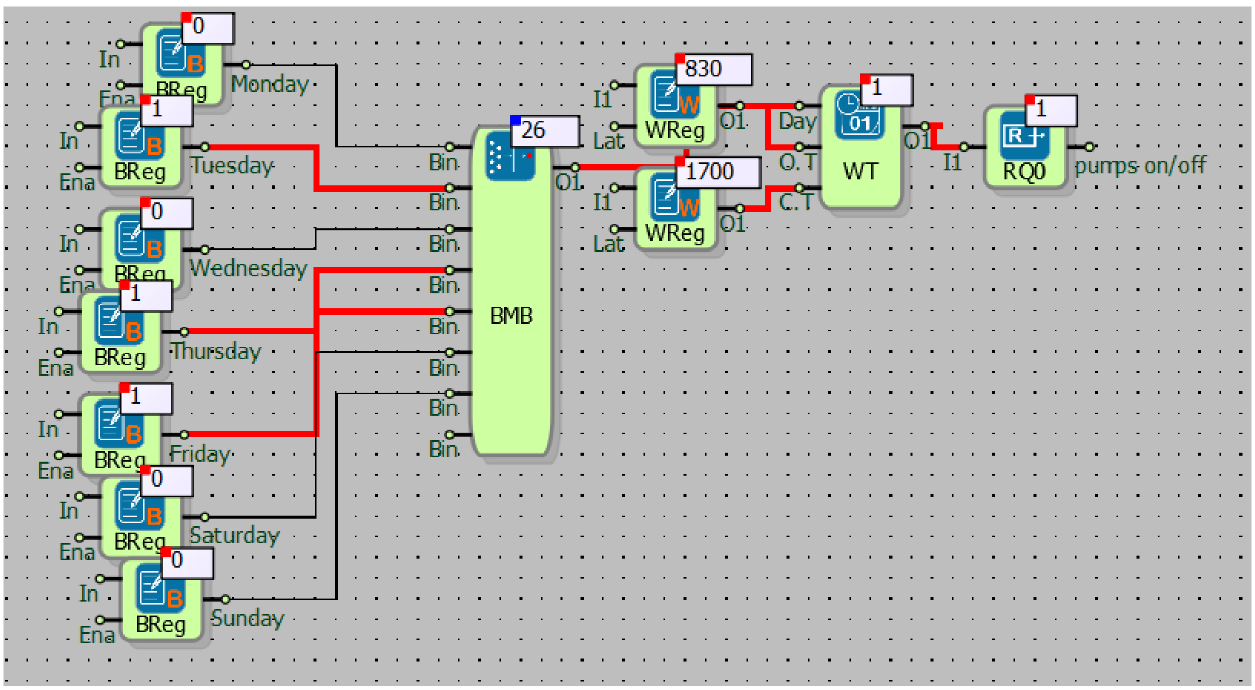
The least significant bit(LSB) represents Monday, the most significant bit (MSB) represents Sunday. Thus 1 for Monday, 2 for Tuesday, 4 for Wednesday, 8 for Thursday, 16 for Friday, 32 for Saturday, 64 for Sunday values must be entered. When more than one day is wanted to be chosen, corresponding values is written as a sum.

For example, when it is wanted to choose Monday, Wednesday and Friday, 1+4+16=21 value must be entered.

To insert O.T. and C.T. values from out of the block, the value is entered with no punctuations in between. For example, 16:30 should be written as 1630. For 01:17, 117 should be entered.

Since weekly timer works in minute resolution, the outputs are updated in a period of +30 seconds.

23.1.5 Sample Application



In the example, Bit Merge Block is connected into weekly timer inputs. For Bit Merge Block

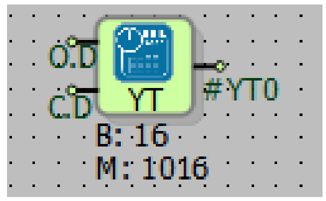
, every input is represented by one day. Binary registers are connected to Bit

Merge block's inputs. One word register is connected to the O.T. and C.T. inputs, and it is aimed to control the on/off state of the pump by using the relay output (RQ0) connected to the output of Weekly Timer.

Pump will operate in the selected days of the week such as Tuesday, Thursday, Friday between 08.30 and 17.00 hours and it will not operate in other days and times.

23.2 YEARLY TIMER

23.2.1 Connections

O.D: Date of opening input C.D: Date of closing input		#YT0: Block output

23.2.2 Connection Explanations

O.D: Date of opening input

It is the 32 bits long opening date input value. This value is of Unix Epoch seconds. The seconds value since 00:00 1/1/1970 is inserted as the opening time.

It is the input for date of opening.

C.D: Date of closing input

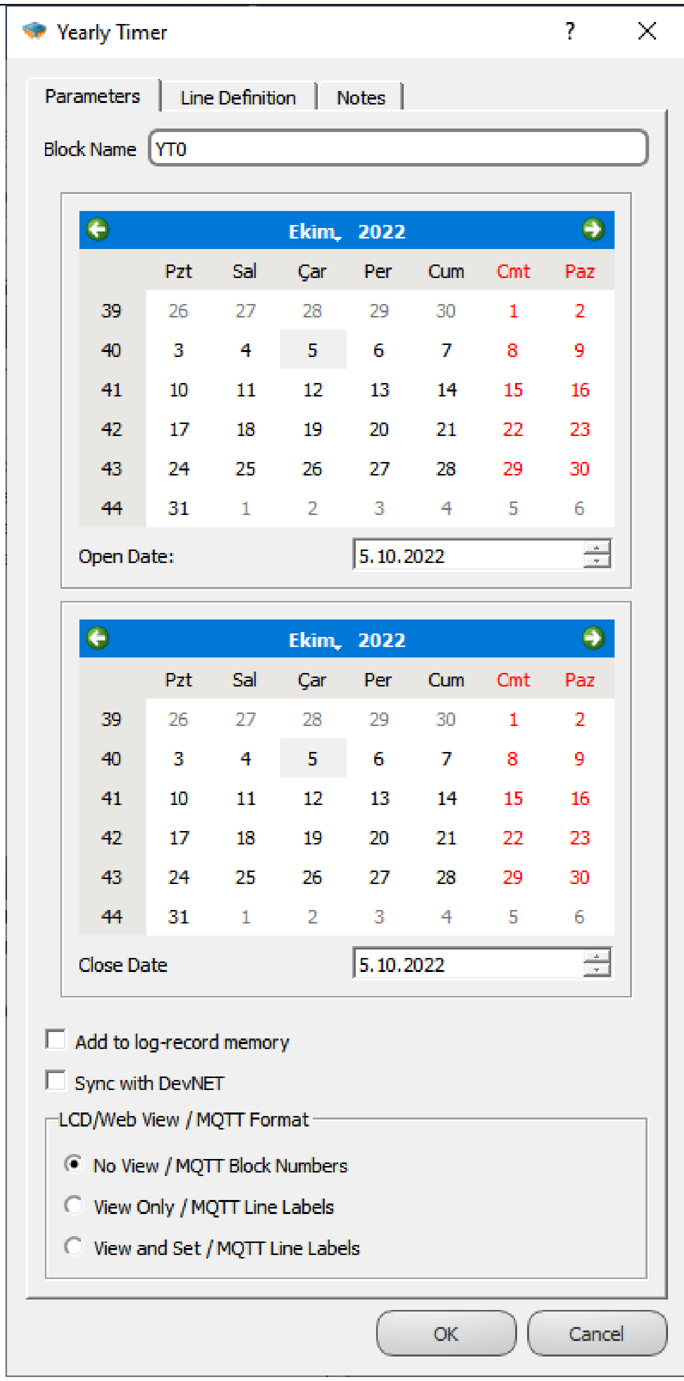
It is the 32 bits long closing date input value. The seconds value since 00:00 1/1/1970 is inserted as the closing time.

It is the input for date of closing.

#YT0: Block Output

The yearly timer block’s output which is logic(0) or logic(1)

23.2.3 Block Settings

	Open Date: The date value which Yearly timer's output will be logic (1) can be determined in the block.
	Close Date: The date value which Yearly timer's output will be logic (0) can be determined in the block.

23.2.4 Block Explanation

It is used to generate a logical (1) output between two selected time intervals of the year. It

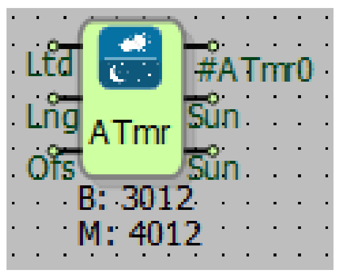
outputs the logic(1) signal in between opening and closing dates which are entered and if else logic(0).

If opening date(O.T) and closing date (C.T) is entered from out of block, Unix Epoch Time type is entered as opening and closing time. In order to calculate Unix Epoch Time from date value, the below link could used.

https://www.epochconverter.com/

23.3 ASTRONOMICAL TIMER

23.3.1 Connections

Ltd: Input for latitude value		#ATmr0: Block Output
Lng: Input for longitude value		SunRise: Sunrise time
Offs: Input for offset		SunSet: Sunset time

23.3.2 Connection Explanations

Ltd: Input for latitude value

It is the latitude coordinate information of the geographic location which is used to calculate the sunrise and sunset time. For example, only 51 must be entered for 51°30'

Lng: Input for longitude value

It is the longitude coordinate information of the geographic location which is used to calculate the sunrise and sunset time. For example, only 39 must be entered for 39°20'

Offs: Input for offset

It is used to select the time zone for summer/winter time. Time period offset is entered such as -10, -9, … +1, +2, .. +9

#ATmr0: Block output

For the location in the entered coordinates, Block output is logic(1) for day time and logic(0) for night time.

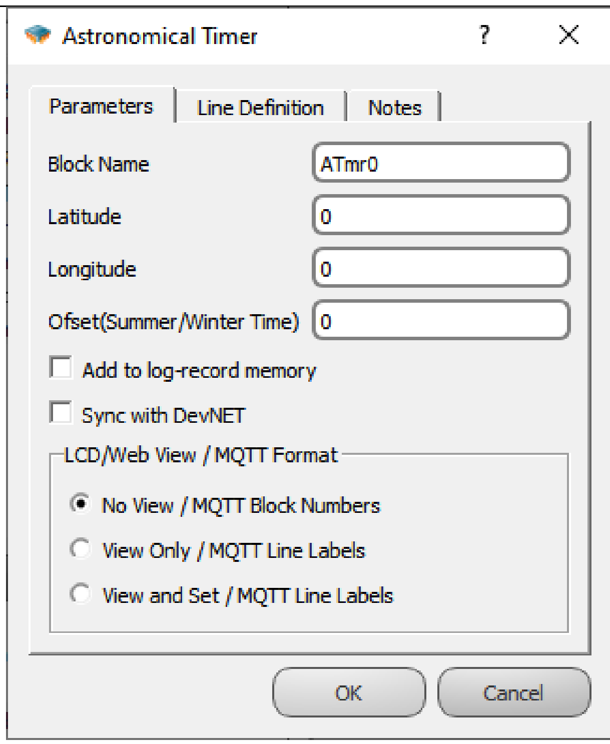
SunRise: Sunrise time

Sunrise time for location in the entered coordinates. For example if sunrise is 05:43, 543 value is read in this block output.

SunSet: Sunset time

Sunset time for location in the entered coordinates. For example if sunset is 18:25, 1825 value is read in this block output.

23.3.3 Block Settings

	Latitude: The value of latitude is entered within the block.
	Longitude: The value of longitude is entered within the block.
	Offset: The time period, can be selected within the block

23.3.4 Block İnformation

Sunset and sunrise time is calculated by using the latitude and longitude values. This time calculation is run once in everyday at midnight. According to sunrise/sunset time, the block output is set. Output of block is updated once every minute.

In the Day output of block, during the daytime logic(1) signal output is generated, after sunset the logic(0) output I generated during the night time for the entered coordinates.

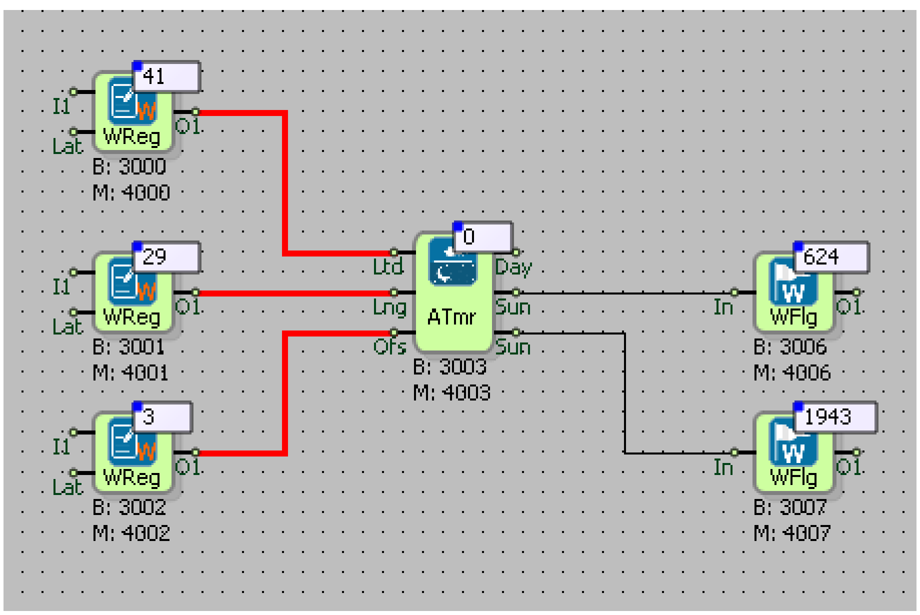
Sunrise output is the sunrise time for specified coordinates. For example, if sunrise time is 05:43, 543 value is read in this block output.

Sunset output is the sunset time for specified coordinates.. For example, if the time of sunset is 18:25, 1825 value is read in this block output.

The Offset Value is the time period for GMT. The information of time period can be entered as a + or – value.

Ltd, Lng and Ofs inputs can be entered within the block.

23.3.5 Sample Application



Astronomical timer's latitude, longitude and offset information is determined with registers. These values may also be determined within the block. The digital output or relay output can be connected to the "Day" output.

For example; latitude 41, longitude 29 and offset 2 values must be entered for İstanbul. When clocks go forward for summer time the offset should be set to 3.

Sunset and sunrise times can be viewed from output of “Sunrise” and “Sunset” outputs.

23.4 SYSTEM SECONDS

23.4.1 Connections

	#SSB0: Block output

23.4.2 Connection Explanations

#SSB0: Block output

Unix Epoch Time seconds value is written to this output

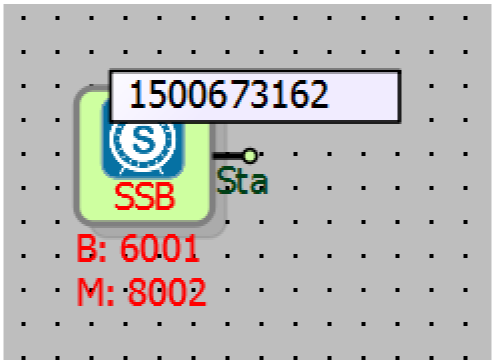
23.4.3 Block Settings

There is no block settings.

23.4.4 Block Explanation

The system second block shows the PLC’s real time clock’s second value. The information which is from PLC real time is calculated as seconds since Linux Epoch (00:00 1/1/1970) and is written to block output.

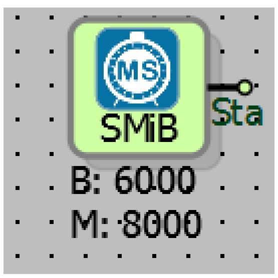
23.4.5 Sample Application



In the example, real second value of the PLC is read.

23.5 SYSTEM MILISECONDS

23.5.1 Connections

	Sta: Block output

23.5.2 Connection Explanations

Sta: Block output

It is block output which shows the system's milliseconds as a 32 bit value.

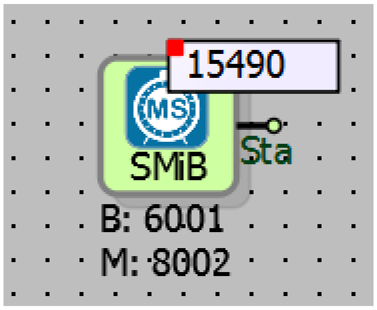
23.5.3 Block Settings

There is no block settings

23.5.4 Block Explanation

"System Milliseconds" block reads the millisecond value from the moment the PLC starts to operate. When the device is rebooted, this counter resets to zero and it starts to counter from 0.

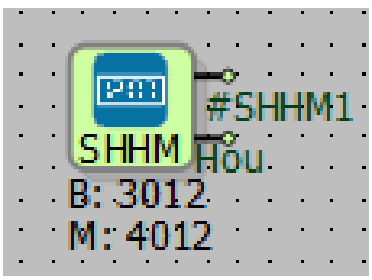
23.5.5 Sample Application



In the example, time since the PLC reset is seen as milliseconds value.(The system is reset nearly before 15 seconds.)

23.6 SYSTEM HHMM (HOUR-MINUTE)

23.6.1 Connections

<table>
<tr><td rowspan="2">
</td><td>#SHHM1: Block minutes output</td></tr>
<tr><td>Hou: Block hour output</td></tr>
</table>

23.6.2 Connection Explanations

<u>#SHHM1: Block minutes output</u>

It is the minute value, 16 bits long Word

<u>Hou: Block hour output</u>

It is the hour value, 16 bits long Word

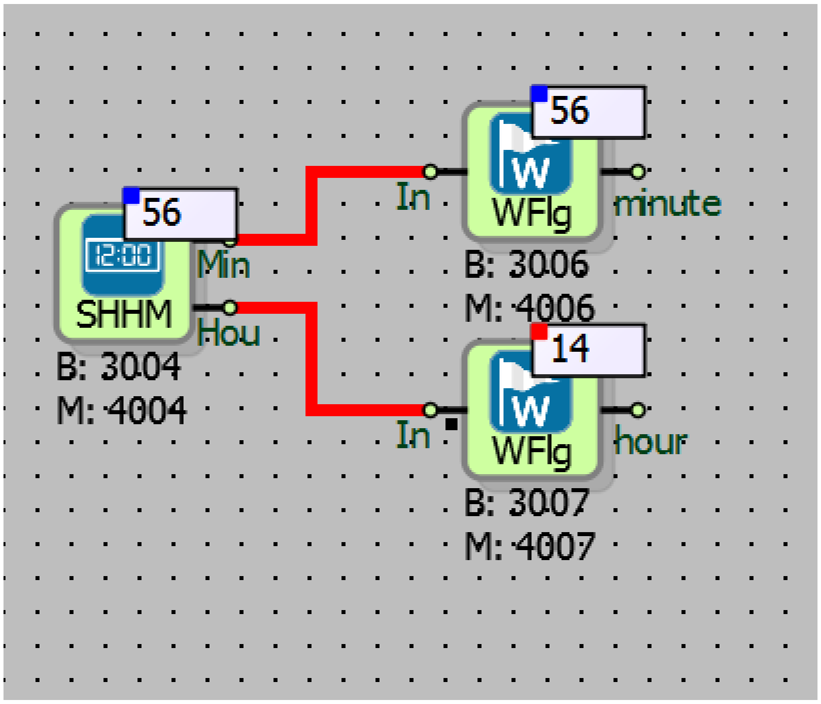
23.6.3 Block Settings

There is no block settings.

23.6.4 Block Explanation

System HHMM block shows the PLC’s real time clock’s hours and minutes value. The minutes is displayed between the 0-59 and hours is displayed between 0-23

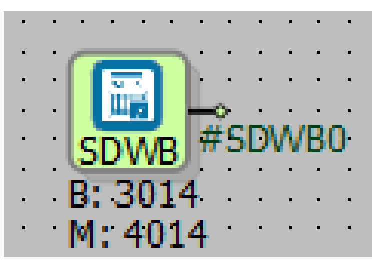
23.6.5 Sample Application



The PLC's hours and minutes information is read and that the current time is seen as 14:56.

23.7 SYSTEM DAY OF WEEK

23.7.1 Connections

	#SDWB0: Block output

23.7.2 Connection Explanations

#SDWB0: Block output

It is 16 bits long word output that read the day of weeks value

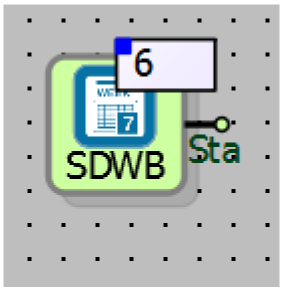
23.7.3 Block Settings

There is no block settings.

23.7.4 Block Explanation

System day of week shows PLC's real time clock's week of day. It is read such as Sunday 0, Monday 1, Tuesday 2,Wednesday 3,Thursday 4, Friday 5,Saturday 6.

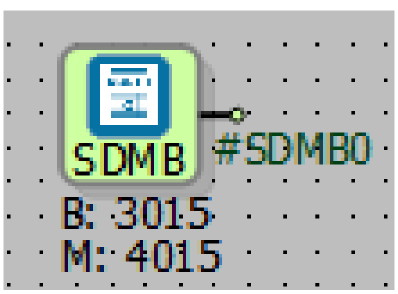
23.7.5 Sample Application



Shows the day of week. If read value is six, then the day is Saturday.

23.8 SYSTEM DAY OF MONTH

23.8.1 Connections

	#SDMB0: Block output

23.8.2 Block Explanation

#SDMB0: Block output

It is 16 bits long word output that read the day of month value.

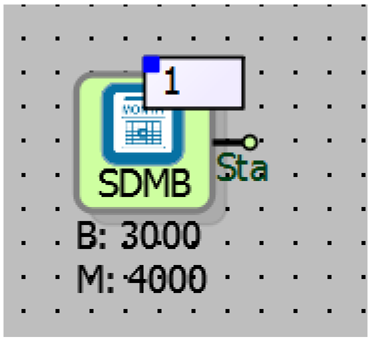
23.8.3 Block Settings

There is no block settings.

23.8.4 Block Explanations

The system day of month block shows PLC's real time clock's day of month. It displays values between 1-31.

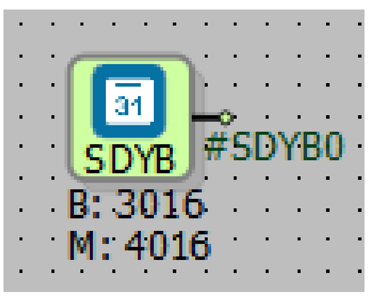
23.8.5 Sample Application



Day of month value is showed on the block.

23.9 SYSTEM DAY OF YEAR

23.9.1 Connections

	#SDYB0: Block output

23.9.2 Connection Explanations

#SDYB0: Block output

It is 16 bits word output that read the day of year value.

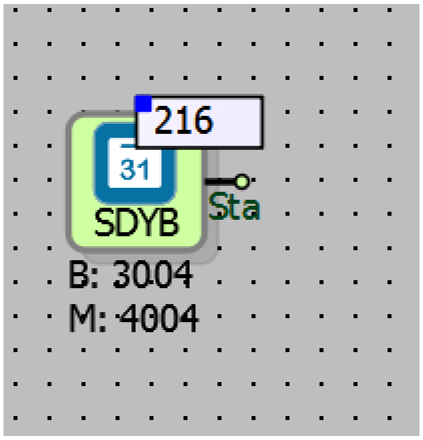
23.9.3 Block Settings

There is no block settings.

23.9.4 Block Explanation

The system day of year block shows PLC's real time clock's day of year value. It can take value s between the 1-365.

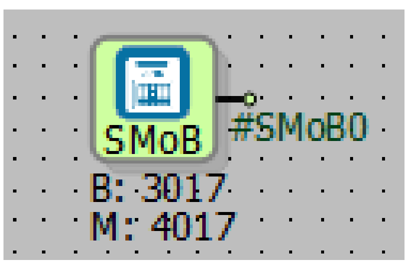
23.9.5 Sample Application



The value of the day of the year is read and it is 216 days since the beginning of the year.

23.10 SYSTEM MONTH

23.10.1 Connections

	#SMoB0: Block output

23.10.2 Block Explanation

<u>#SMoB0: Block output</u>

It is the block output

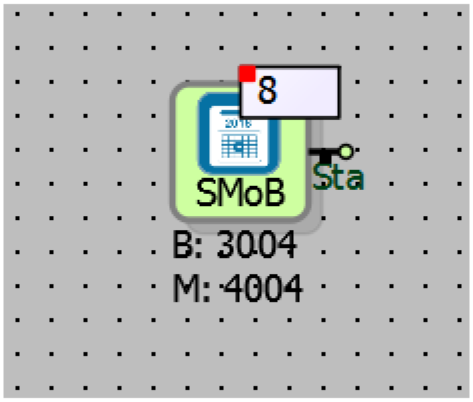
23.10.3 Block Settings

There is no block settings

23.10.4 Block Explanation

System Month block shows PLC's real time clock's month of year value.

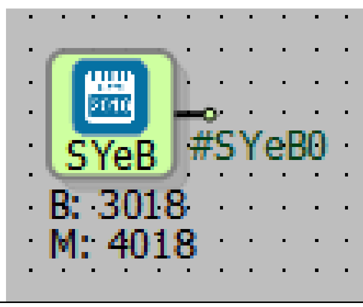
23.10.5 Sample Application



It is seen that it is the eighth month of the year (August).

23.11SYSTEM YEAR

23.11.1 Connections

	#SYeB0: Block output

23.11.2 Connection Explanations

#SYeB0: Block output

It is the connection of block output

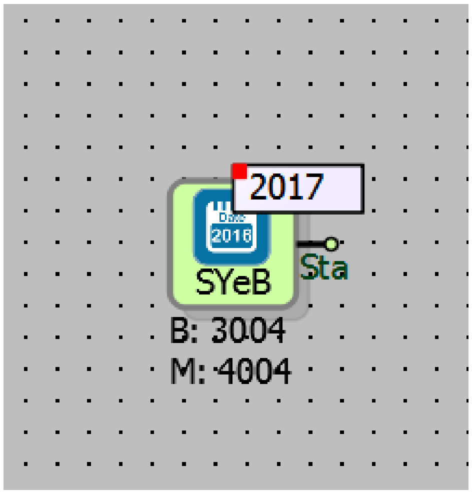
23.11.3 Block Settings

There is no block settings

23.11.4 Block Explanation

System Year Block shows PLC's real time clock's year value.

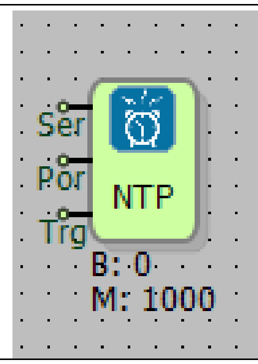
23.11.5 Sample Application



It is read the year value of the system.

23.12 NTP SYNCRONISE BLOCK

23.12.1 Connection

Ser: NTP Server Input	
Por: NTP Server Port Input	
Trg: Trig Input	

23.12.2 Connection Explanations

<u>Ser: NTP Server Input</u>

NTP Server IP can be defined from this entry in the block.

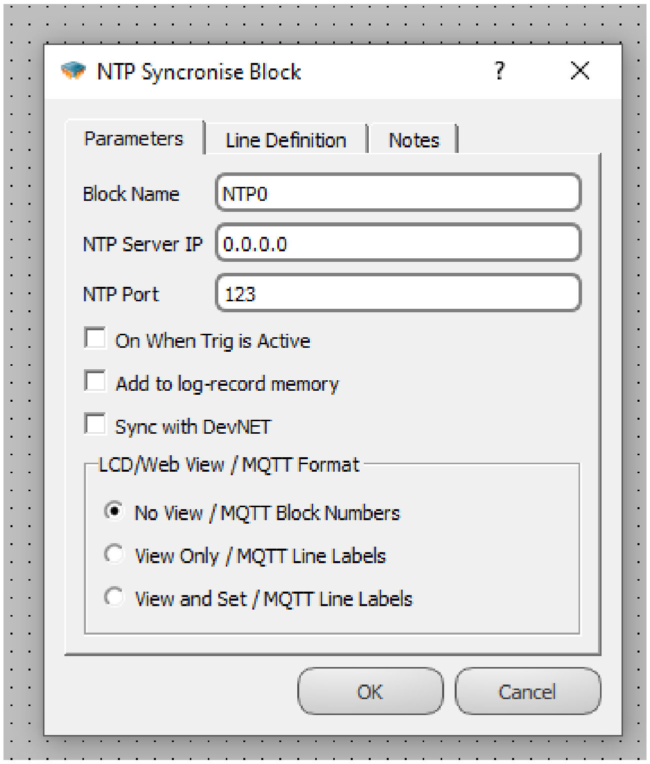
<u>Por: NTP Server Port Input</u>

NTP Server Port number can be defined from this entry in the block

<u>Trg: Trig Input</u>

It is the trigger input for synchronization. It works as a rising edge.

23.12.3 Block Settings

	NTP Server IP: NTP Server IP number can be entered from the block entry or can be set from the block special settings..
	NTP Server Port: NTP Server Port number can be entered from the block entry or can be set from the block special settings.

Note: In order for the trigger to work, the "On When Trig is Active” option must be selected from the block block settings.

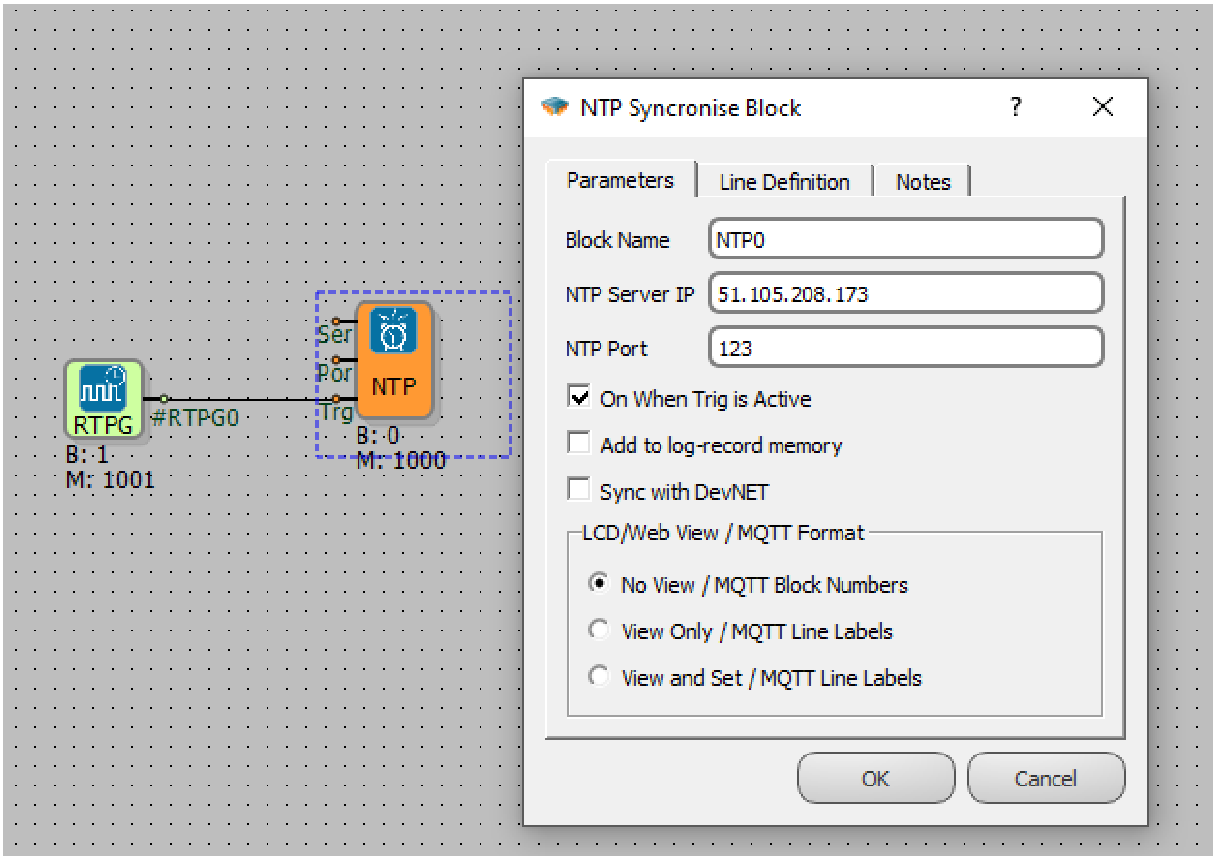
23.12.4 Block Explanation

Since the NTP Synchronization Block is active with the high edge signal coming to the Trg-Trigger pin, Real Time Pulse Generator, Symmetrical Pulse Generator or Binary Register block can be connected to the block trigger input. Blocks connected to the trigger input are used to set the match frequency with the NTP server.

For NTP server settings, NTP server IP is entered in the NTP Server IP section of the function block. In the NTP port part, the server port is entered. On When Trig is Active option, on the other hand, enables the block to run as a result of the trigger.

If desired, NTP Server IP and NTP Port information can also be defined by connecting to the Ser and Por pins of the NTP Syncronise Blocks.

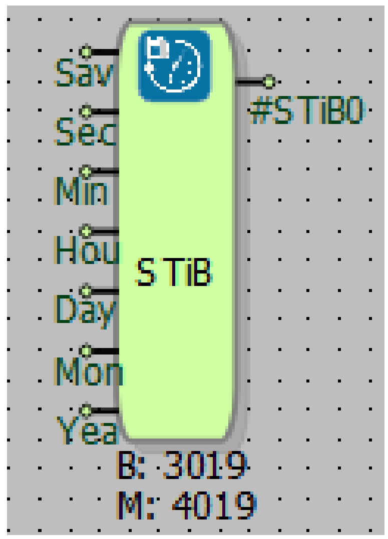
23.12.5 Sample Application



The timing frequency of the real-time pulse generator is 1 per second. The RTPG block sends a trigger once per second to the NTP synchronization block, performing a time synchronization with the NTP server once per second

23.13 SAVE TIME

23.13.1 Connections

Sav: Saving trigger input		#STiB0: Block output
Sec: Second input		
Min: Minute input		
Hou: Hour input		
Day: Day input		
Mon: Month input		
Yea: Year input		

23.13.2 Connection Explanations

<u>Sav: Saving trigger input</u>

It is the input to be triggered in rising edge for saving time.

<u>Sec: Second input</u>

It is the seconds input of Save Time.

<u>Min: Minute input</u>

It is the minutes input of Save Time.

<u>Hou: Hour input</u>

It is the hour input of Save Time.

<u>Day: Day input</u>

It is the day input of Save Time.

<u>Month: Month input</u>

It is the month input of Save Time.

Year: Year input

It is the year input of Save Time.

#STiB0: Block output

It is the block output connection.

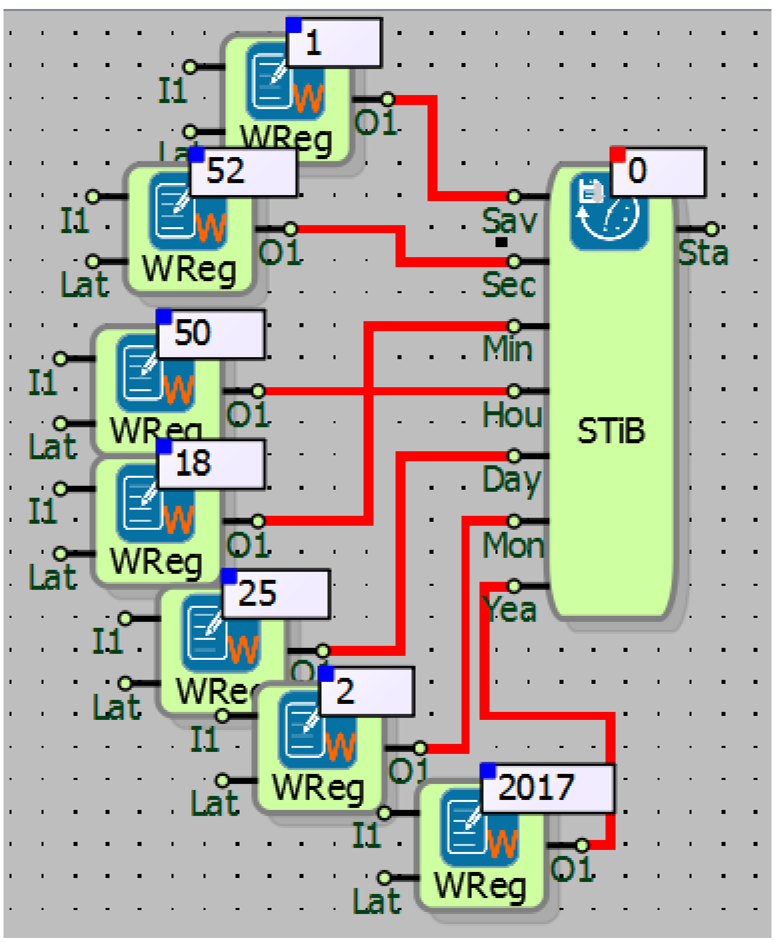
23.13.3 Block Settings

There is no block settings.

23.13.4 Block Explanation

It is used to set the PLC's time and date within the logic project. It saves the values written into the block inputs to the real time clock of the PLC at the rising edge instance of the "Save Input".

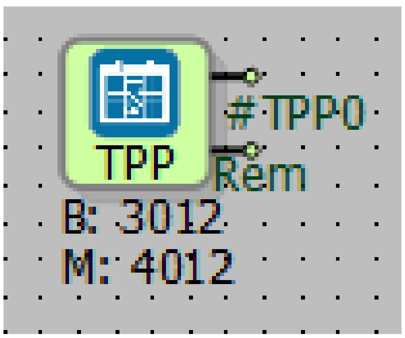
23.13.5 Sample Application



In the example; the time and date values written to the inputs of the save time block are written to the real time clock at the rising edge trigger of the "Sav" input.

23.14 TIME PLAN PICKER

23.14.1 Connections

	#TPP0: Default output
	Rem: Remaining output

23.14.2 Connection Explanations

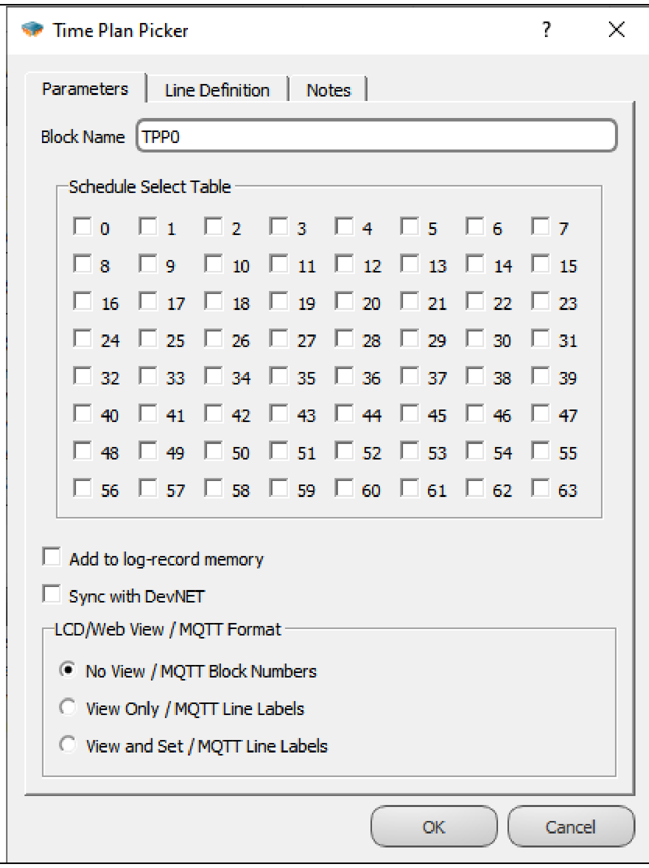
#TPP0: Default output

It is the default output.

Rem: Remaining output

It is the remaining output.

23.14.3 Block Settings



Schedule Select Table:
Indexes for tags are defined.

23.14.4 Blok Explanation

It can be used in conjunction with Mikrodev ViewPLUS SCADA. It CAN NOT BE USED alone.

If the index is selected in the Mikrodiagram "Time Plan Picker", the same index of the "Schedule Tag" must be selected in ViewPLUS SCADA. In order to make settings for "Time Plan Picker", at "View PLUS SCADA" ; "Scheduler" must be added to "Scada Editor" and "Schedule tag" must be selected.

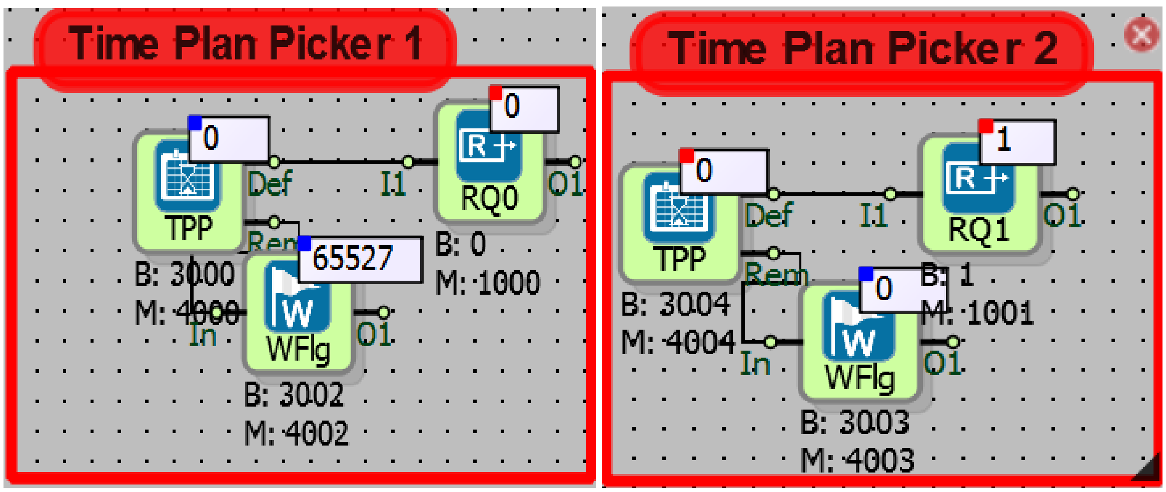
If more than one index is selected in the time plan picker and the tags defined in these indexes are added to ViewPLUS SCADA, output logic (1) occurs between the indexes if the index condition is met with OR operation.

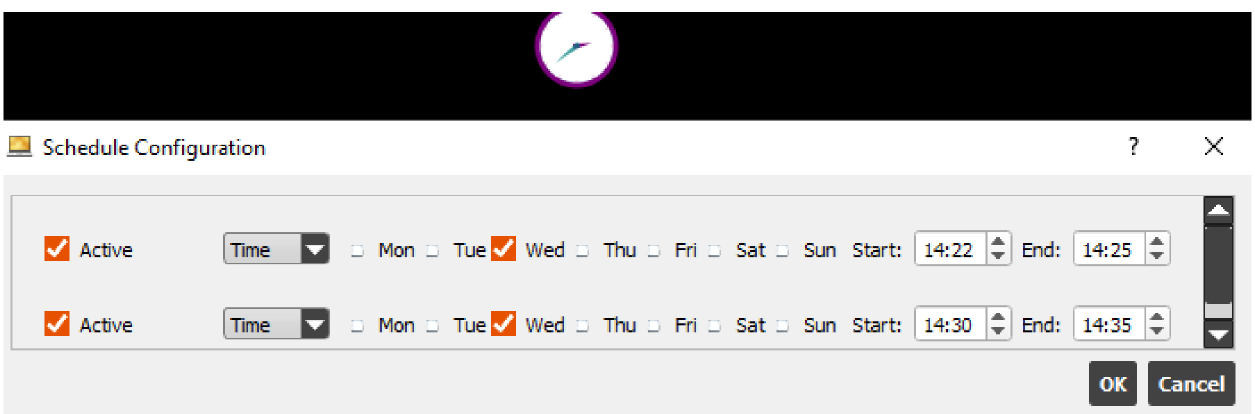
The "Def" output of the "Time Plan Picker" is in logic(1) if the day and time is in selection range of the PLC clock selected from "ViewPLUS SCADA", while in other cases the "Def" output is logic(0).

"Rem" output block is logic low(0), if it satisfies the time zone condition selected from the ViewPLUS SCADA; if it does not, it shows how long remained for the condition to be satisfied.

Note: A maximum of 63 different indices can be defined in the PLC, if an index is defined on more than one "Time Plan Picker", the block outputs give the same output.

23.14.5 Sample Application





In the examples; PLC program is in first picture and ViewPLUS SCADA interface is in the second picture.

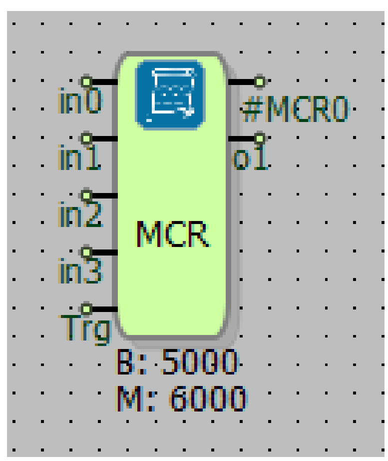
In the case of “time plan picker 1”, the output of "Def" output and relay (RQ0) is logic (1) because the PLC time is on one of the selected days and the time is between 14:22 and 14:25. The “Rem” output is logic low(0) because of the output “Def” is logic (1).

In the case of “time plan picker 2”, the PLC time date is on one of the selected days but since the time is not between 14:30 and 14:35, the "Def" output signals logic (0) and the relay (RQ0) is inactive. The "Rem" output shows how many minutes are left until 14:30. In this case, it can be estimated that PLC time is 14:30 since at the output of "Rem" is the value of 0.

24 MACRO BLOCKS

24.1 MACRO

24.1.1 Connection

in0: First data input		#MCR0: First data output
in1: Second data input		
in2: Third data input		o1: Second data output
in3: Fourth data input		
Trg: Trigger input		

24.1.2 Connection Explanations

<u>in0: First data input</u>

It is the first data input.

<u>in1: Second data input</u>

It is the second data input.

in2: Third data input

It is the third data input.

in3: Fourth data input

It is the fourth data input.

Trg: Trigger input

It is trigger connection input.

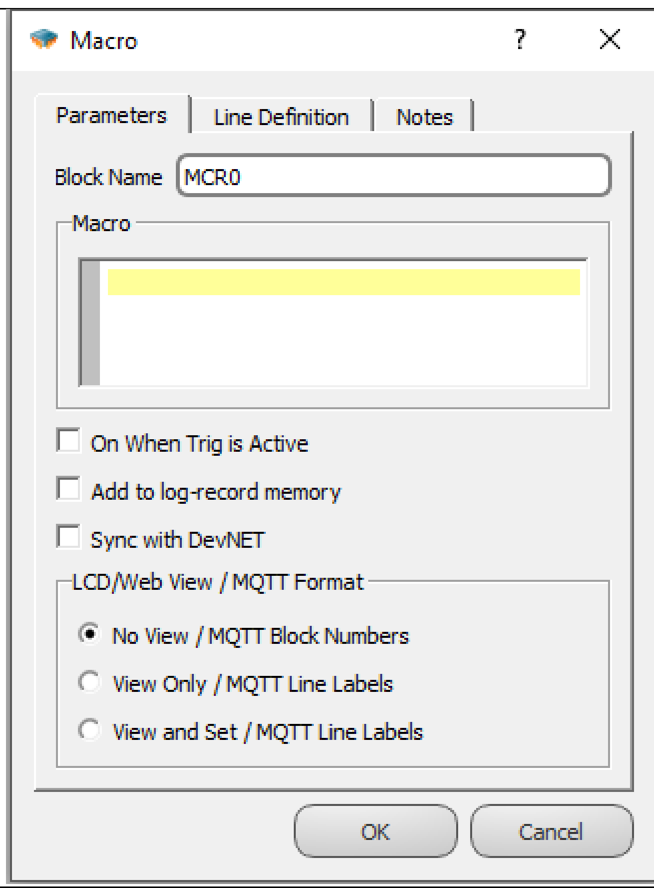
#MCR0: First data output

It is the first data output.

o1: Second data output

It is the second data output.

24.1.3 Block Settings

	Macro: It is the field where custom command definitions are made.

24.1.4 Block Explanations

Custom blocks can be designed by inserting special command definitions in the macro field of the block.

There are 50 analog variables you could use in the macro block. You can use variable definitions in the macro using the addresses "v0", "v1", "v2" ... and "v49". The variables are off floating point-analog type .

The addresses "i0", "i1", "i2" and "i3" can be used to read data from the inputs of the macro block.

The addresses "#MCR0" and "o1" can be used to transfer data to the outputs of the macro block.

If you want to read any block value in Mikrodiagram application within the macro, you can use it by specifying "$" expression and block number.

For example; It is enough to write "$1056" to address the value of block with “block number 1056” in the macro. Mikrodiagram macro addressing solution allows all blocks in the diagram area to be written and read.

"[" Character is created with the command line start. "]" Creates a command line break.

Basic command line usage is :

["addressing" = "addressing", "command", "addressing"].

The expression "[E]" specifies the macro end. Use of conditional expression (IF);

[IF, <State 0/1>, <jump line>]

For example; If the “State” value is “0”, it is passed to the next command line by the step count specified in the jump line. If the state value is 1, execution is continued from the next command which is just below the “IF” expression line. Positive values for the bottom rows and negative values for the top rows are used to jump between lines in the “IF” command line. “2” is written to go to lower two lines of the IF command and “-2” to go to the upper two lines of the IF command.

Example code;

[v1 = v0> $1504]

[IF, v1,2]

[v2 = $1504 + 0]

In the above example;

[v1 = v0> $1504]

> If the value of "v0" is greater than the value of block $1504, logic (1) will be assigned to v1.

[IF, v1,2]

> If "v1" value is logic (0), skip two lines; If "v1" is logic (1) continue to the next command line.

[v2 = $1504 + 0]

> If the result of the "IF" command in the previous line is logic (1), ie v0 is greater than $1504, assign value $1504 to v2. "+0" is added in order to comply with macro line format in assignment process.

[E] -> Macro end

> That line indicates that macro is completed.

24.1.4.1 Commands

Command	Command Definition
+	Plus
-	Minus
*****	Multiply
/	Divided by
%	Modular arithmetic
&	Logical “AND” operation
|	Logical “OR” operation
^	Logical “X-OR” operation
>	Greater than
<	Less than
e	Equal to
b	Greater than or equal to
k	Less than or equal to
n	Not equal to
IF	Logical “IF”
[	Command line start
]	Command line end
E	Macro end
$	Block Addressing
v0,v1, ..	Variable

24.1.5 Sample Application

Control of 8 binary register values by the logical "and" operation written in the macro:

The macro block will only operate when the trig is active.

Macro commands:

[v0 = $3000 & $3001] -> Evaluate the registers addressed with $3000 and $3001 in the logical "and" operation and assign the result to variable 0(v0).

[v0 = v0 & $3002] -> Evaluate the registers addressed with v0 and $3002 in the logical "and" operation and assign the result to variable 0(v0).

[v0 = v0 & $3003]

[v0 = v0 & $3004]

[v0 = v0 & $3005]

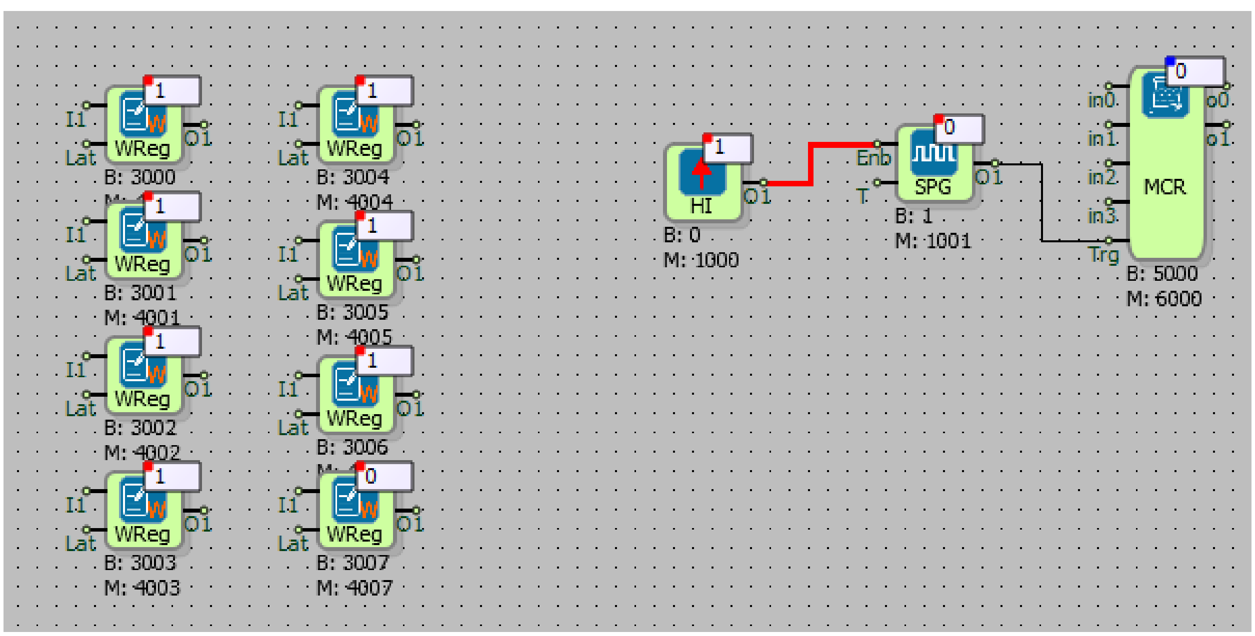
[v0 = v0 & $3006]

[v0 = v0 & $3007]

[o0 = v0 + 0] –> The value of variable 0(v0) is passed to the output of macro block o0.

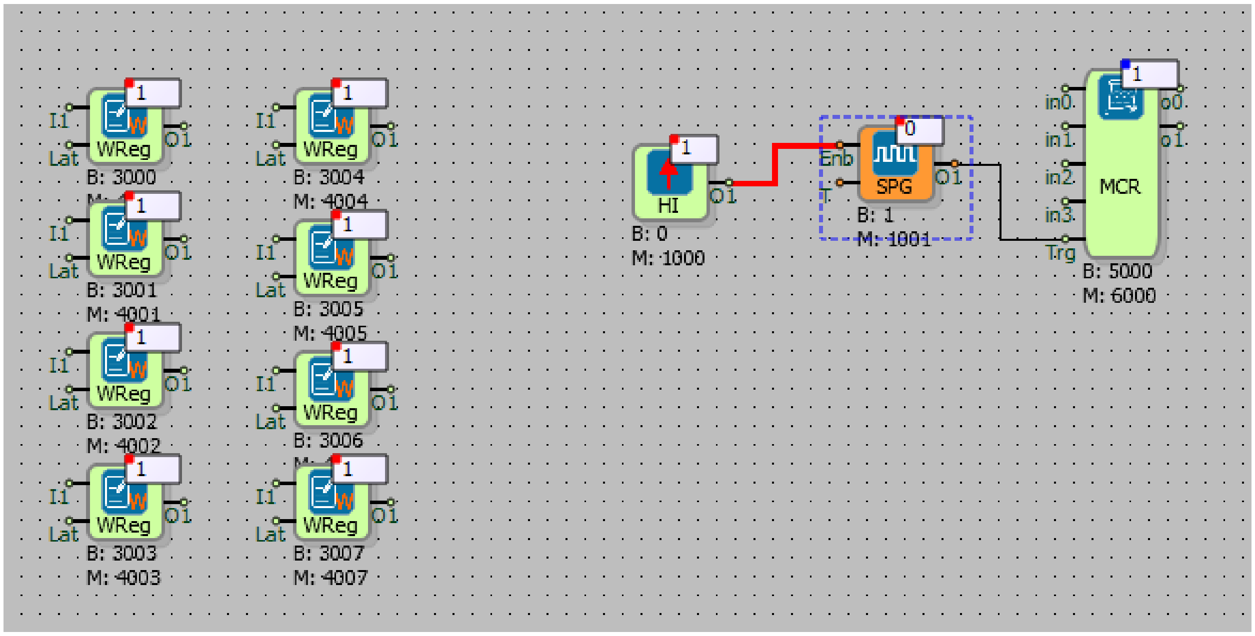
[E] –> Macro process ends.

The result of operation for $3007 register value is 0 and other register value is 1;



The definition of the AND operation is that if any of the inputs are logic (0), the output is logic (0), so the macro block output is logic (0).

Process result with all register values are logic(1):

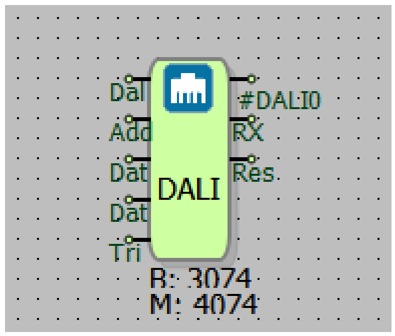


The definition of the AND operation is that if all of the inputs are logic (1), the output is logic (1), so the macro block output is logic (1).

25 DALI BLOCKS

25.1 DALI Block

25.1.1 Connection

<table>
<tr><td>Dal: Expansion Switch ID input</td><td rowspan="5">
</td><td rowspan="2">#DALI0: Light level output of the armature</td></tr>
<tr><td>Add: Armature address input</td></tr>
<tr><td>Dat: Data Byte 1 input</td><td rowspan="2">Rx: Number of messages from the armature</td></tr>
<tr><td>Dat: Data Byte 2 input</td></tr>
<tr><td>Tri: Trigger input</td><td>Res: Number of messages sent to the armature</td></tr>
</table>

25.1.2 Connection Explanations

It is used to send data packets to the DALI line that cannot be sent from the DALI manager block.

<u>Dal: Expansion Switch ID input</u>

It is the input for the Expansion Switch ID.

<u>Add: Armature address input</u>

It is the input for the armature address.

<u>Dat: Data Byte 1 input</u>

Byte 1: It represents the 1st byte of the data packet to be sent to the DALI line. (The value is sent in decimal format.)

<u>Dat: Data Byte 2 input</u>

It represents the 2nd byte of the data packet to be sent to the DALI line. (The value is sent in decimal format.)

Tri: Trigger input

This is the trigger input of the block. When a trigger is applied here, the data packet intended to be sent through the block is transmitted to the DALI line.

DALI0: Light level output of the armature

It provides the light level of the armature.

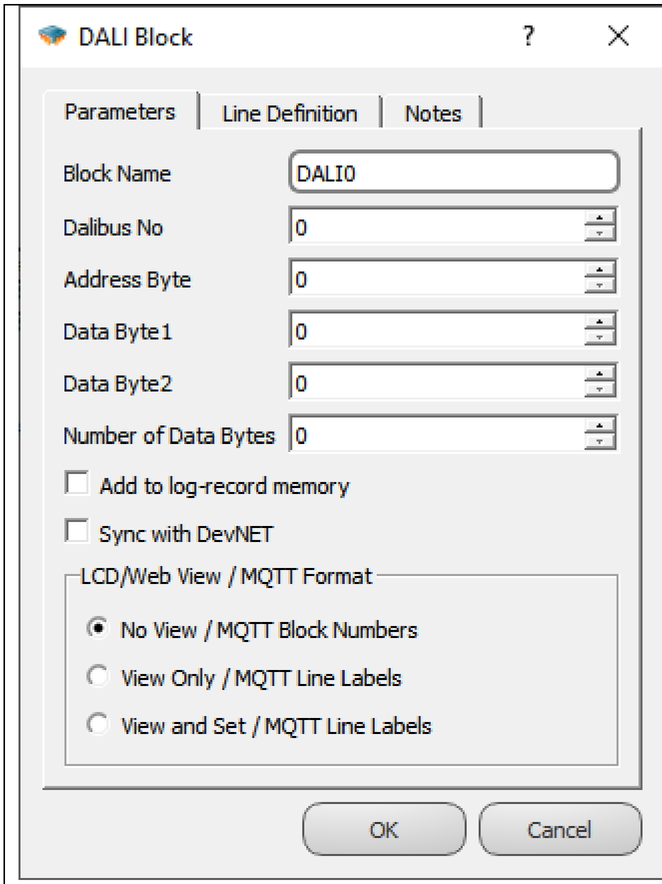
Rx: Number of messages from the armature

It is number of messages from the armature

Res: Number of messages sent to the armature

It is the number of messages sent to the armature.

25.1.3 Block Settings



Dalibus No: It is the input for the Expansion Switch ID. This value can be entered either through the block options or the block input.

Adress Byte: It is the input for the armature address. This value can be entered either through the block options or the block input.

Data Byte1: It represents the first byte of the data packet to be sent to the DALI line. This value can be entered either through the block options or the block input.

Data Byte2: It represents the second byte of the data packet to be sent to the DALI line. This value can be entered either through the block options or the block input.

Number of Data Bytes: It is used to specify how many bytes the data packet to be sent to the DALI line consists of.

Sync with DevNet: When this checkbox is selected, it sets the data outputs to 0 if no response is received from the armature.

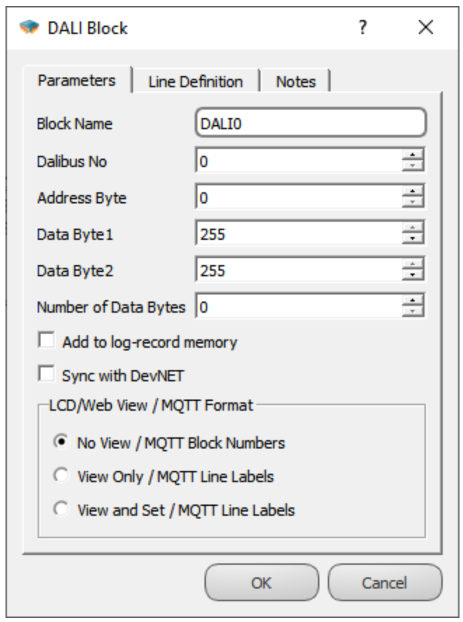
25.1.4 Block Explanation

The DALI block allows you to send data packets directly to armatures. Data packets that cannot be sent from the DALI Manager block can be parsed into bytes and transmitted to the DALI line through this block, enabling remote control of armatures. You need to enter the decimal equivalents of the data packet you want to send into the data byte sections.

Before starting operations with the Mikrodev DALI Expansion Module, configuration must be performed using a USB-DALI Converter. The configuration should include addressing, grouping, power-on, minimum and maximum level selections, and scene selections. The PLC program is written based on these addresses.

Note: Data packets sent from DALI switching elements with DALI-2 support can be read via RTU. For this purpose, the DALI block is used.

Example: Suppose a DALI switching element is connected at address 0 on DALI bus number 0. To read the data packet from this element, the special settings of the DALI block should be configured as follows:



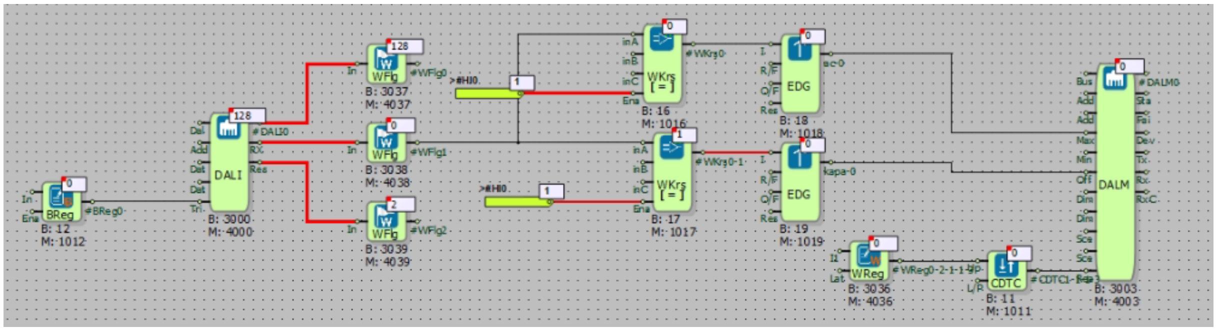
Dalibus No: 0 (The switch ID of the DALI expansion module to which the relevant switching element is connected should be entered.)

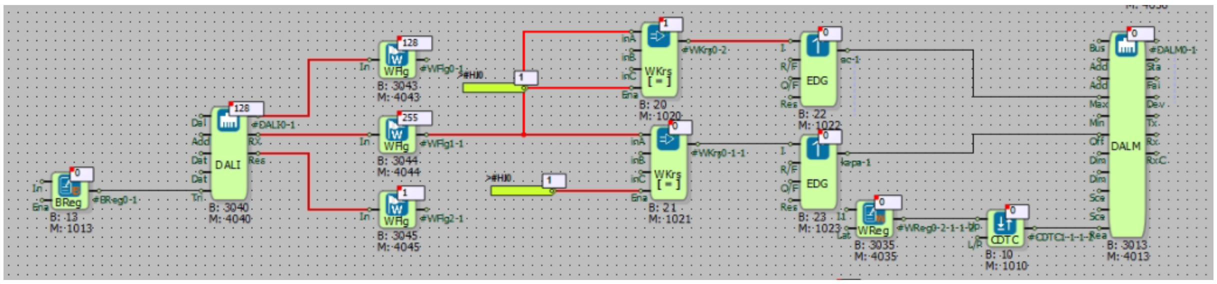
Address Byte: 0 (The address number to which the switching element is connected should be entered.)

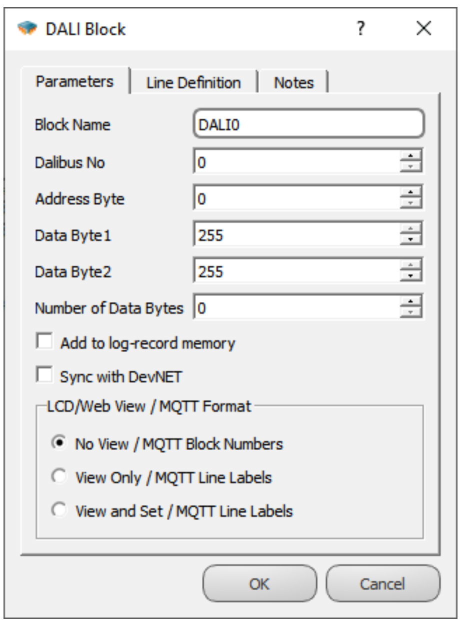
Data Byte1: 255 (To read the data packet sent from the switching element, the value 255 should be entered here.)

Data Byte2: 255 (To read the data packet sent from the switching element, enter the value 255 here.)

25.1.5 Sample Application



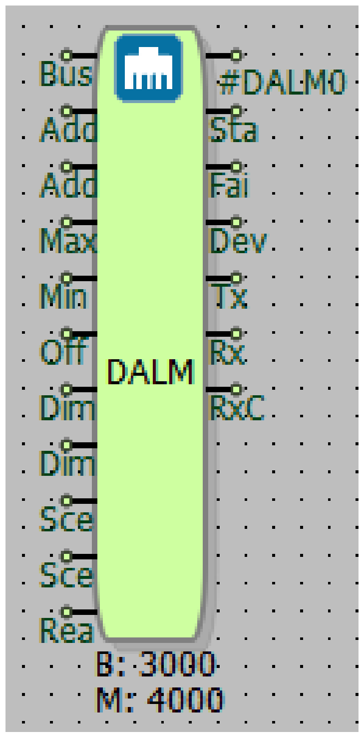




In the example application, a DALI switching element is connected to DALI bus number 0 at address 0. When the switch is in the open position, a value of 0 is read from the Rx output of the DALI block, and when it is in the closed position, a value of 255 is read. Based on these values read from the switching element, the DALI armatures in the field are controlled.

25.2 Dali Manager Block

25.2.1 Connection

Inputs	Block	Outputs
Bus: Expansion Switch ID input		#DALM0: Light level output of the armature
Add: Armature address input		Sta: Status information output of the armature
Add: Armature/Group selection		Fai: Armature alarm information output
Max: Trigger input for maximum brightness		Dev: Output for monitoring whether the armature is active
Min: Trigger input for minimum level illuminance		Tx: Number of messages sent to the armature
Off: Armature/Close group input		Rx: Number of messages from the armature
DimValue: Armature brightness value input		Rx_C: Read operation status information
DimTrig: Brightness value app trigger input		
Sce: Scene number input		
Sce: Apply scene number trigger input		
Rea: From the armature; Trigger input to read dim level, status and alarm bytes		

25.2.2 Connection Explanations

Dal: Expansion Switch ID input

It is the input for the Expansion Switch ID.

Add: Armature address input

It is the armature address input.

Add: Armature/Group selection

If this input is 0, it is treated as a armature, and if it is 1, as a group address.

Max: Trigger input for maximum brightness

When a trigger is applied to this input, the armature /group gives maximum level illumination.

Min: Trigger input for minimum level illuminance

When a trigger is applied to this input, the armature /group gives a minimum level of illumination.

Off: Armature/Close group input

When this input is triggered, armature/group closes.

Dim: Armature brightness value input

Brightness value between 85-254 is input to the armature.

DimTrig: Brightness value app trigger input

When this input is triggered, it applies the value applied to the "Dim Value" input to the armature.

Sce: Scene number input

Scene number entry. A value between 0-15 is entered.

Sce: Apply scene number trigger input

When this input is triggered, it sends the value applied to the “Scene No” input to the DALI line and the scene application is started.

Rea: Read trigger input

Read Trig: From the armature; Trigger is given to read dim level, status and alarm bytes. (ADDRESS TYPE MUST BE 0!)

Note: Updates when read trig is applied.

#DALM0: Light level output of the armature

Returns the luminous level of the armature.

Sta: Status information output of the armature

It contains the status information of the armature. By doing bit parsing, we can obtain the following information;

Bit 0: General alarm. 0 = No Error, 1 = Error

Bit 1: Lamp Fault. 0 = No Error, 1 = Error

Bit 2: Lamp Status. 0 = Off, 1 = On

Bit 3: Limit Error. 0 = No Error, 1 = If value other than Min/Max level is entered, it will be 1.

Bit 4: Fade Operation. 0 = No Fade operation 1 = Fade operation on.*

Bit 5: Reset status. 0 = Not in reset state, 1 = All variables (dim, fade, ..etc) are in initial state.

Bit 6: Address Error. 0 = Addressing done, 1 = Addressing not done.

Bit 7: Energy Cycle. 0 = Normal operation, 1 = 1 when first energized, decreases to 0 at next dim level changes etc.

*fade: It makes the armature flash slowly like an analog ramp, for example, it is the process of going from dim level 0 to 100 in 10 seconds. This 10 sec should be specified in the configuration, we do not do any extra operations on the plc.

Fai: Armature alarm information output

It contains the alarm information of the armature. By doing bit parsing, we can obtain the following information;

0. Bit: Led short circuit

1. Bit: Led open circuit

2. Bit: Load reduction

3. Bit: Load increase

4. Bit: Overcurrent protection

5. Bit: Thermal shutdown

6. Bit: Thermal overload

7. Bit: Reference Error

Dev: Output for monitoring whether the armature is active

It will be 0 when the read trigger comes, if there is a response from the armature, it will be 1, this timeout value can be adjusted with the off delay block and it can be monitored whether the armature is active or not.

Tx: Tx count ouput

Number of messages sent to the armatüre

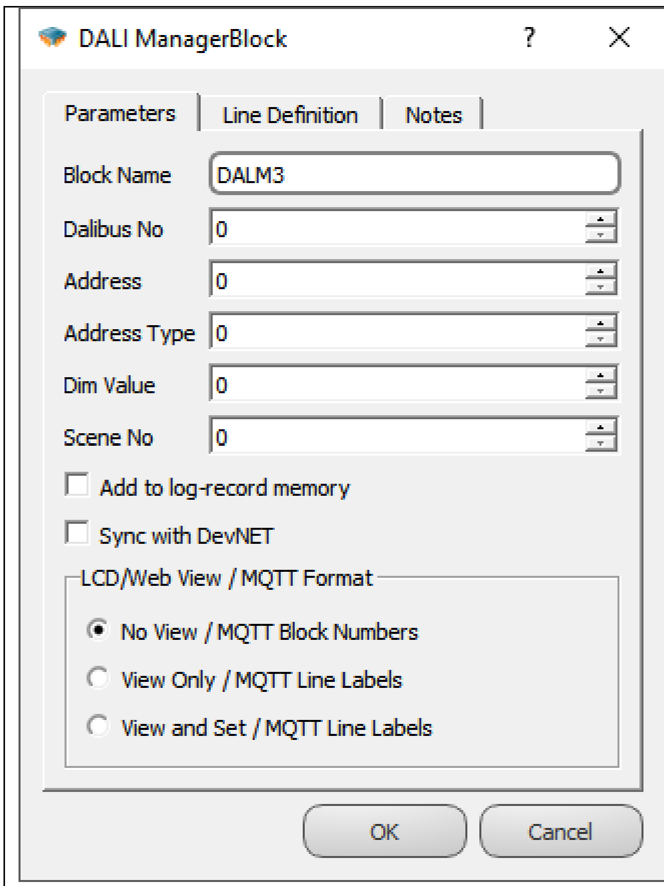
Rx: Rx count Output

Number of messages from the armatüre

Rx_C: Read operation status information

It becomes 0 during the read operation and 1 when the read operation is finished.

25.2.3 Block Settings

	Dalibus No: Expansion Switch ID entry. This value can be entered from the block options or from the block input.
	Address: It is the armature address input. This value can be entered from block options or block input
	Address Type: If it is 0, it is treated as a armature, if it is 1, as a group address. This value can be entered from the block options or from the block input.
	Dim Value: Brightness value between 85-254 is input to the armature. This value can be entered from the block options or from the block input.
	Scene No: Scene number entry. A value between 0-15 is entered. This value can be entered from the block options or from the block input.
	Sync with DevNet: When this box is checked; If there is no answer from the armature, it makes the data outputs 0.

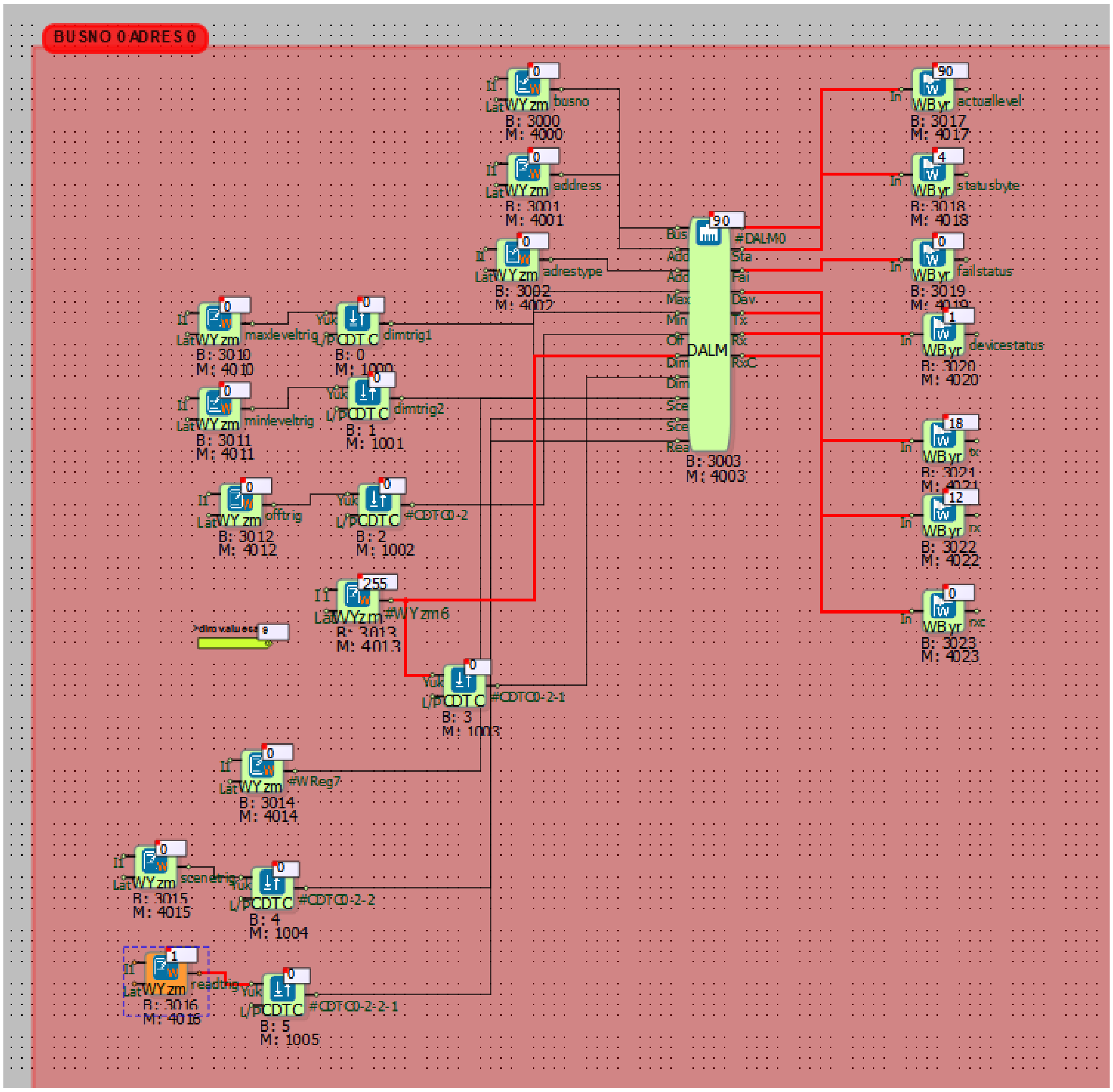
25.2.4 Block Explanation

DALI Manager block provides remote access to armatures or groups. Thanks to the DALI Manager block, we can adjust the armature brightness, read the light level of the armetures in the field, armature status information and armature error information from the block outputs.

Before starting the process with the Mikrodev DALI Expansion Module, a USB-DALI Converter must be configured. Configuration; addressing, grouping, power-on, min, max level selections and scene selections. PLC program is written according to these addresses.

You can refer to the **DALI Configurator document** for configuration details.

25.2.5 Sample Application

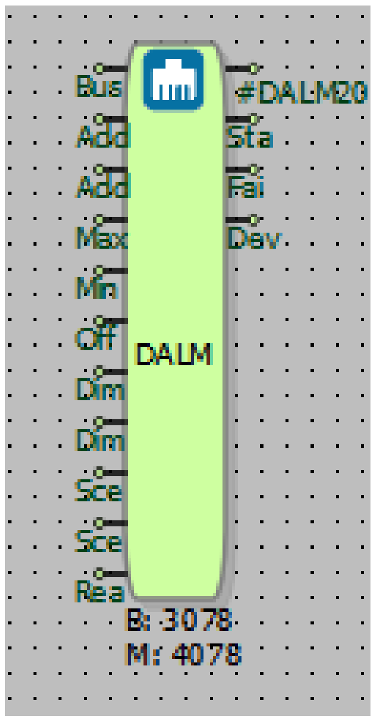


In the example application, a dim value of 90 was sent to the armature at address 0, connected to the Switch ID 0 input of the DALI expansion module, using the DALI manager block.

When the Read trig input was triggered, the DALI manager block's output read the dim value of 90.

25.3 Dali Manager Block 2

25.3.1 Connection

<table>
<tr><td>Bus: Expansion Switch ID input</td><td rowspan="11">
</td><td rowspan="4">#DALM20: Light level output of the armature</td></tr>
<tr><td>Add: Armature address input</td></tr>
<tr><td>Add: Armature /Group selection</td></tr>
<tr><td>Max: Trigger input for maximum brightness</td></tr>
<tr><td>Min: Trigger input for minimum level illuminance</td><td rowspan="2">Sta: Status information output of the armature</td></tr>
<tr><td>Off: Armature /Close group input</td></tr>
<tr><td>DimValue: Armature brightness value input</td><td rowspan="3">Fai: Armature alarm information output</td></tr>
<tr><td>DimTrig: Brightness value app trigger input</td></tr>
<tr><td>Sce: Scene number input</td></tr>
<tr><td>Sce: Apply scene number trigger input</td><td rowspan="2">Dev: Output for monitoring whether the armature is active</td></tr>
<tr><td>Rea: From the armature; Trigger input to read dim level, status and alarm bytes</td></tr>
</table>

25.3.2 Connection Explanations

Dal: Expansion Switch ID input

It is the input for the Expansion Switch ID.

Add: Armature address input

It is the armature address input.

Add: Armature/Group selection

If this input is 0, it is treated as a armature, and if it is 1, as a group address.

Max: Trigger input for maximum brightness

When a trigger is applied to this input, the armature /group gives maximum level illumination.

Min: Trigger input for minimum level illuminance

When a trigger is applied to this input, the armature /group gives a minimum level of illumination.

Off: Armature/Close group input

When this input is triggered, armature/group closes.

Dim: Armature brightness value input

Brightness value between 85-254 is input to the armature.

DimTrig: Brightness value app trigger input

When this input is triggered, it applies the value applied to the "Dim Value" input to the armature.

Sce: Scene number input

Scene number entry. A value between 0-15 is entered.

Sce: Apply scene number trigger input

When this input is triggered, it sends the value applied to the “Scene No” input to the DALI line and the scene application is started.

Rea: Read trigger input

Read Trig: From the armature; Trigger is given to read dim level, status and alarm bytes. (ADDRESS TYPE MUST BE 0!)

Note: Updates when read trig is applied.

#DALM0: Light level output of the armature

Returns the luminous level of the armature.

Sta: Status information output of the armature

It contains the status information of the armature. By doing bit parsing, we can obtain the following information;

Bit 0: General alarm. 0 = No Error, 1 = Error

Bit 1: Lamp Fault. 0 = No Error, 1 = Error

Bit 2: Lamp Status. 0 = Off, 1 = On

Bit 3: Limit Error. 0 = No Error, 1 = If value other than Min/Max level is entered, it will be 1.

Bit 4: Fade Operation. 0 = No Fade operation 1 = Fade operation on.*

Bit 5: Reset status. 0 = Not in reset state, 1 = All variables (dim, fade, ..etc) are in initial state.

Bit 6: Address Error. 0 = Addressing done, 1 = Addressing not done.

Bit 7: Energy Cycle. 0 = Normal operation, 1 = 1 when first energized, decreases to 0 at next dim level changes etc.

*fade: It makes the armature flash slowly like an analog ramp, for example, it is the process of going from dim level 0 to 100 in 10 seconds. This 10 sec should be specified in the configuration, we do not do any extra operations on the plc.

Fai: Armature alarm information output

It contains the alarm information of the armature. By doing bit parsing, we can obtain the following information;

0. Bit: Led short circuit

1. Bit: Led open circuit

2. Bit: Load reduction

3. Bit: Load increase

4. Bit: Overcurrent protection

5. Bit: Thermal shutdown

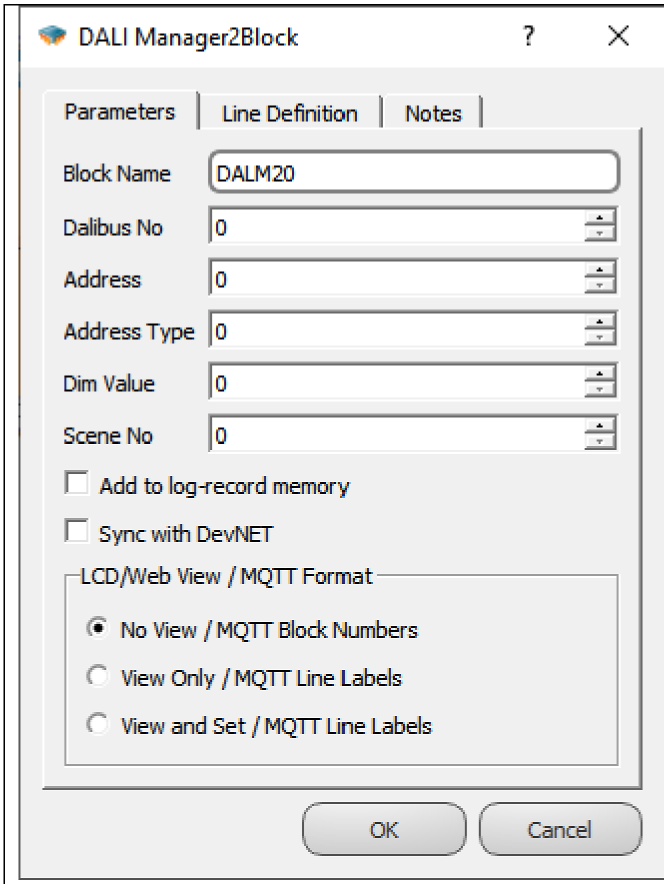
6. Bit: Thermal overload

7. Bit: Reference Error

Dev: Output for monitoring whether the armature is active

It will be 0 when the read trigger comes, if there is a response from the armature, it will be 1, this timeout value can be adjusted with the off delay block and it can be monitored whether the armature is active or not.

25.3.3 Block Settings



Dalibus No: Expansion Switch ID entry. This value can be entered from the block options or from the block input.
Address: It is the armature address input. This value can be entered from block options or block input
Address Type: If it is 0, it is treated as a armature, if it is 1, as a group address. This value can be entered from the block options or from the block input.
Dim Value: Brightness value between 85-254 is input to the armature. This value can be entered from the block options or from the block input.
Scene No: Scene number entry. A value between 0-15 is entered. This value can be entered from the block options or from the block input.
Sync with DevNet: When this box is checked; If there is no answer from the armature, it makes the data outputs 0.

25.3.4 Block Explanation

DALI Manager block provides remote access to armatures or groups. Thanks to the DALI Manager block, we can adjust the armature brightness, read the light level of the armatures in the field, armature status information and luminaire error information from the block outputs.

Before starting the process with the Mikrodev DALI Expansion Module, a USB-DALI Converter must be configured. Configuration; addressing, grouping, power-on, min, max level selections and scene selections. PLC program is written according to these addresses.

You can refer to the **DALI Configurator document** for configuration details.