An enhanced communication scheme for 4DIAC

Georgios Sfiris, Researcher
Georgios Hassapis, Supervisor Professor

Laboratory of Computer System Architecture
Department of Electrical and Computer Engineering
Aristotle University of Thessaloniki

An enhanced communication scheme for 4DIAC

In this presentation two subjects will be dealt:

- The implementation of the Modbus protocol using IEC 61499 standard's Communication Function Blocks.
- An automated scheme for the insertion of PUBLISH and SUBSCRIBE Function Blocks for the FBDK – UDP/IP protocol.

IEC 61499 Communication Framework

Modbus protocol offers a Master/Slave communication scheme giving the advantage of deterministic operation and low bandwidth usage.

But it is much more complex to implement than other simpler communication protocols, such as the UDP/IP protocol.

Modbus Protocol

Modbus Slave

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		
…				

Each Modbus Slave has a data table in its memory containing data of four different data types.

Modbus Protocol

Modbus Master

Modbus Slave #1

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

Modbus Slave #2

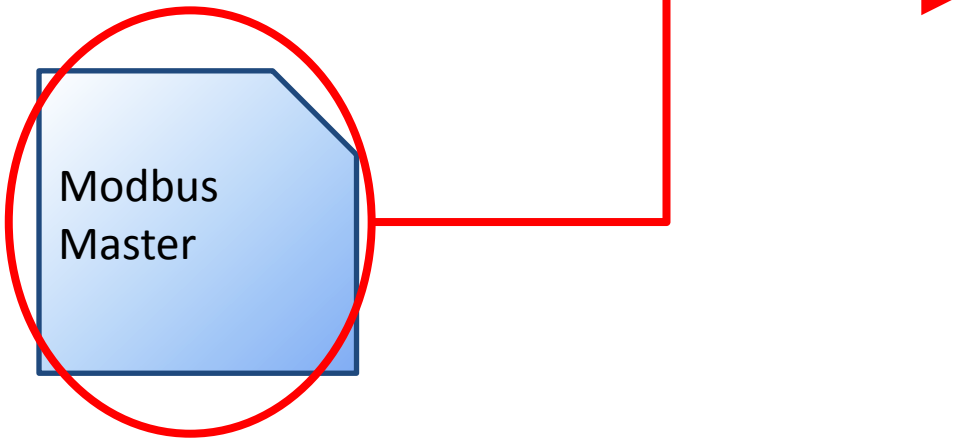
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1		12	
#2	1		15	
#3	0		256	
#4	0			
...				

Modbus Slave #3

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

In each Modbus network only one Master can exist, but several Slaves can be connected.

Modbus Protocol



Modbus
Slave #1

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

Modbus
Slave #2

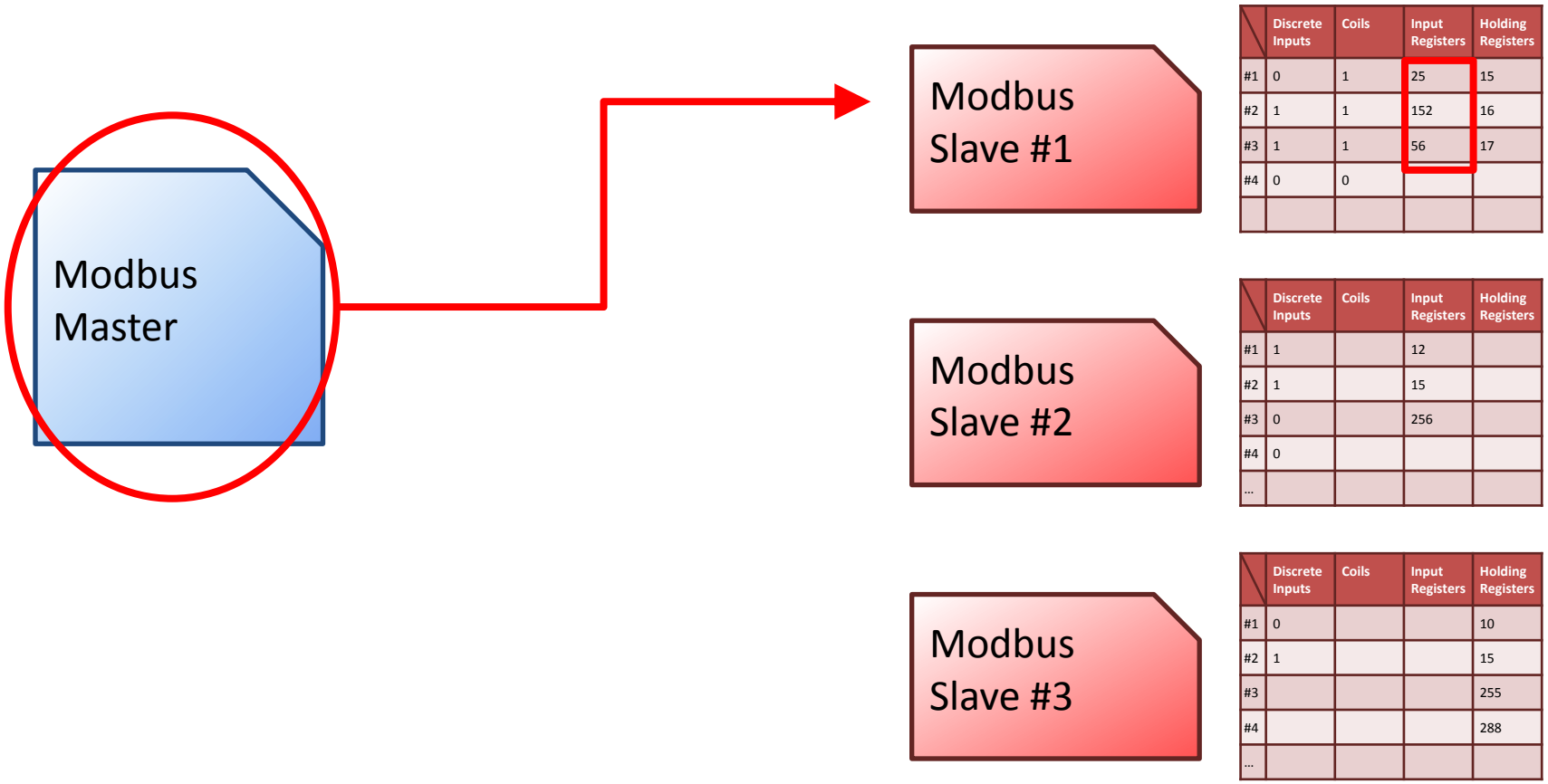
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1		12	
#2	1		15	
#3	0		256	
#4	0			
...				

Modbus
Slave #3

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

The Master will send requests to the Slaves …

Modbus Protocol



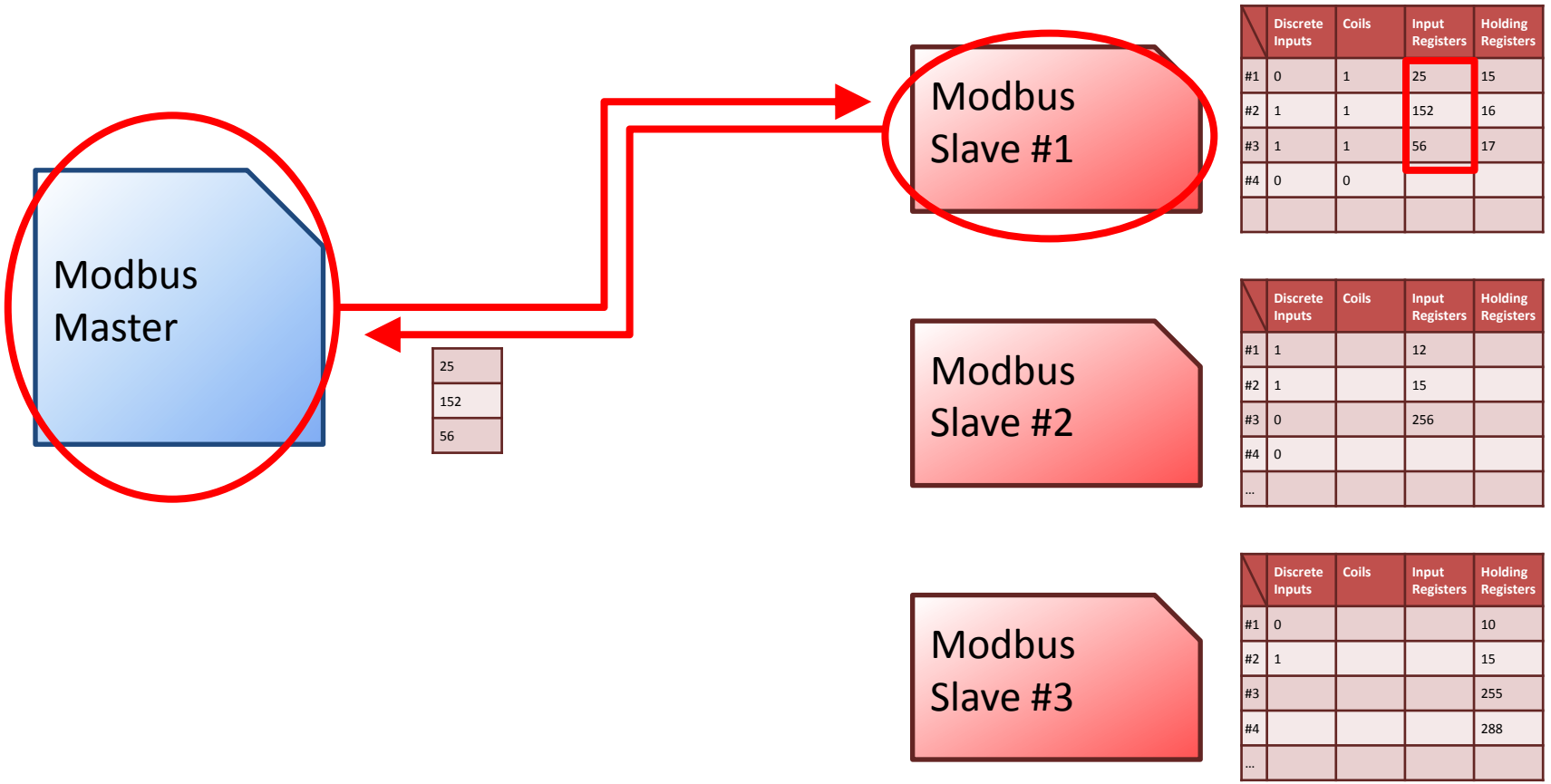
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1		12	
#2	1		15	
#3	0		256	
#4	0			
...				

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

The Master will send requests to the Slaves asking for some particular data …

Modbus Protocol



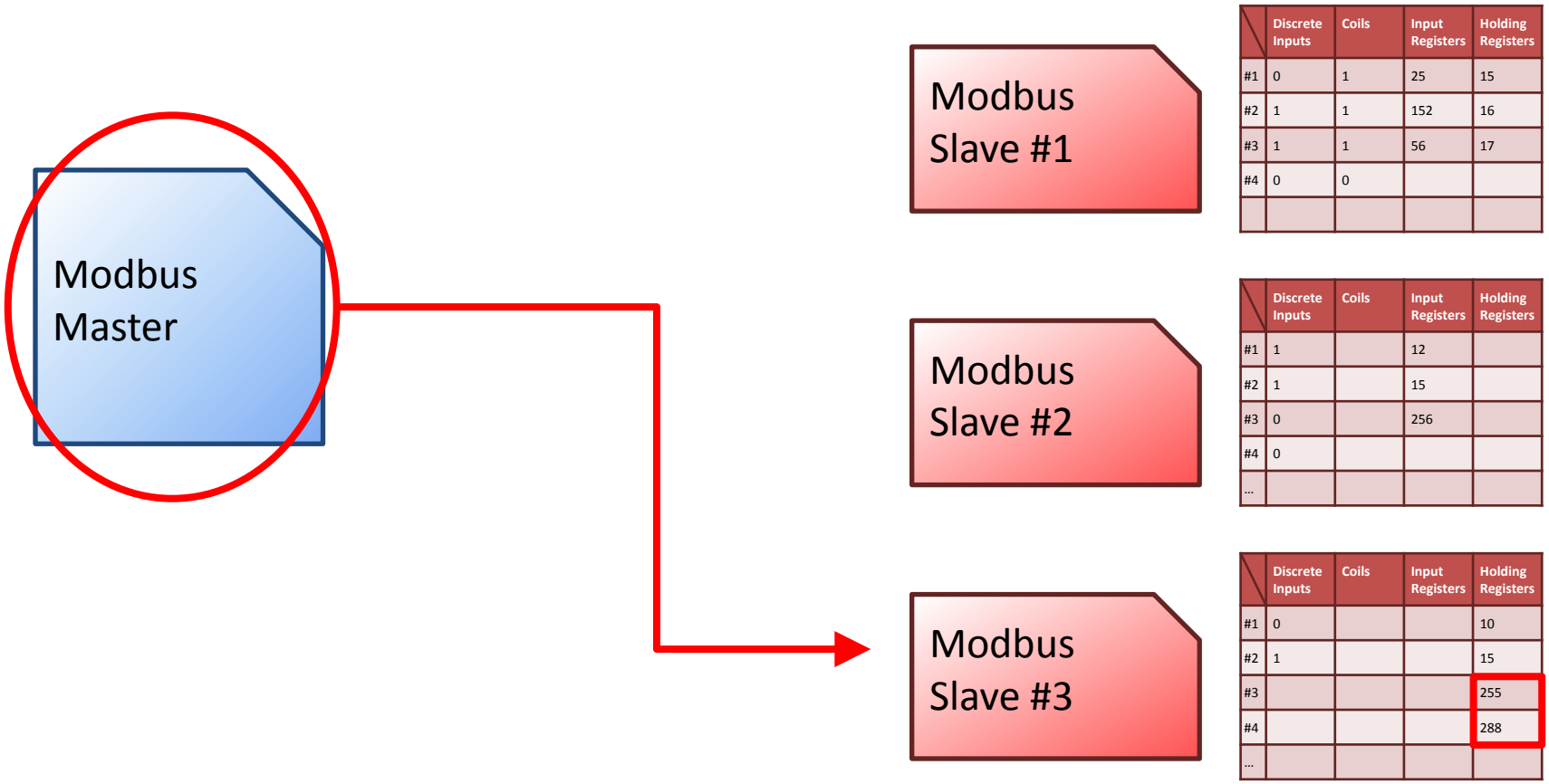
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1		12	
#2	1		15	
#3	0		256	
#4	0			
...				

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

The Master will send requests to the Slaves asking for some particular data and the Slaves will serve the request.

Modbus Protocol



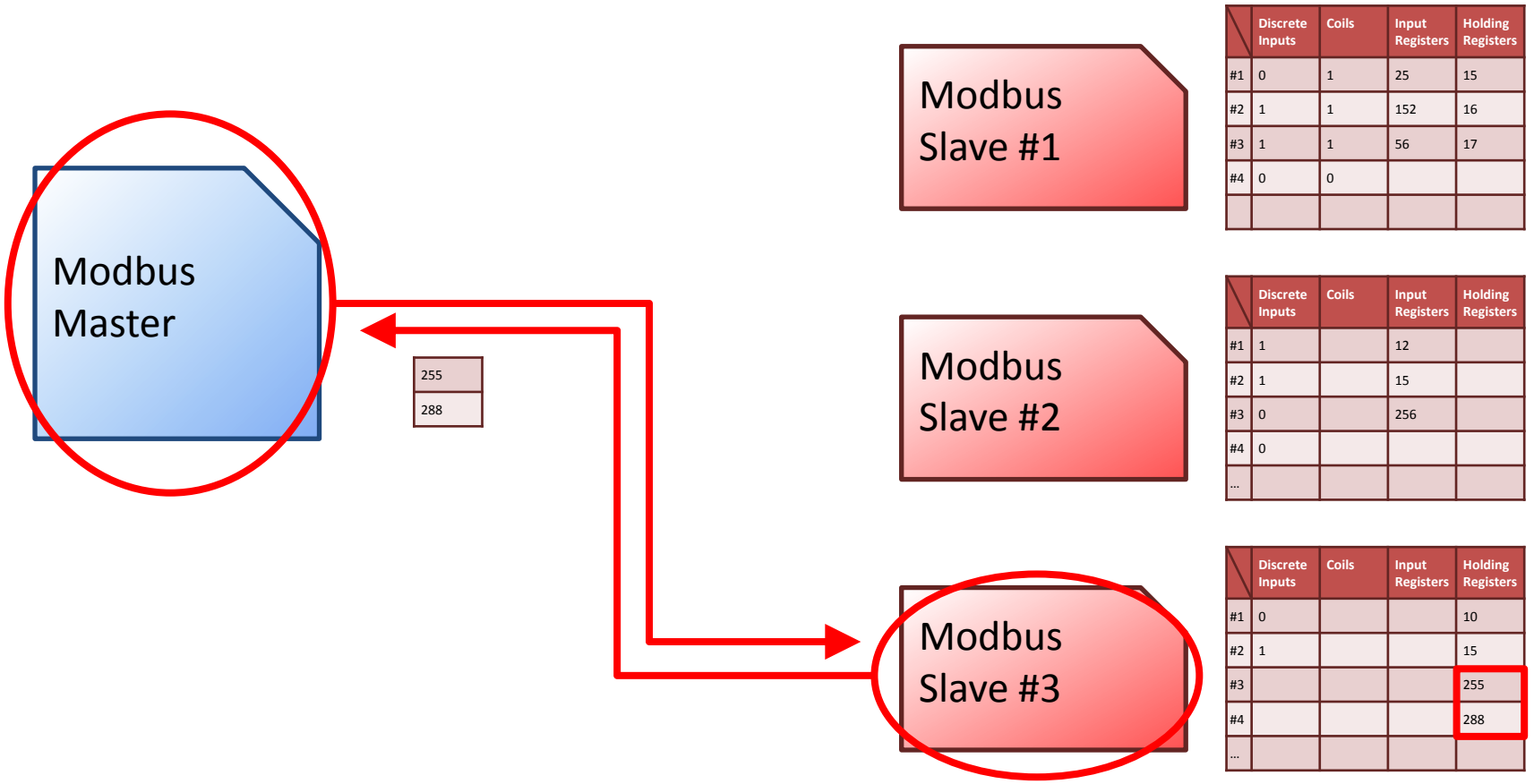
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1		12	
#2	1		15	
#3	0		256	
#4	0			
...				

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

The Master will send requests to the Slaves asking for some particular data and the Slaves will serve the request.

Modbus Protocol



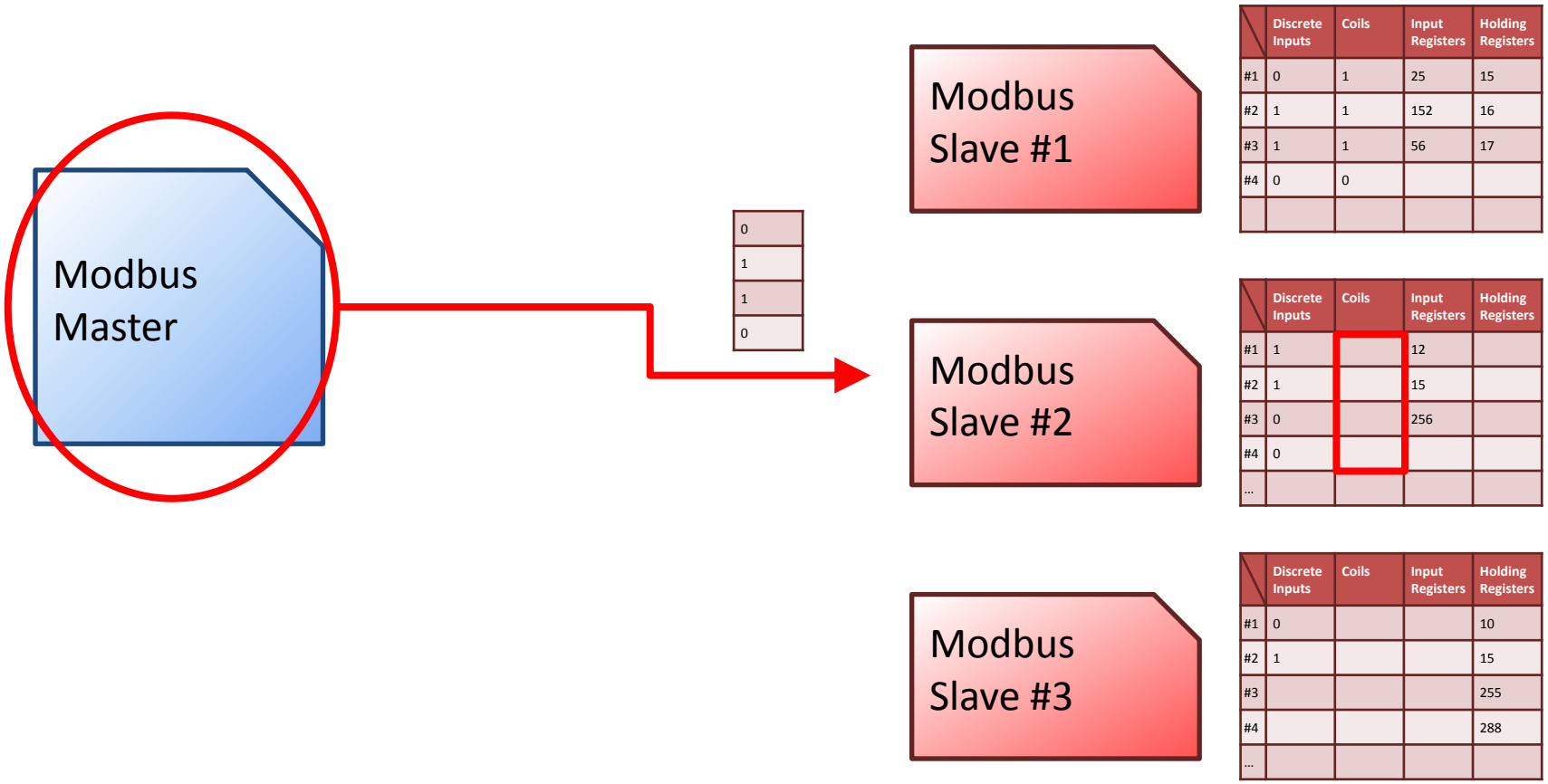
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1		12	
#2	1		15	
#3	0		256	
#4	0			
...				

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

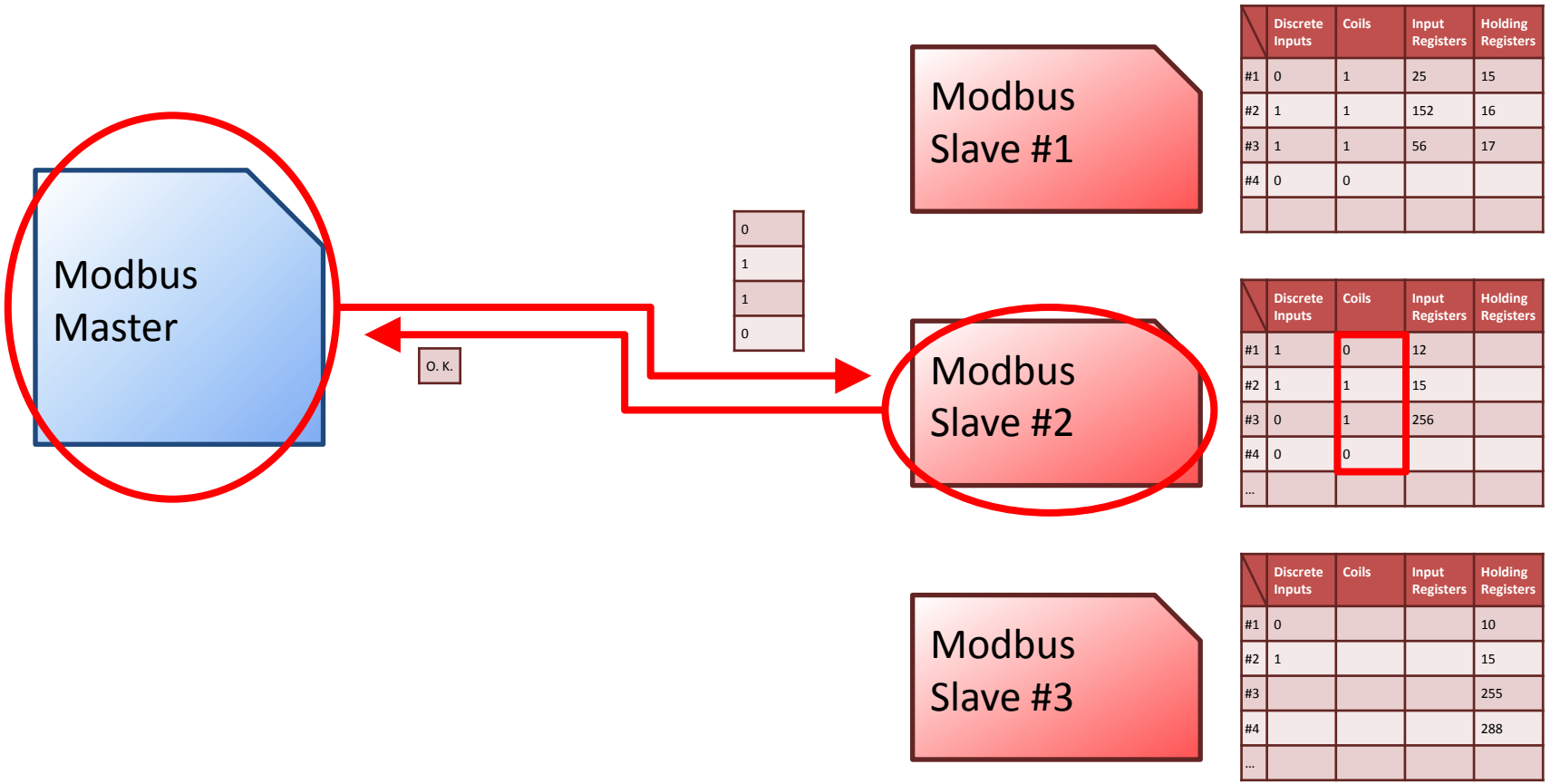
The Master will send requests to the Slaves asking for some particular data and the Slaves will serve the request.

Modbus Protocol



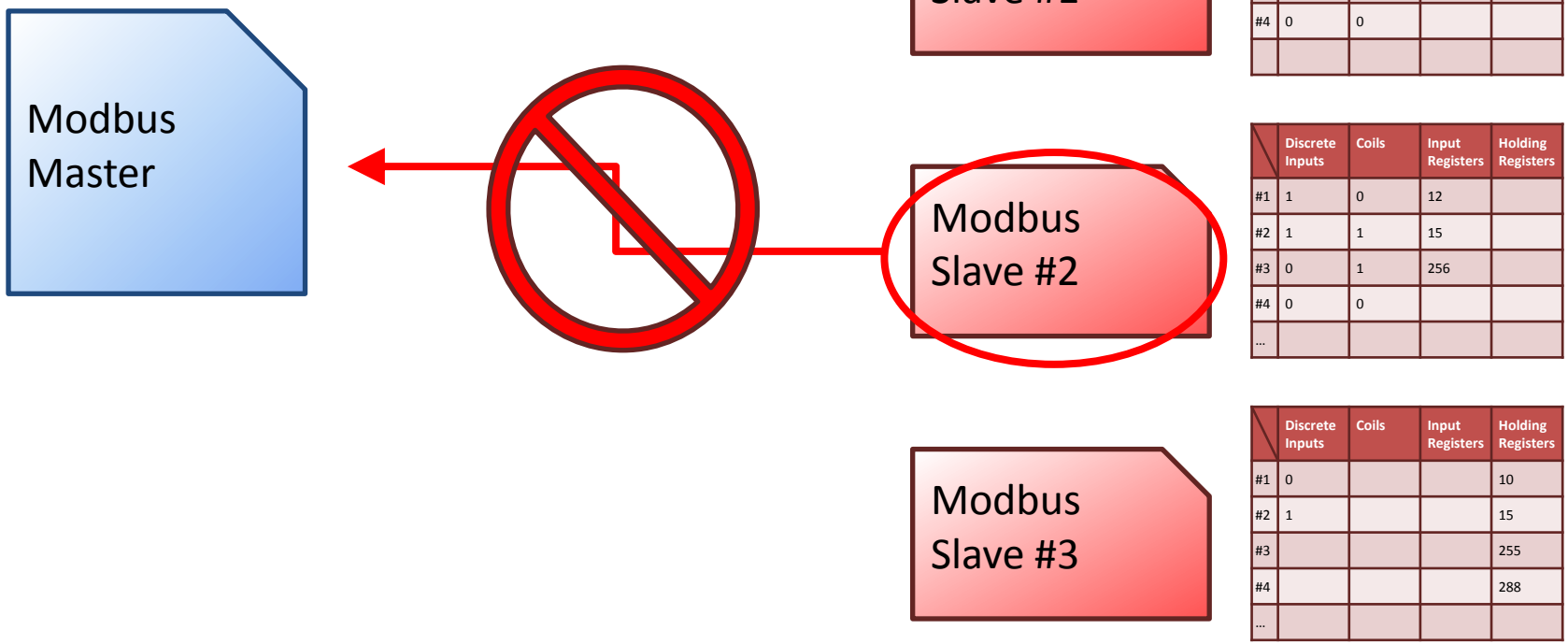
The Master will send requests to the Slaves asking for some particular data and the Slaves will serve the request.

Modbus Protocol



The Master will send requests to the Slaves asking for some particular data and the Slaves will serve the request.

Modbus Protocol



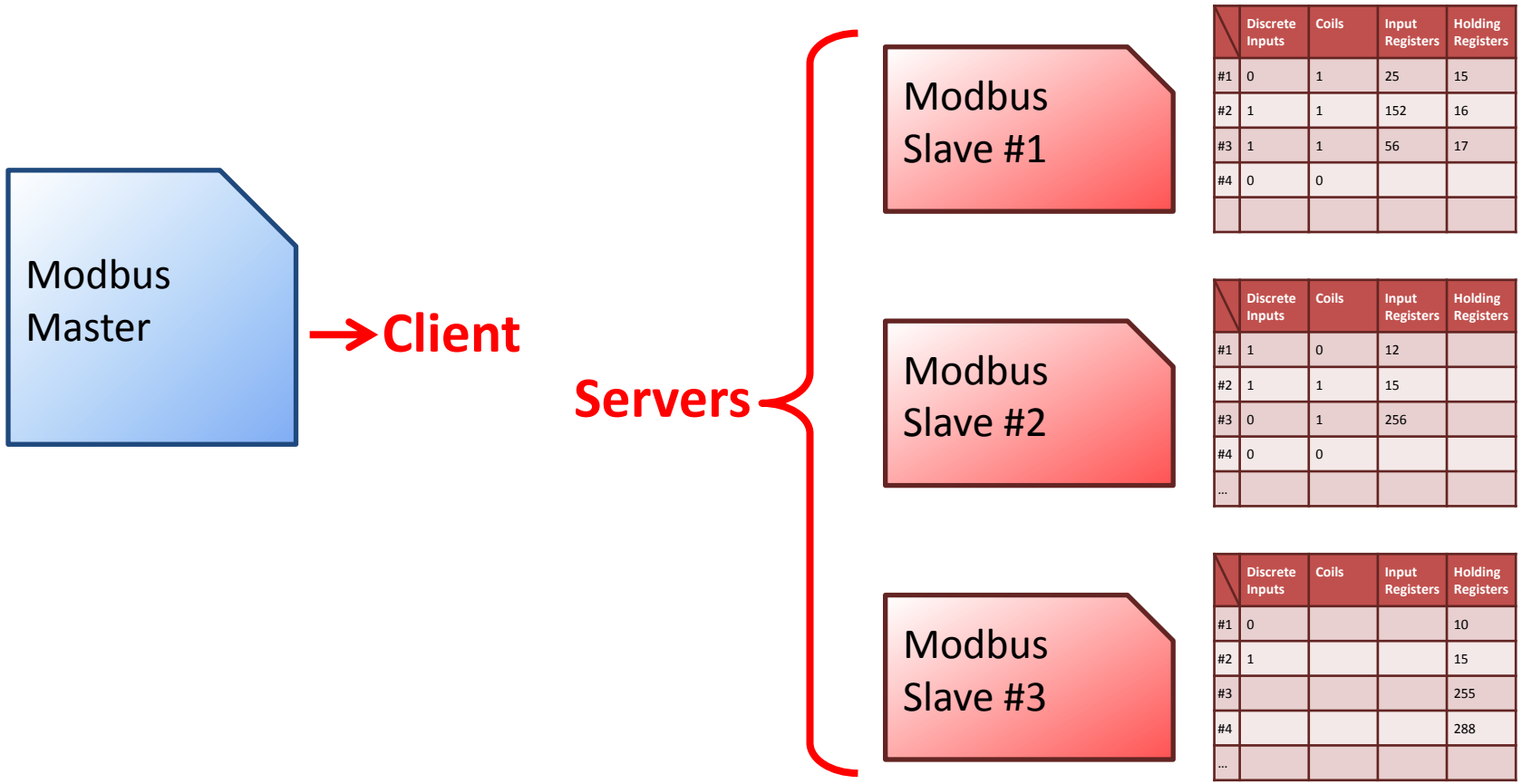
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1	0	12	
#2	1	1	15	
#3	0	1	256	
#4	0	0		
...				

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

Modbus Slaves can only send messages to the network if they are asked by the Master.

Modbus Protocol



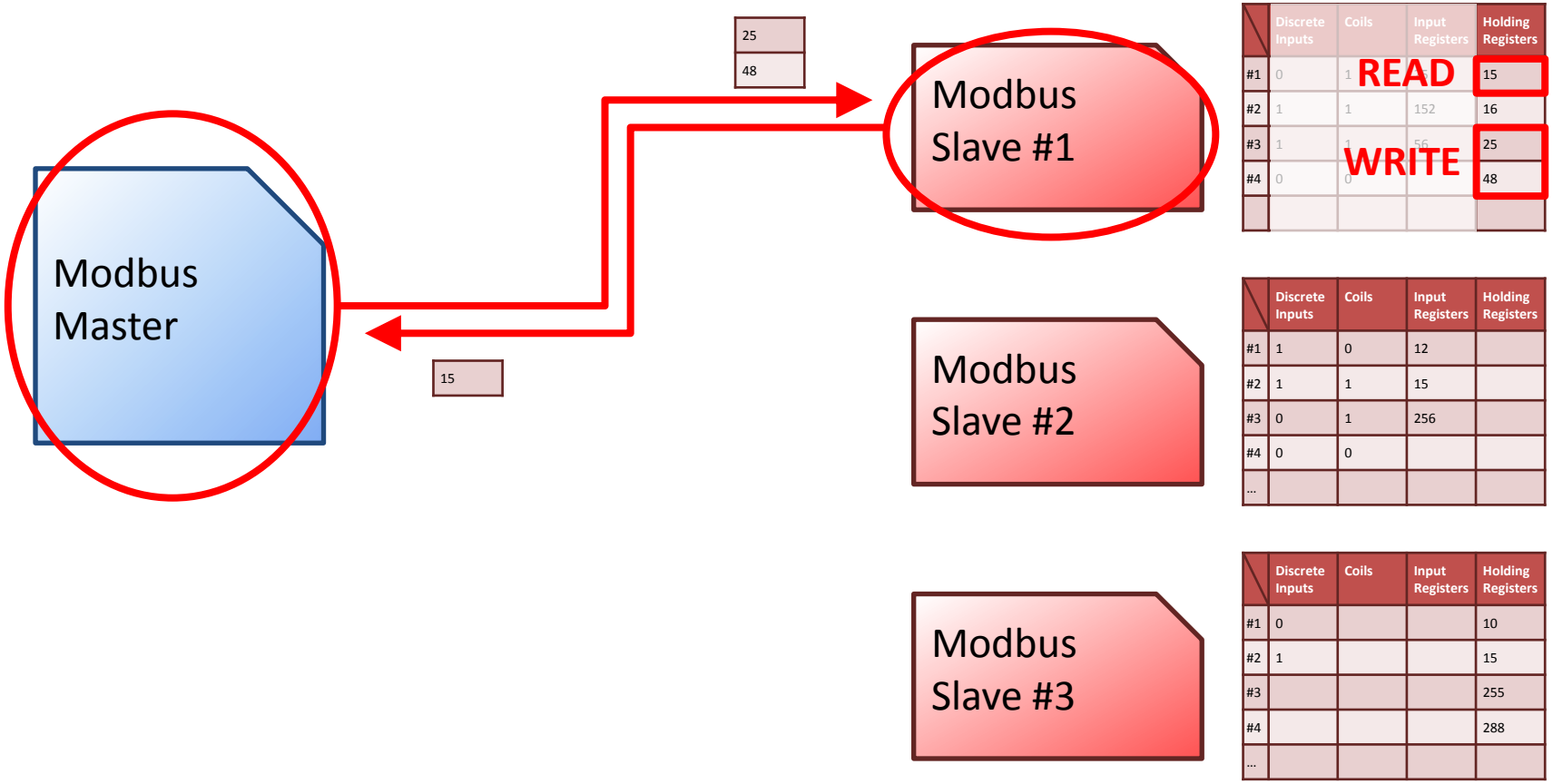
	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0	1	25	15
#2	1	1	152	16
#3	1	1	56	17
#4	0	0		

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	1	0	12	
#2	1	1	15	
#3	0	1	256	
#4	0	0		
...				

	Discrete Inputs	Coils	Input Registers	Holding Registers
#1	0			10
#2	1			15
#3				255
#4				288
...				

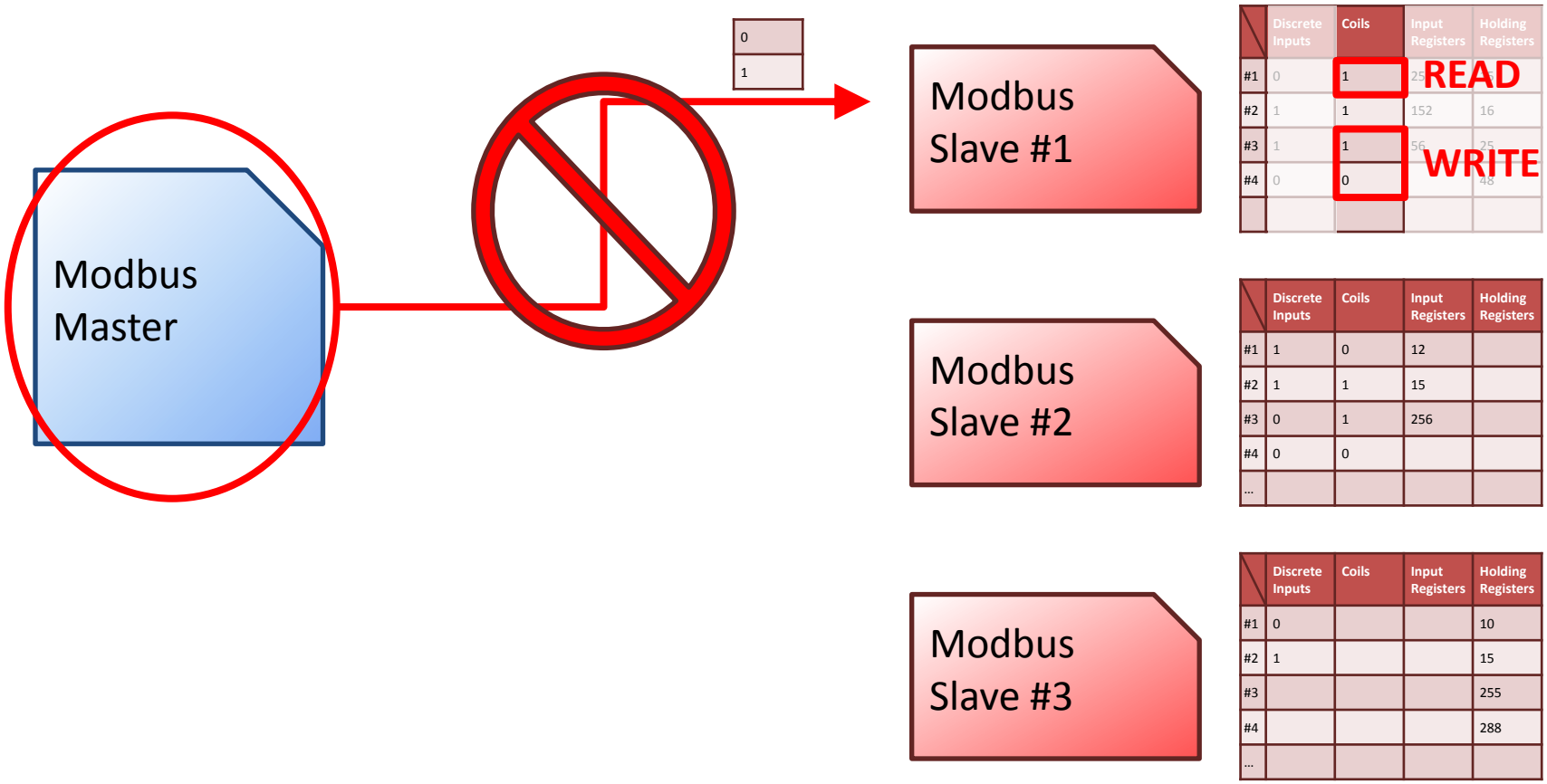
Thus usually Modbus Masters are implemented as Clients making requests to Modbus Slaves implemented as Servers.

Modbus Protocol



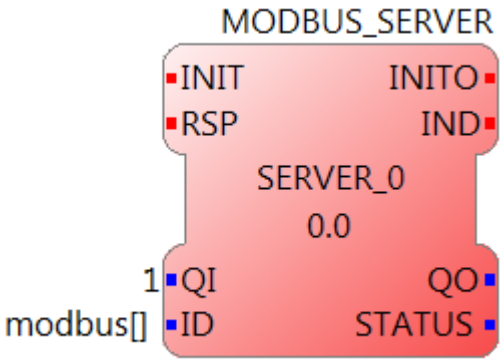
However the Modbus Master can only send a Read/Write request for Holding Register data.

Modbus Protocol



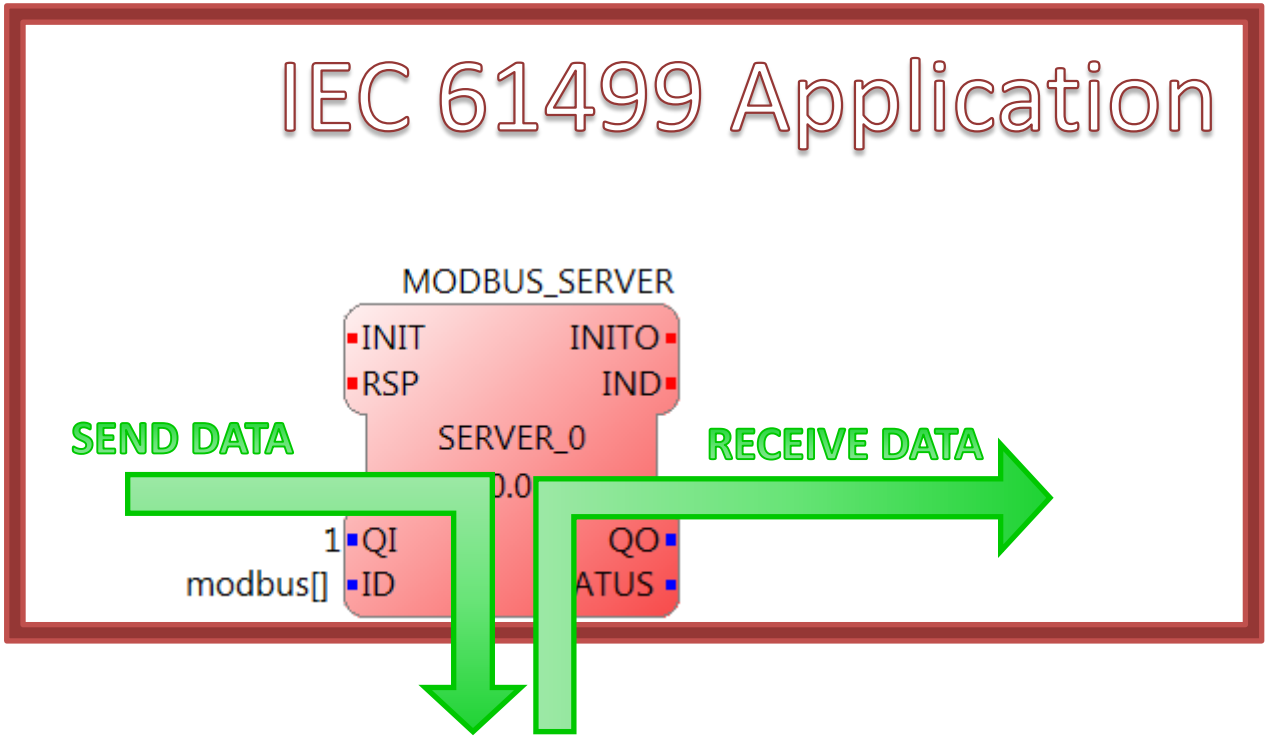
However the Modbus Master can only send a Read/Write request for Holding Register data.

Modbus Protocol in 4DIAC



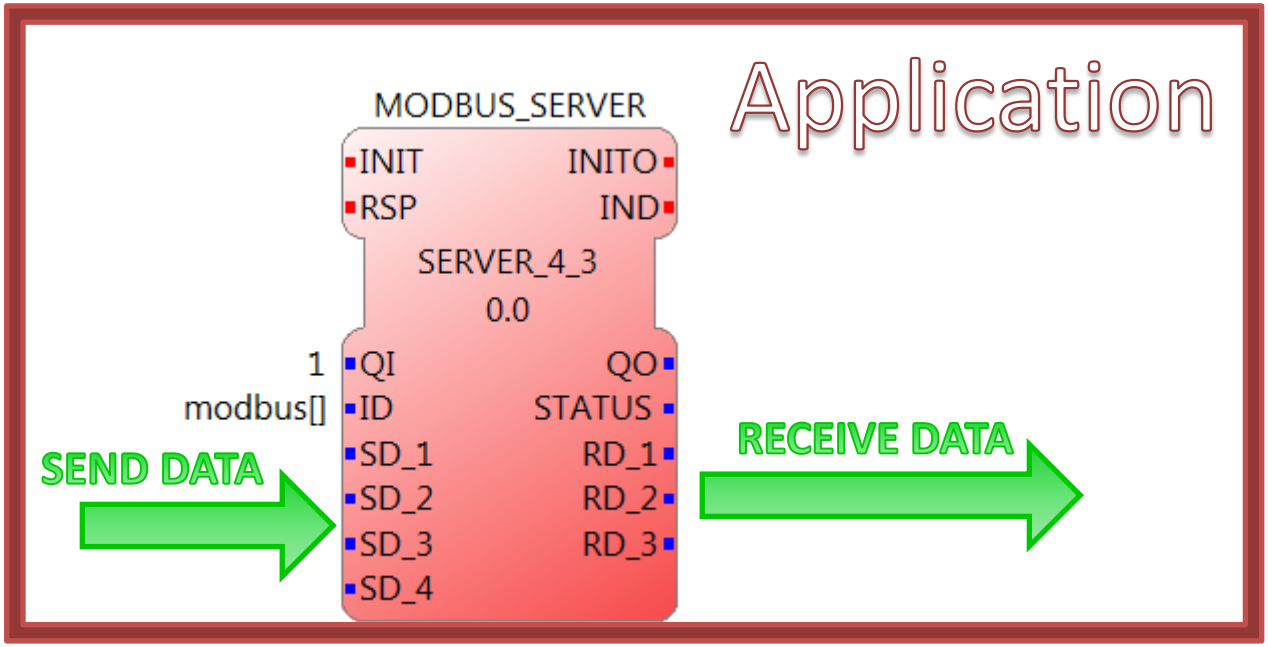
Trying to implement a Modbus Slave in IEC 61499 comes to the obvious solution of a single SERVER Function Block.

Modbus Protocol in 4DIAC



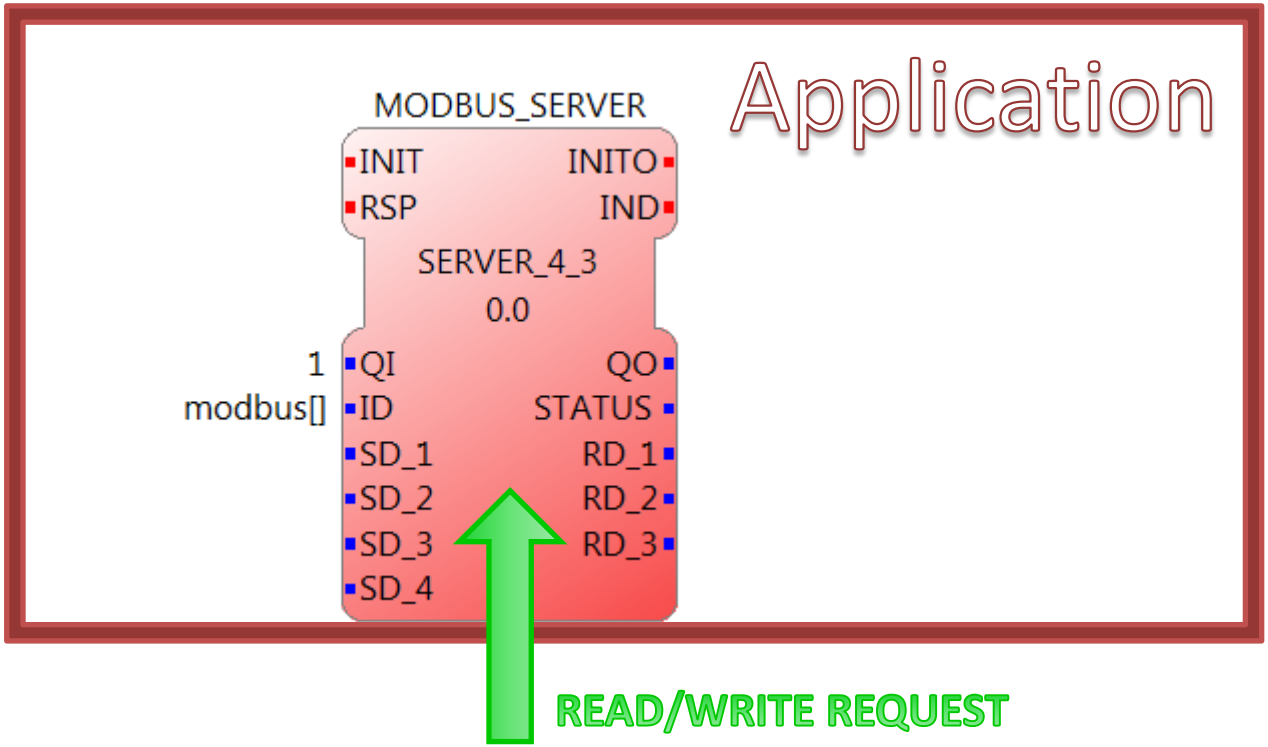
But having a single SERVER FB means that all Modbus input and output of the device is done by that single SERVER FB.

Modbus Protocol in 4DIAC



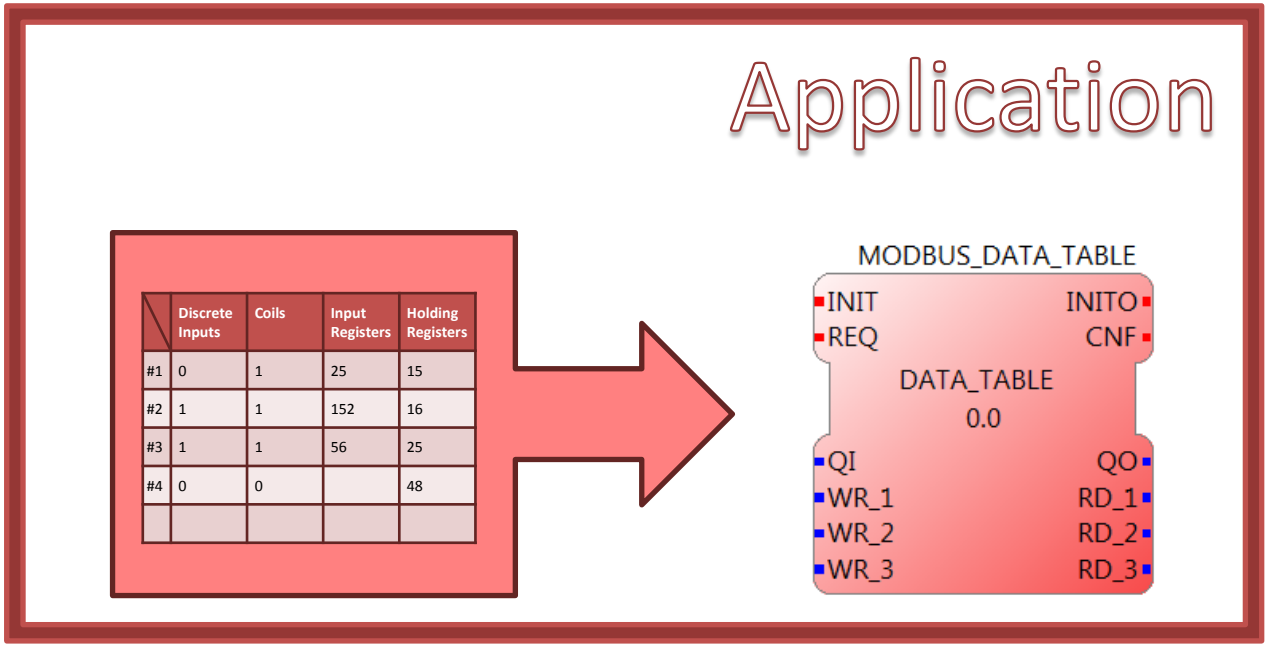
So we would like to add to that single SERVER FB all the inputs and outputs of our application.

Modbus Protocol in 4DIAC



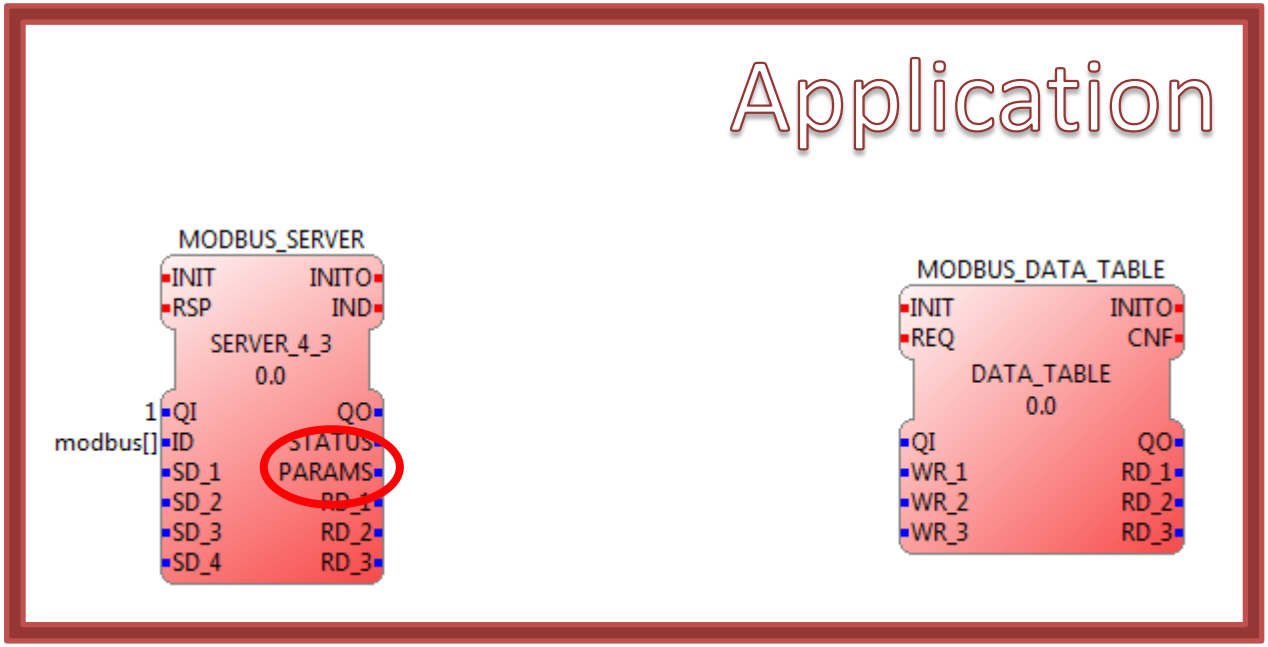
But then we would expect that device to receive only mixed Read/Write requests. This limits the Modbus data type to Holding Registers.

Modbus Protocol in 4DIAC



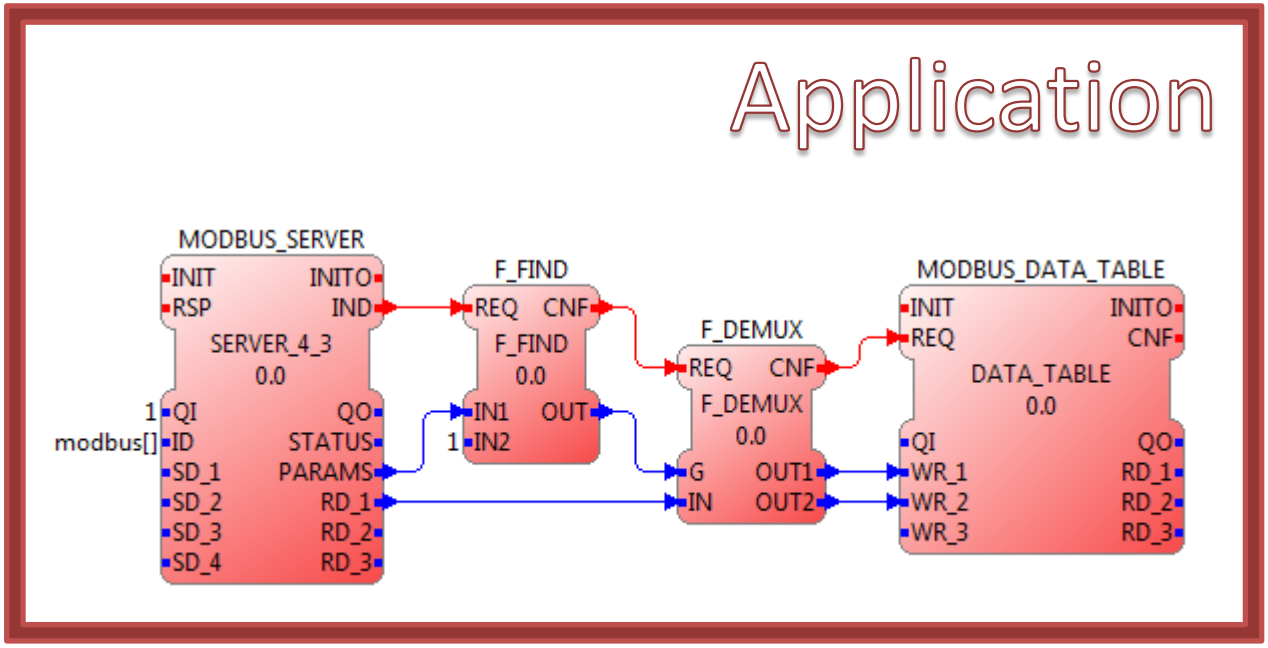
Moreover in such implementation the Modbus data table would be implemented inside the IEC 61499 Application's Function Block network as a special Function Block.

Modbus Protocol in 4DIAC



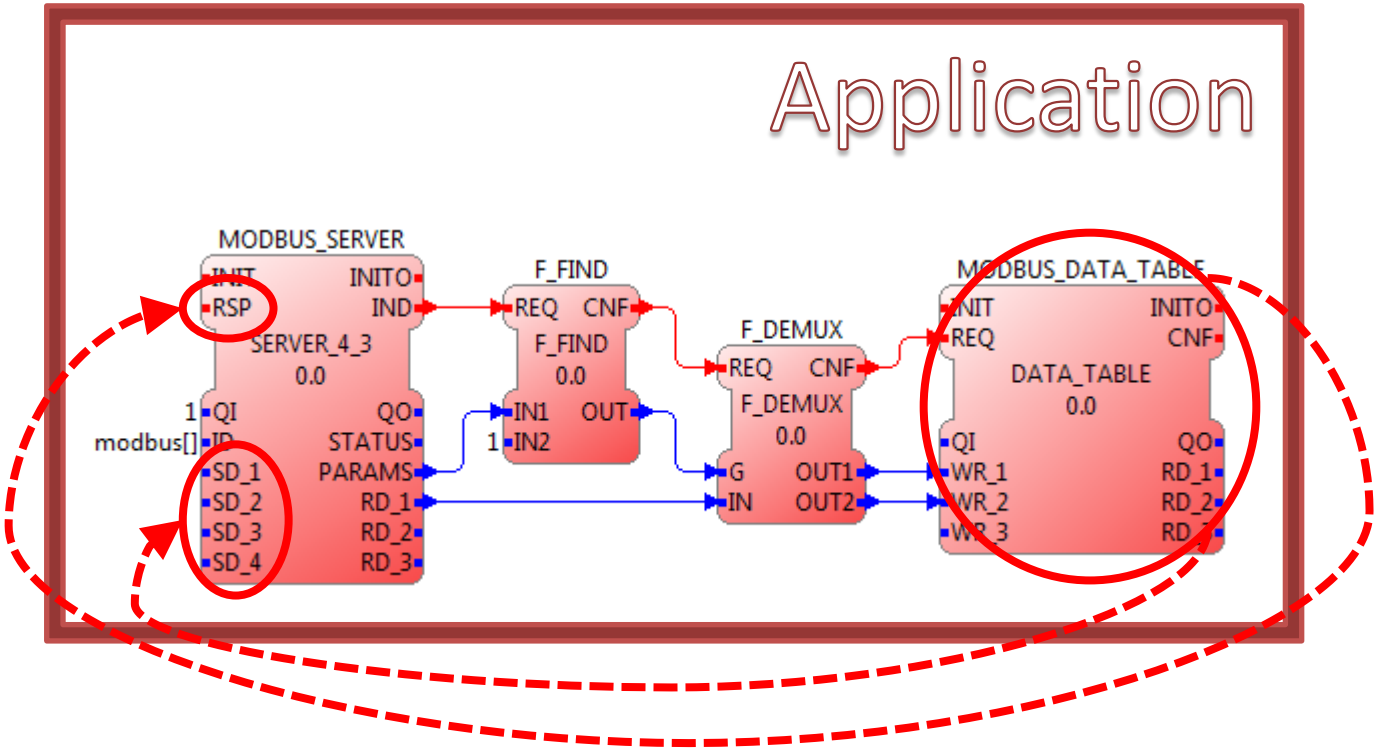
Modbus messages would enter the Function Block network as parameters in some string format.

Modbus Protocol in 4DIAC



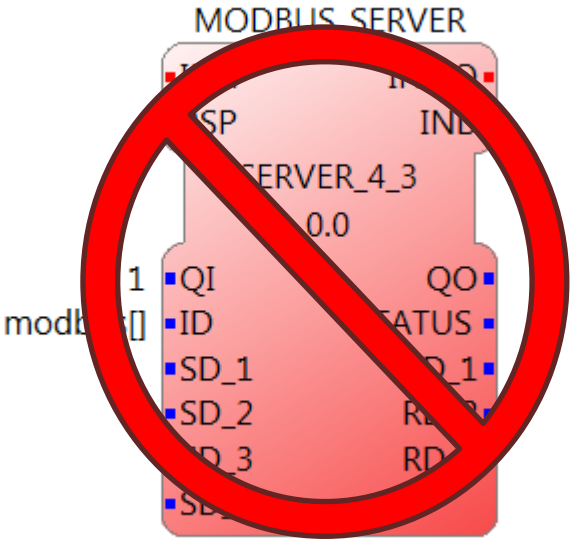
And finally the Application would interpret the parameters and connect the MODBUS_SERVER with the MODBUS_DATA_TABLE using a complex Function Block network.

Modbus Protocol in 4DIAC



Not to mention that another complex Function Block network would connect the MODBUS_DATA_TABLE back to the MODBUS_SERVER to take care of the MODBUS_SERVER responses.

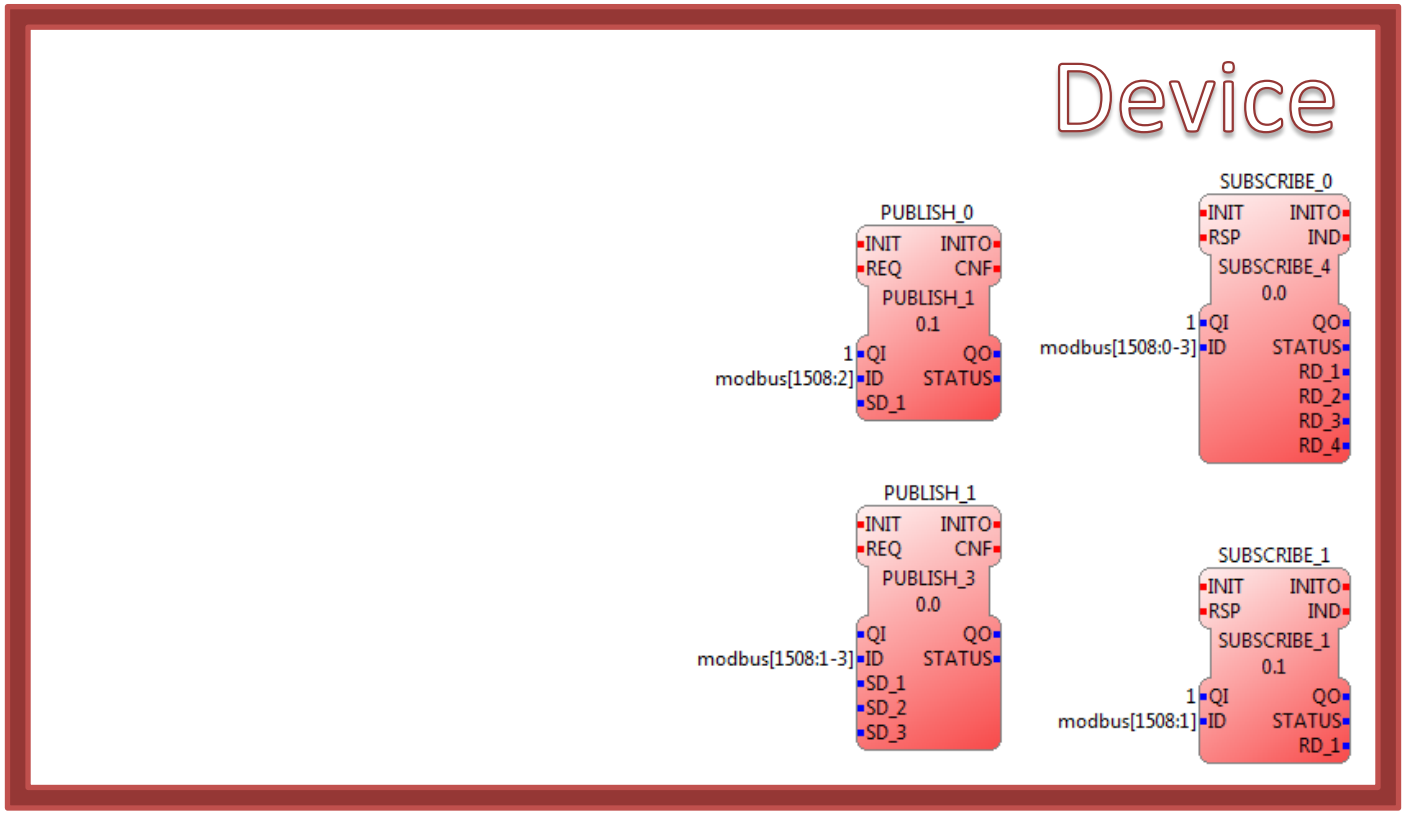
Modbus Protocol in 4DIAC



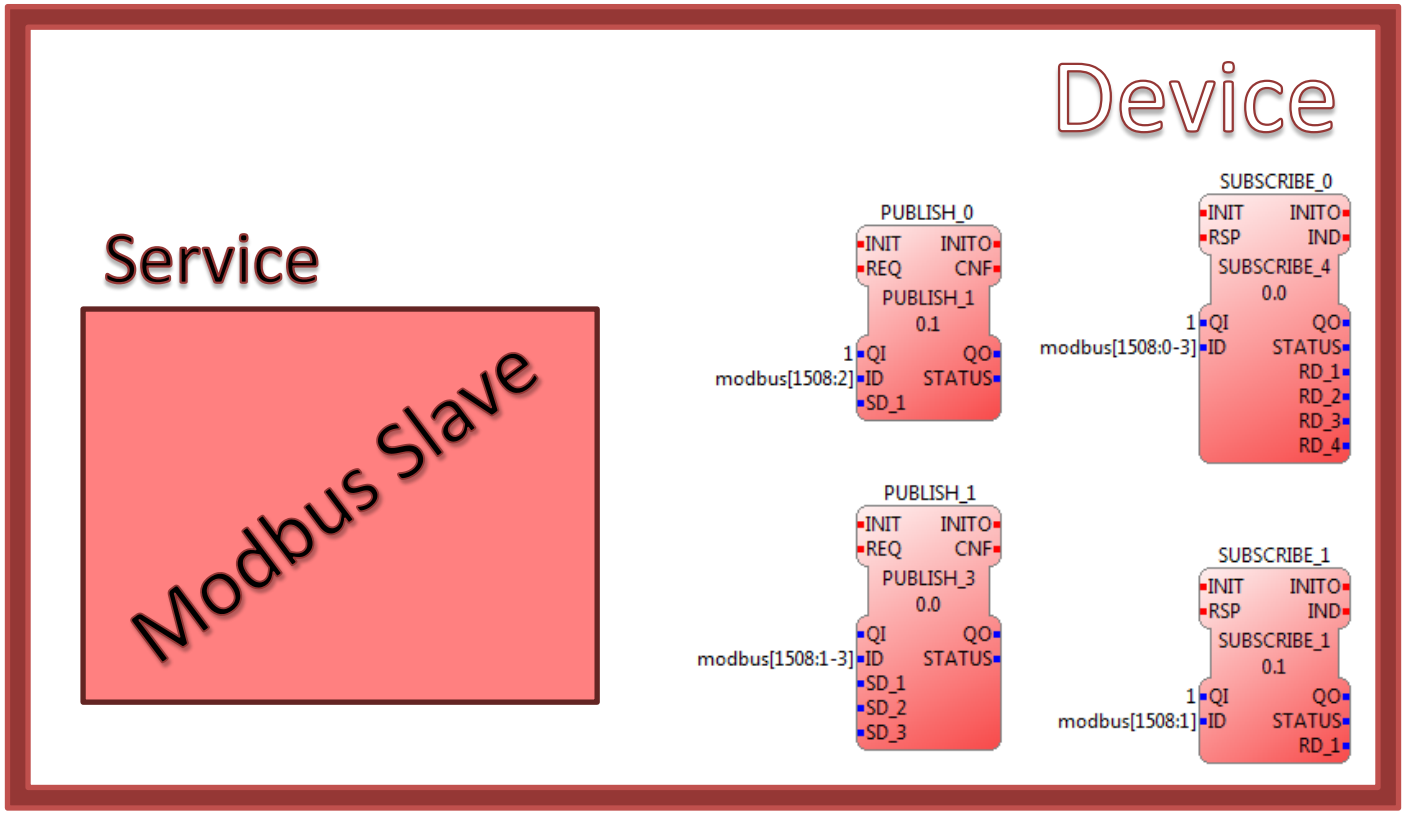
All these add to the complexity of the implementation.

So trying to implement a Modbus Slave in IEC 61499 using a single SERVER Function Block is very cumbersome and limiting. Instead PUBLISH and SUBSCRIBE Function Blocks can be used.

Modbus Protocol in 4DIAC

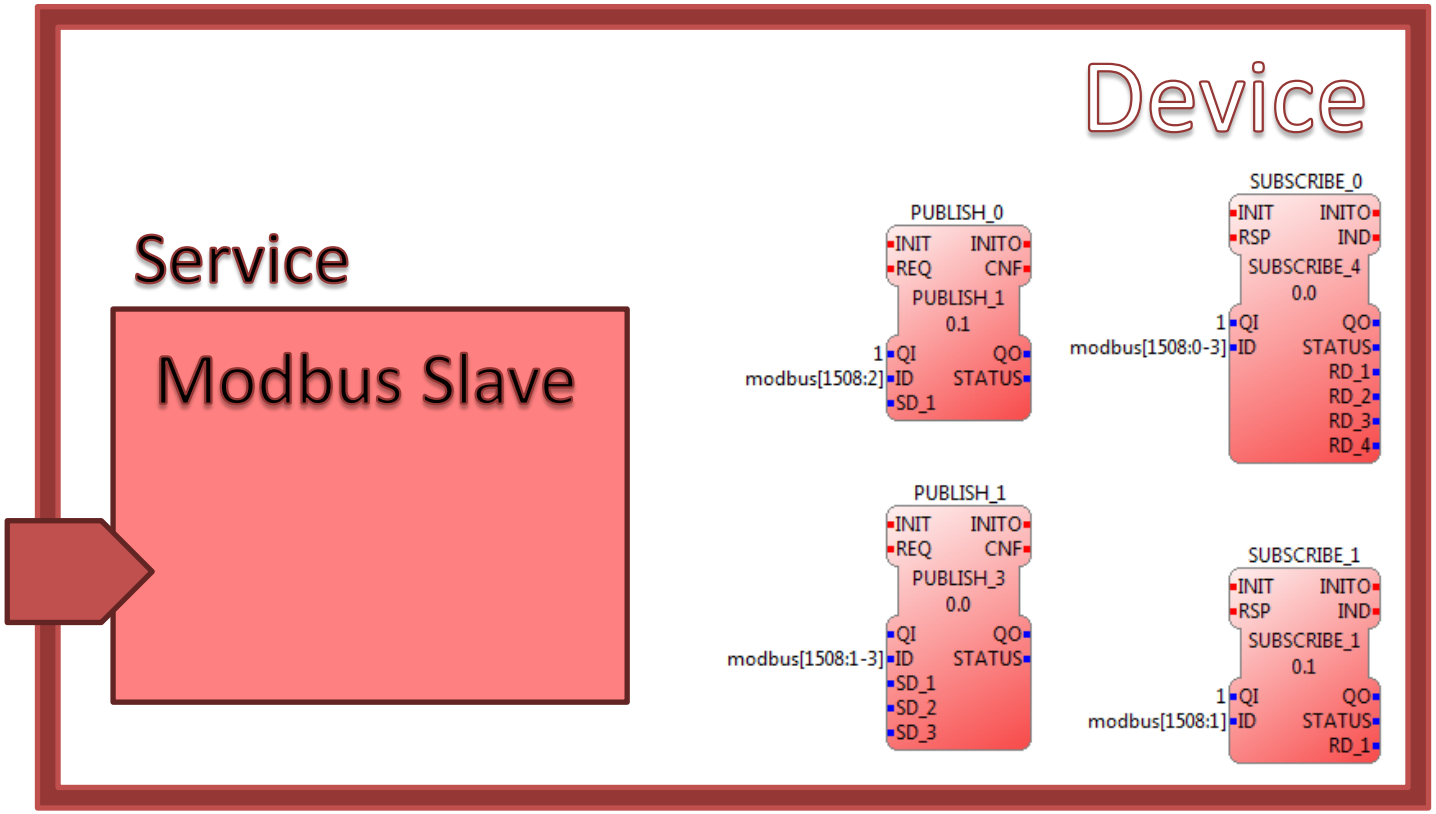


Modbus Protocol in 4DIAC



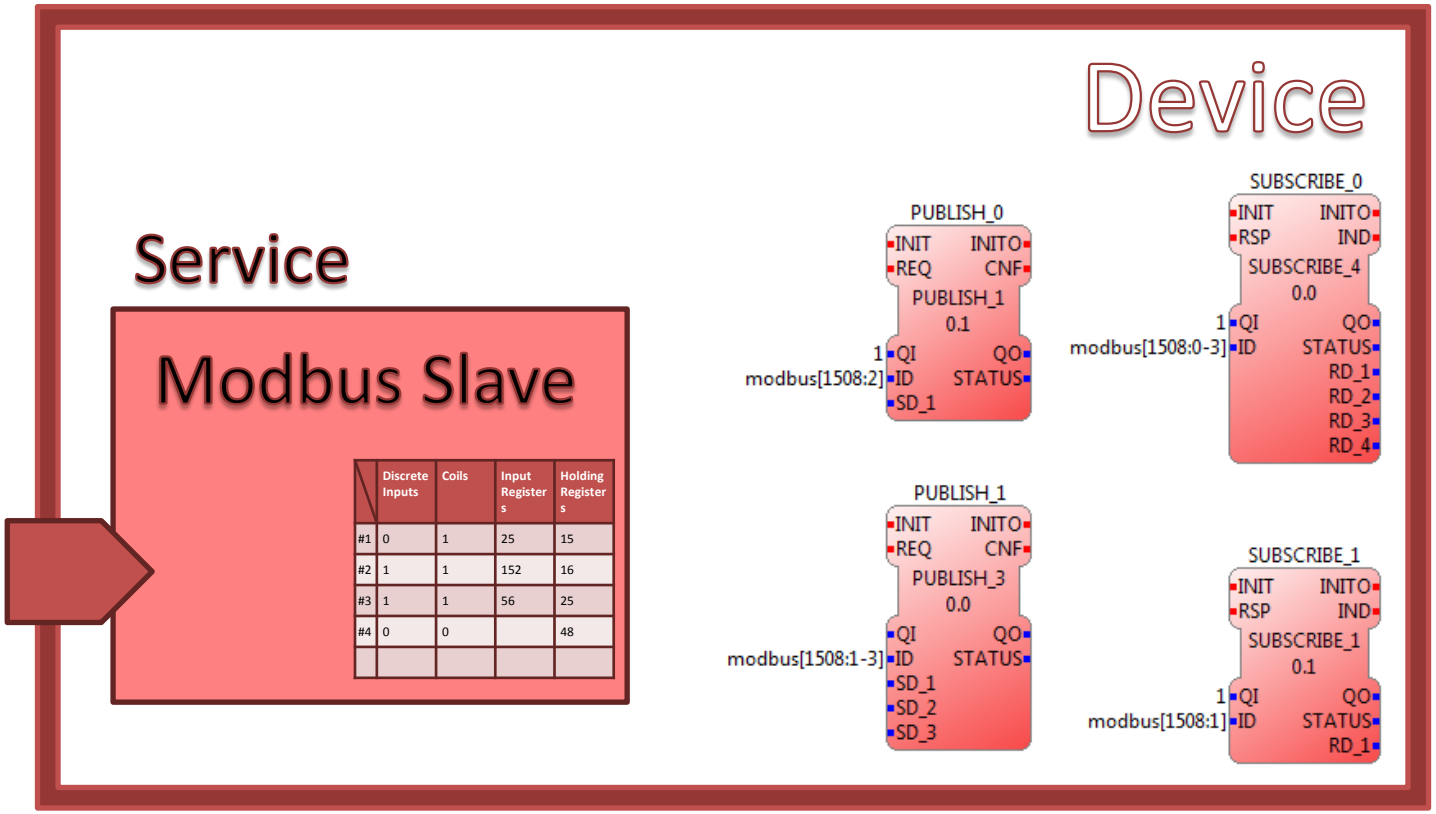
In such case the Modbus Slave is implemented as a Service inside the Device.

Modbus Protocol in 4DIAC



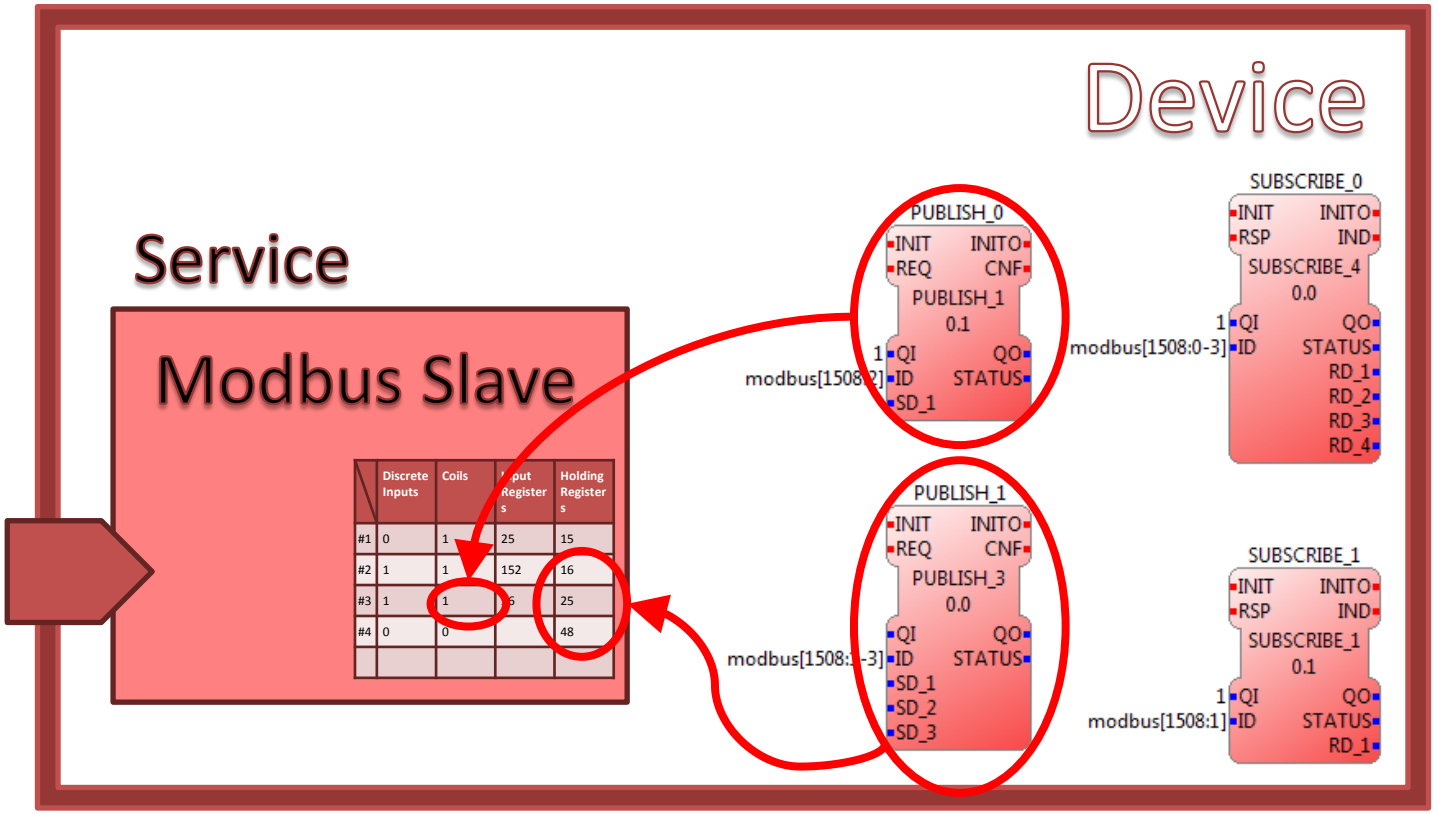
The Service will set a listening port for incoming Modbus requests …

Modbus Protocol in 4DIAC



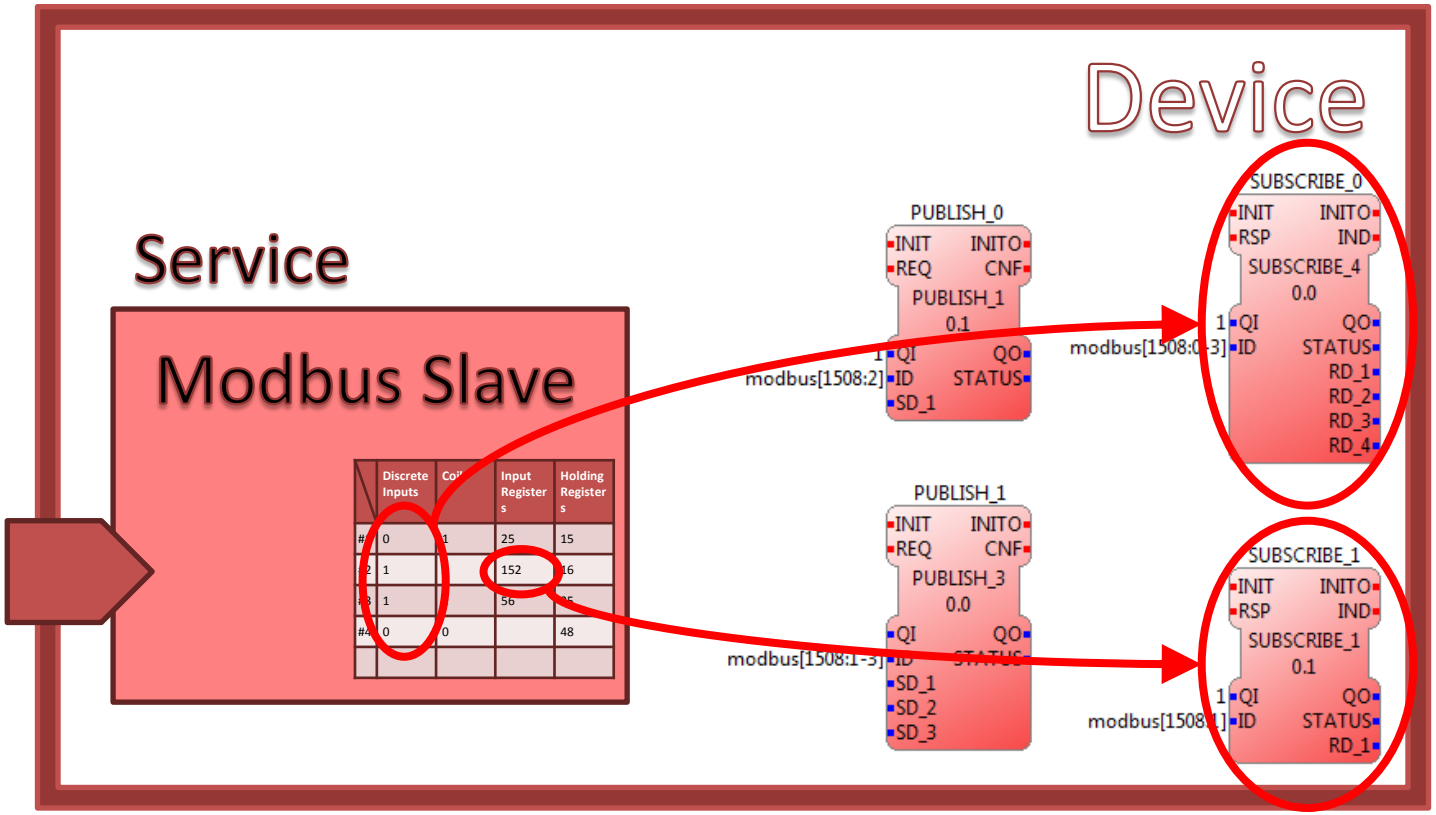
The Service will set a listening port for incoming Modbus requests and allocate some memory for the Modbus data table.

Modbus Protocol in 4DIAC



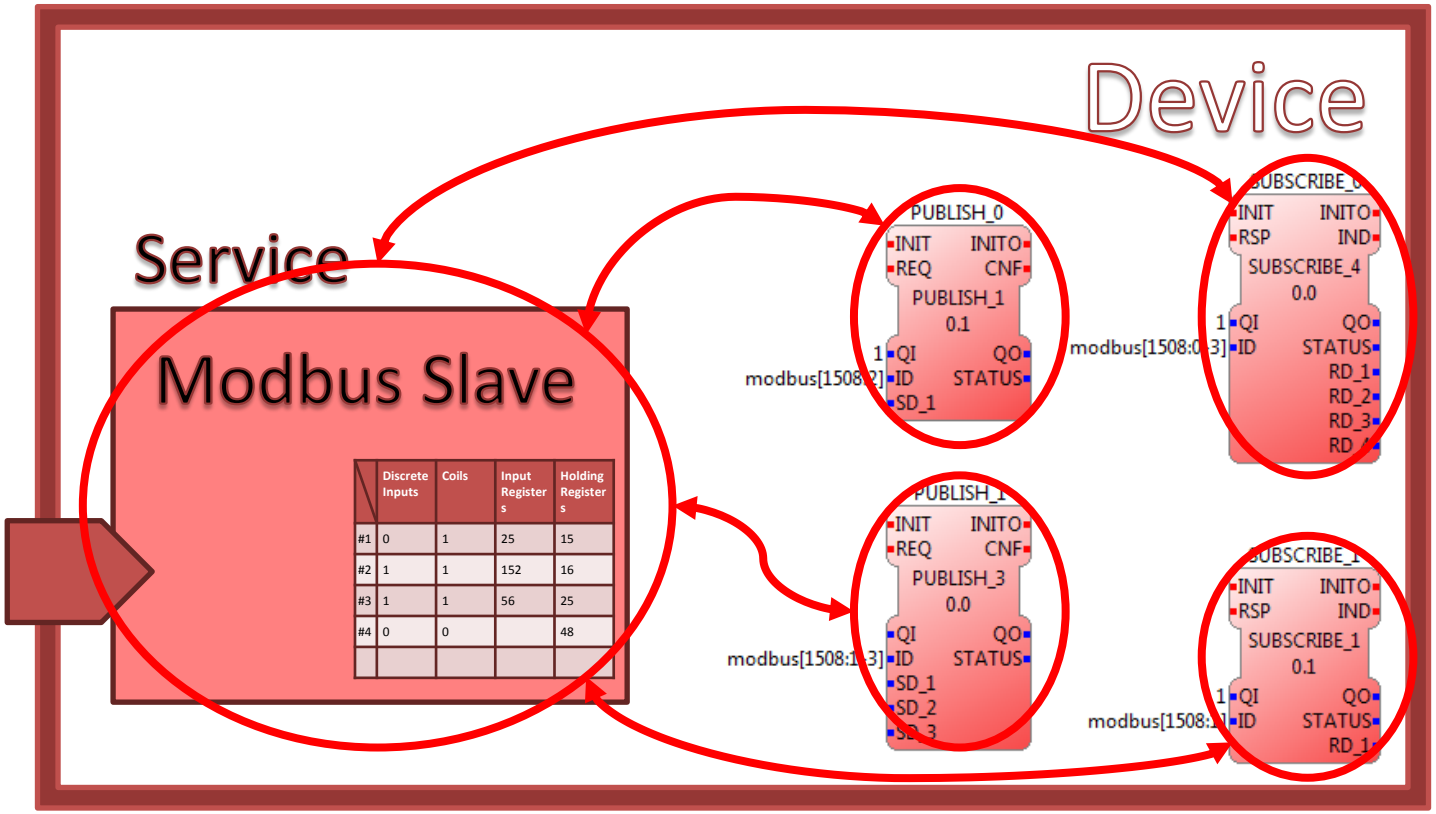
PUBLISH Function Blocks will “send” requests to the Modbus Slave Service to change some variables in the Modbus data table.

Modbus Protocol in 4DIAC



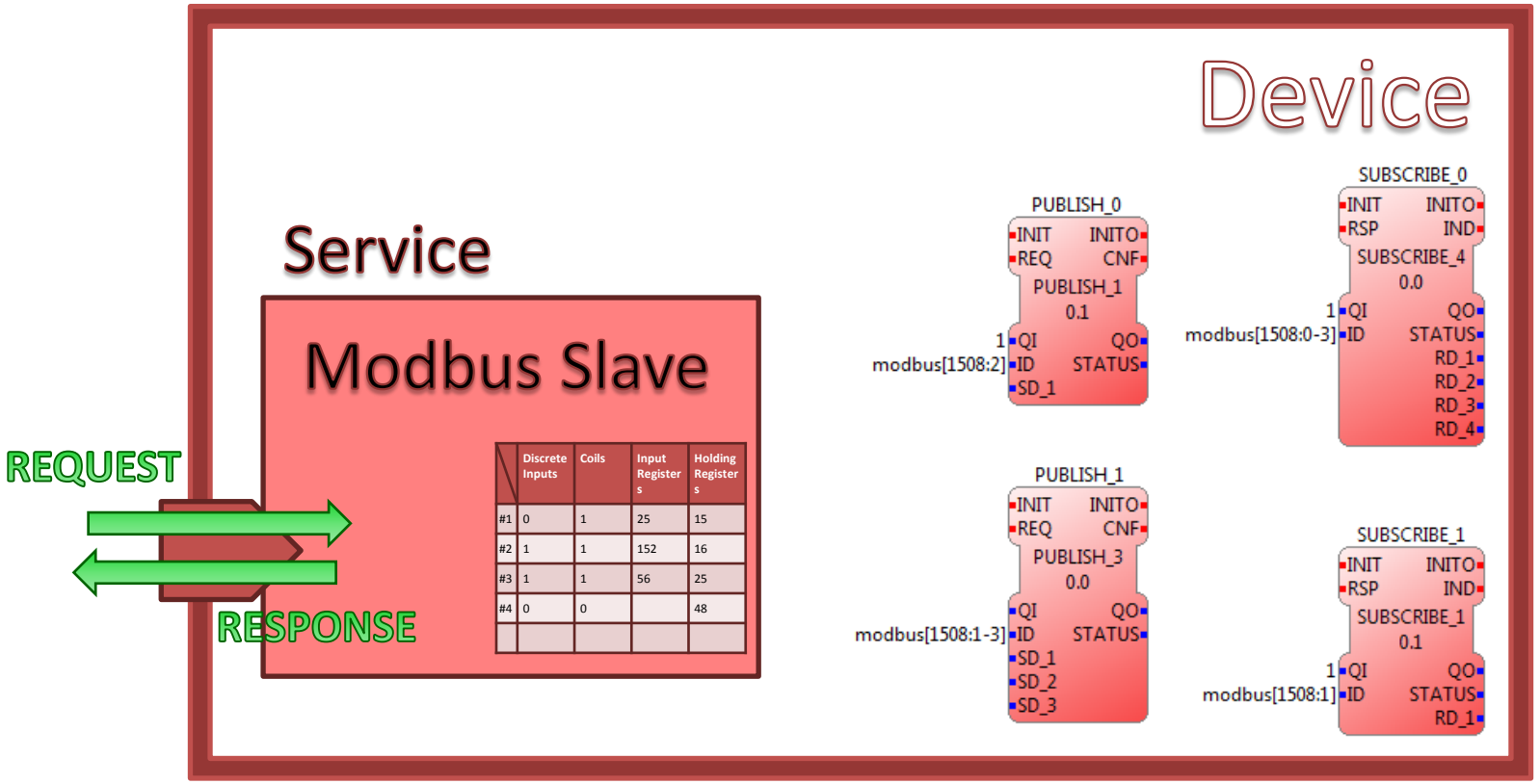
SUBSCRIBE Function Blocks will “listen” to changes in the Modbus data table and will be informed by the Modbus Slave Service about such changes.

Modbus Protocol in 4DIAC



Many PUBLISH and SUBSCRIBE Function Blocks can be associated to a single Modbus Slave Service and function simultaneously.

Modbus Protocol in 4DIAC



On the other hand the Modbus Slave Service will serve requests from the Modbus Master independently of the IEC 61499 Application.

Modbus Protocol in 4DIAC

The advantages of the PUBLISH/SUBSCRIBE scheme are:

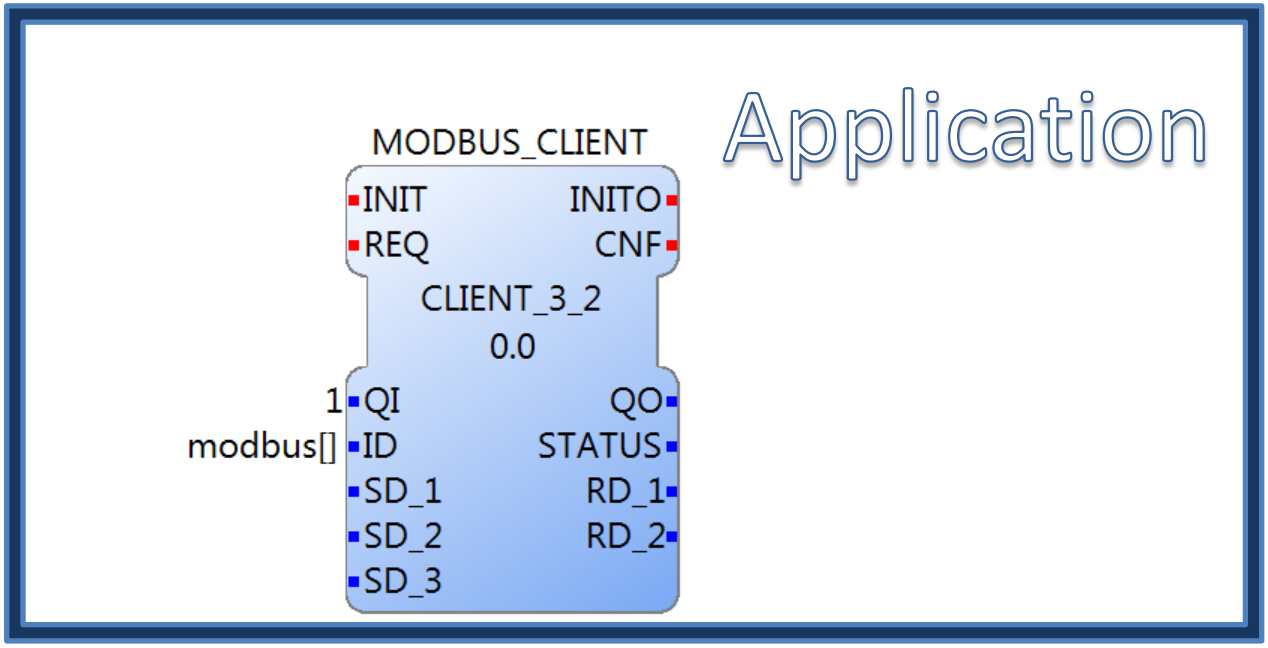
- A much simpler implementation within the IEC 61499 Application.
- Support of all Modbus data types.
- Devices capable of serving both read and write Modbus requests.
- Ability to have many CFBs to interact with the Modbus Slave Service from many points within a IEC 61499 Application.
- Separation of IEC 61499 Application’s and network’s access of the Modbus data table.
- Elimination of IEC 61499 Application’s interfere with the Modbus communication.

Modbus Protocol in 4DIAC

On the other hand an IEC 61499 Application implementing a Modbus Master must be aware of the Modbus Slaves' responses either because they contain data or to have knowledge of the write request's progress.

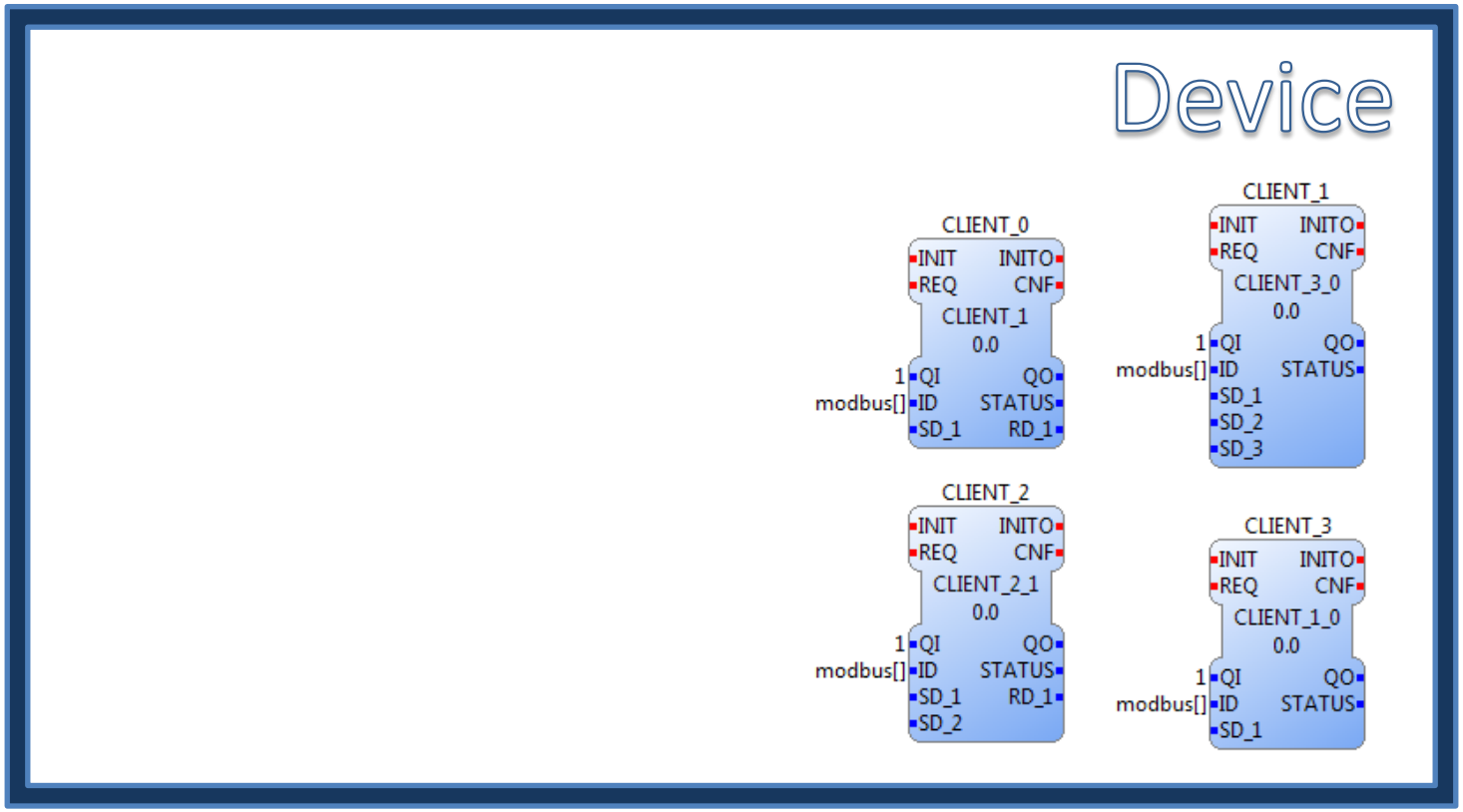
So CLIENT Communication Function Blocks would fit perfect for the Modbus Master implementation.

Modbus Protocol in 4DIAC



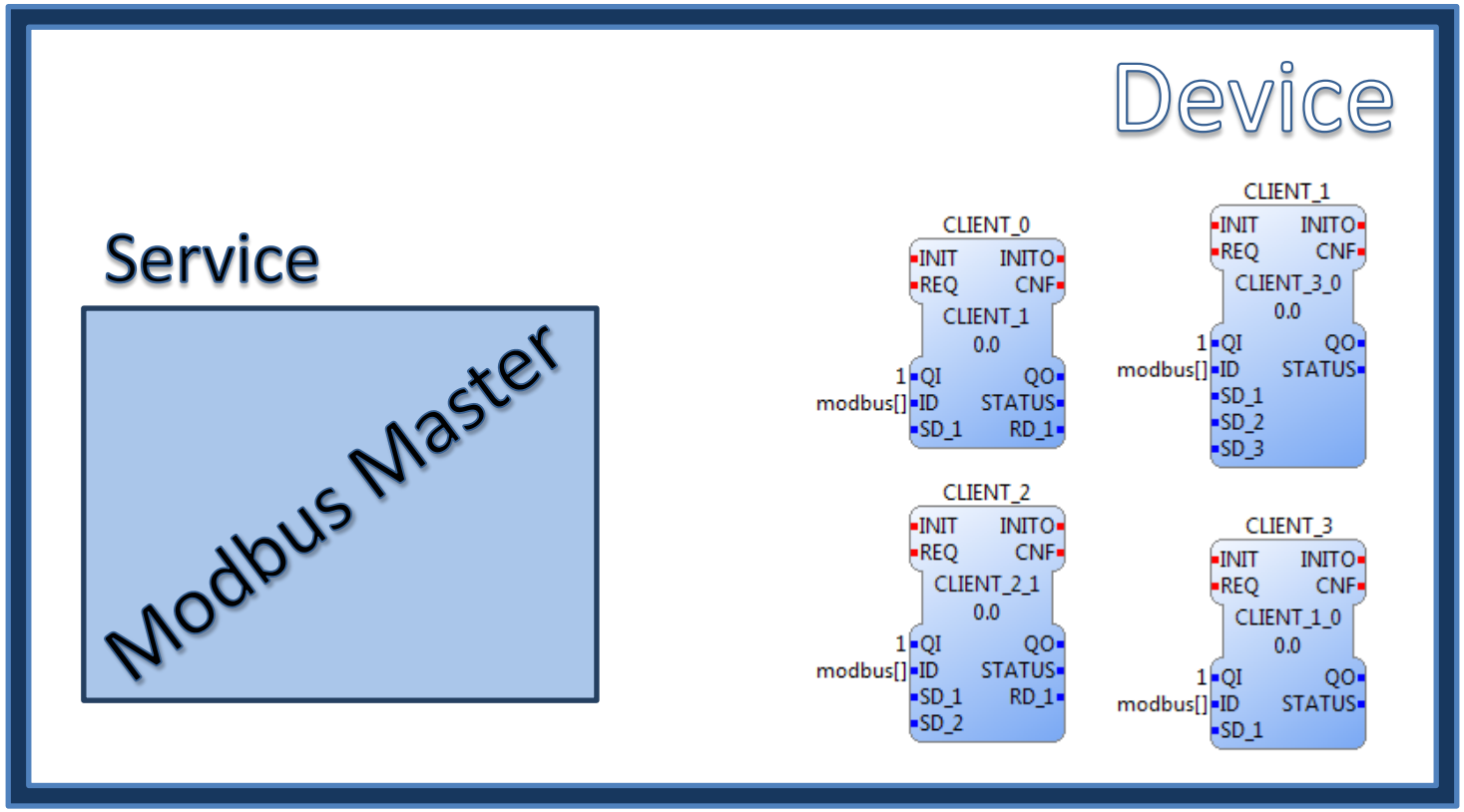
But implementing the Modbus Master with a single CLIENT FB comes to the same deadlock as the implementation of the Modbus Slave with a single SERVER FB.

Modbus Protocol in 4DIAC



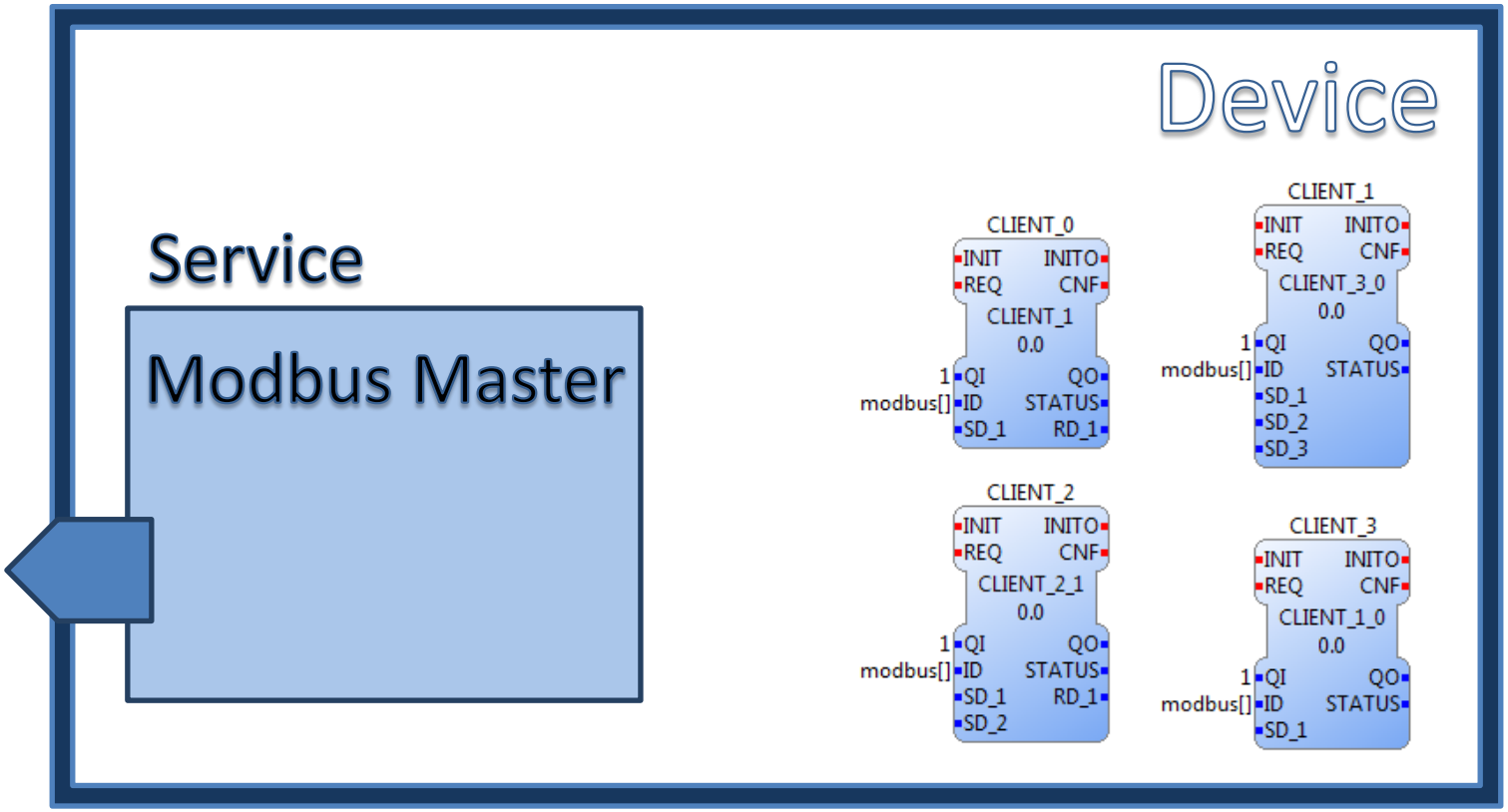
Instead multiple CLIENT FBs interacting with a single Modbus Master Service can be used.

Modbus Protocol in 4DIAC



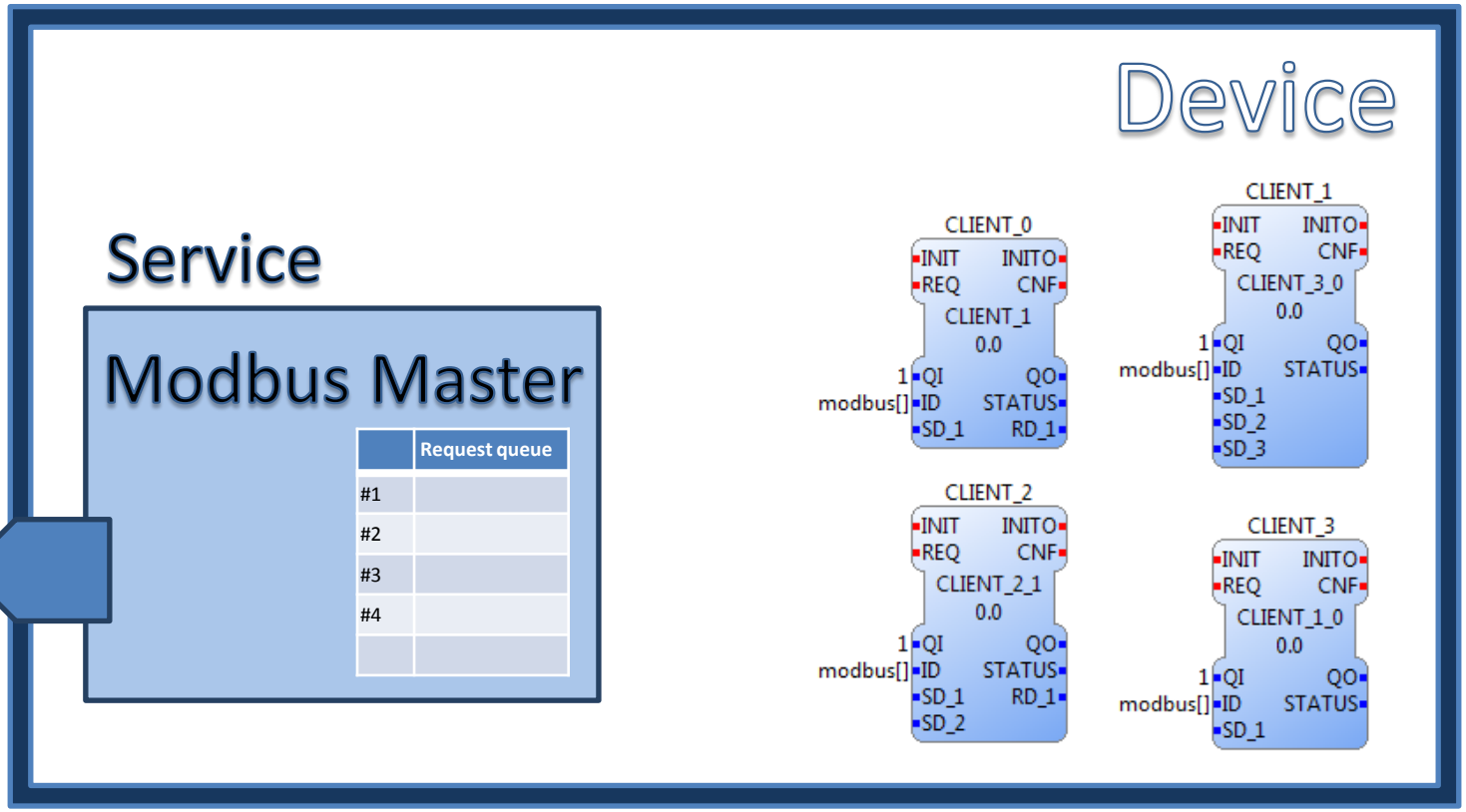
So the Modbus Master is implemented as a Service inside the Device.

Modbus Protocol in 4DIAC



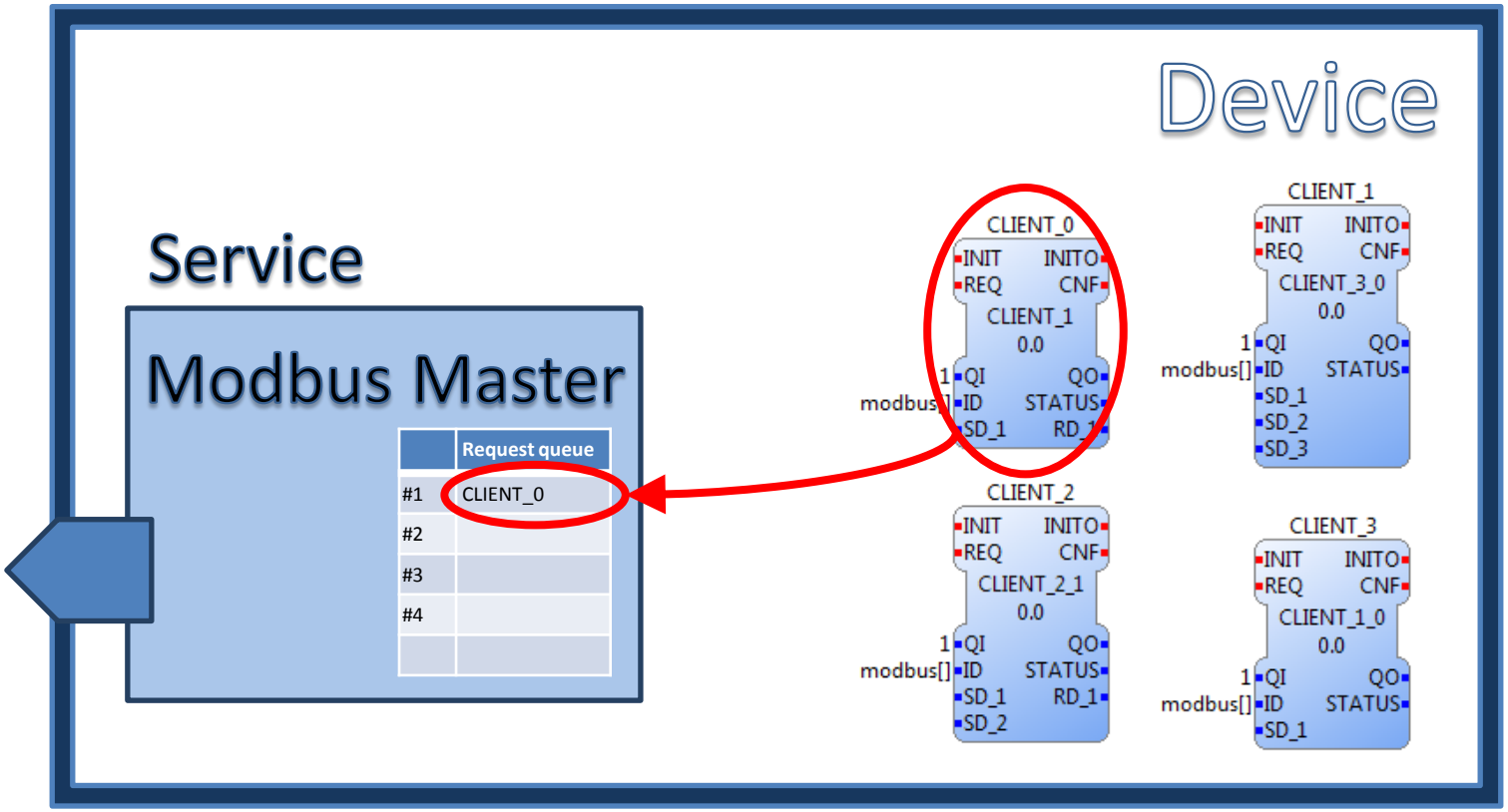
The Service will open a port for sending Modbus requests …

Modbus Protocol in 4DIAC



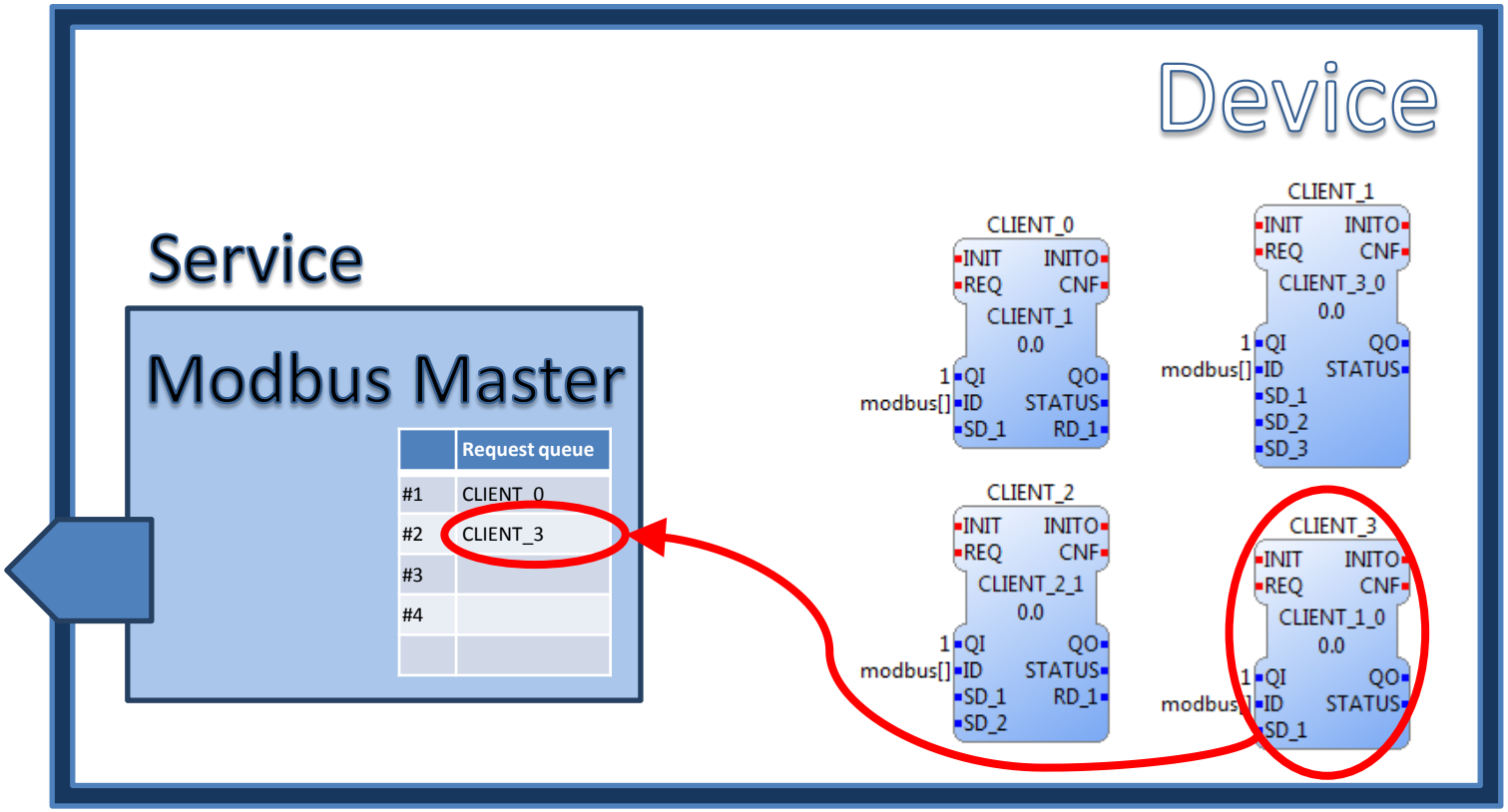
The Service will open a port for sending Modbus requests and create a queue to store requests registered by CLIENT Function Blocks.

Modbus Protocol in 4DIAC



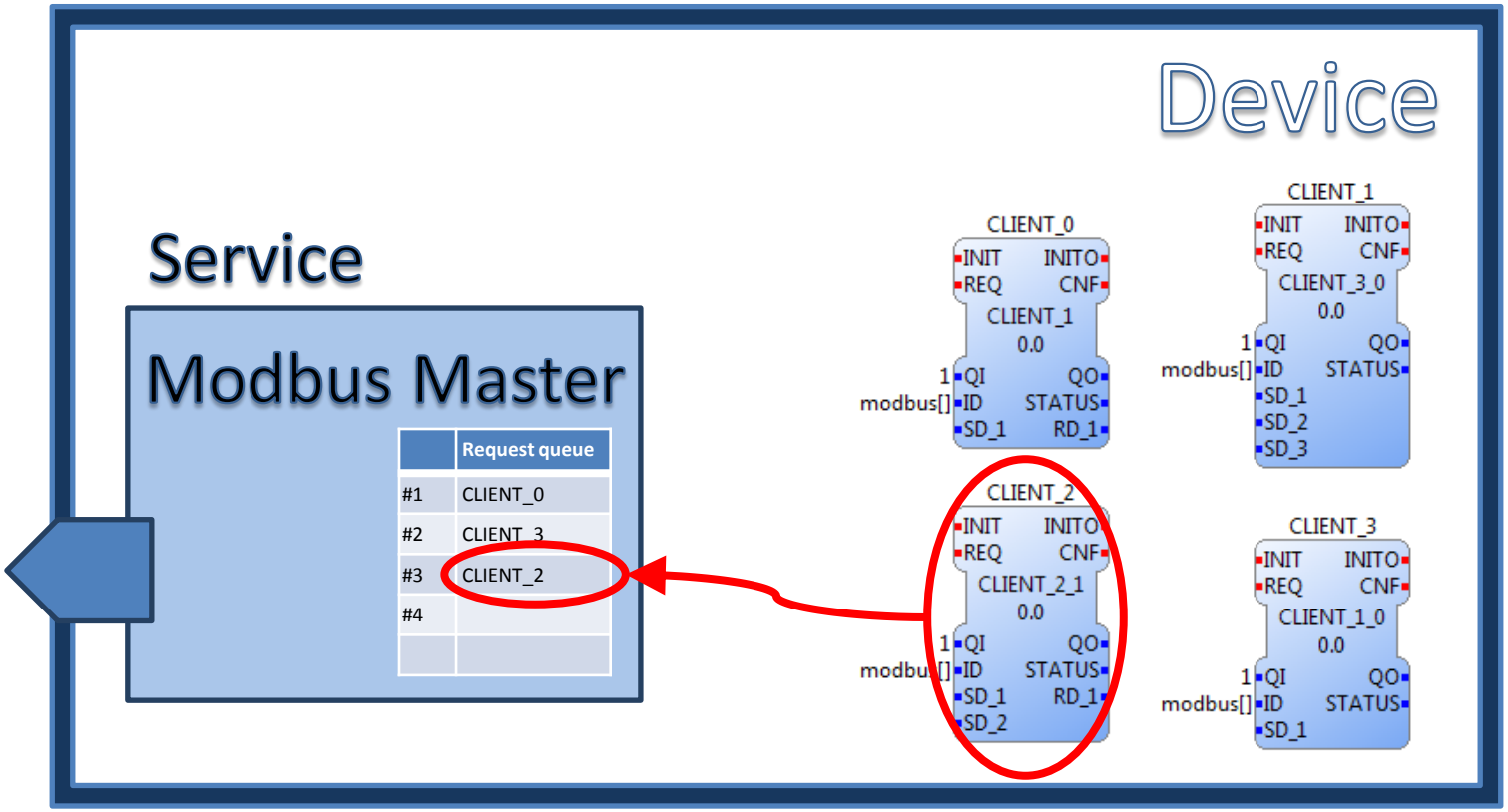
Each time a CLIENT Function Block receives a REQ event its request is appended to the Request queue of the Modbus Client Service.

Modbus Protocol in 4DIAC



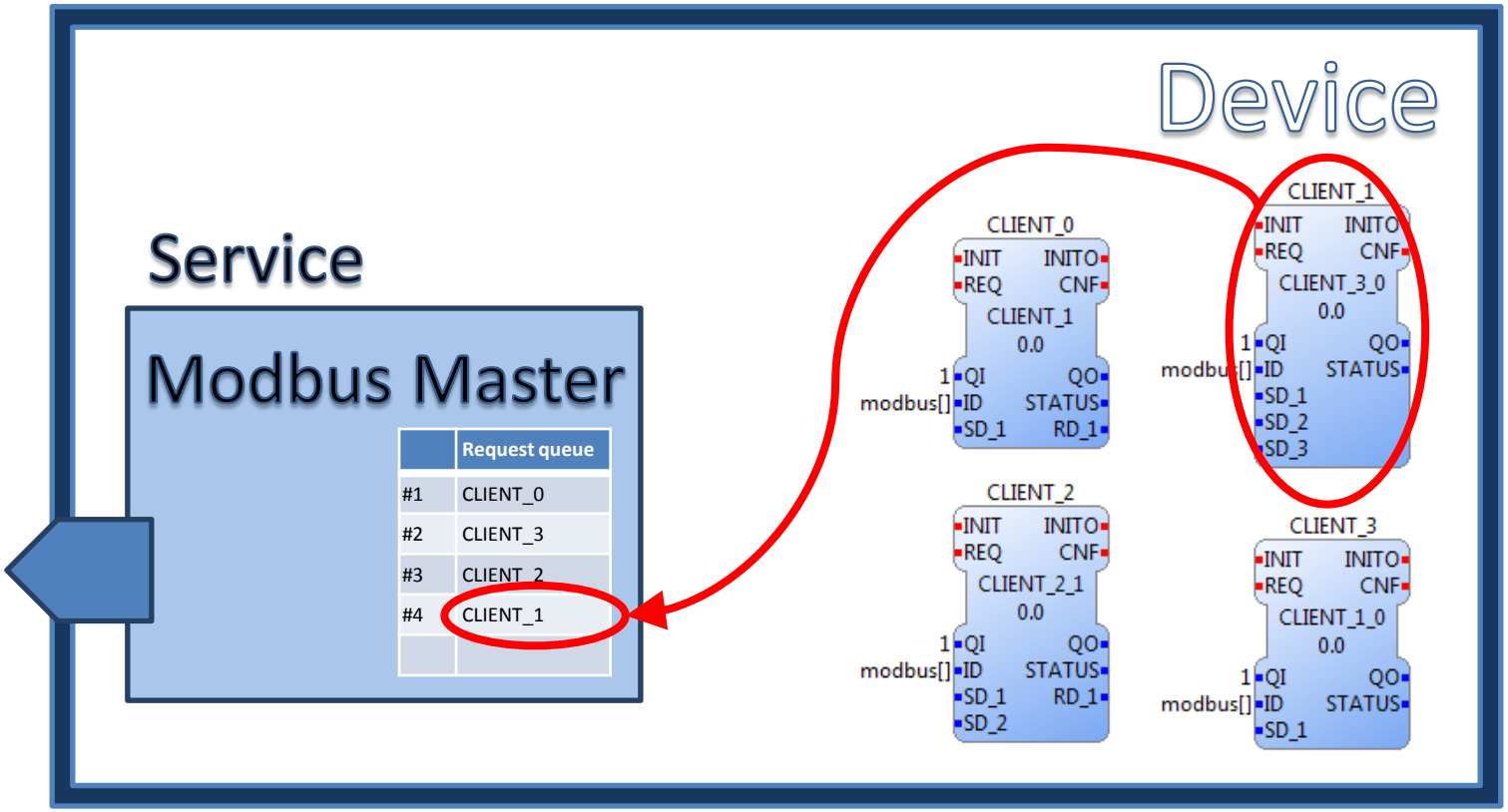
Each time a CLIENT Function Block receives a REQ event its request is appended to the Request queue of the Modbus Client Service.

Modbus Protocol in 4DIAC



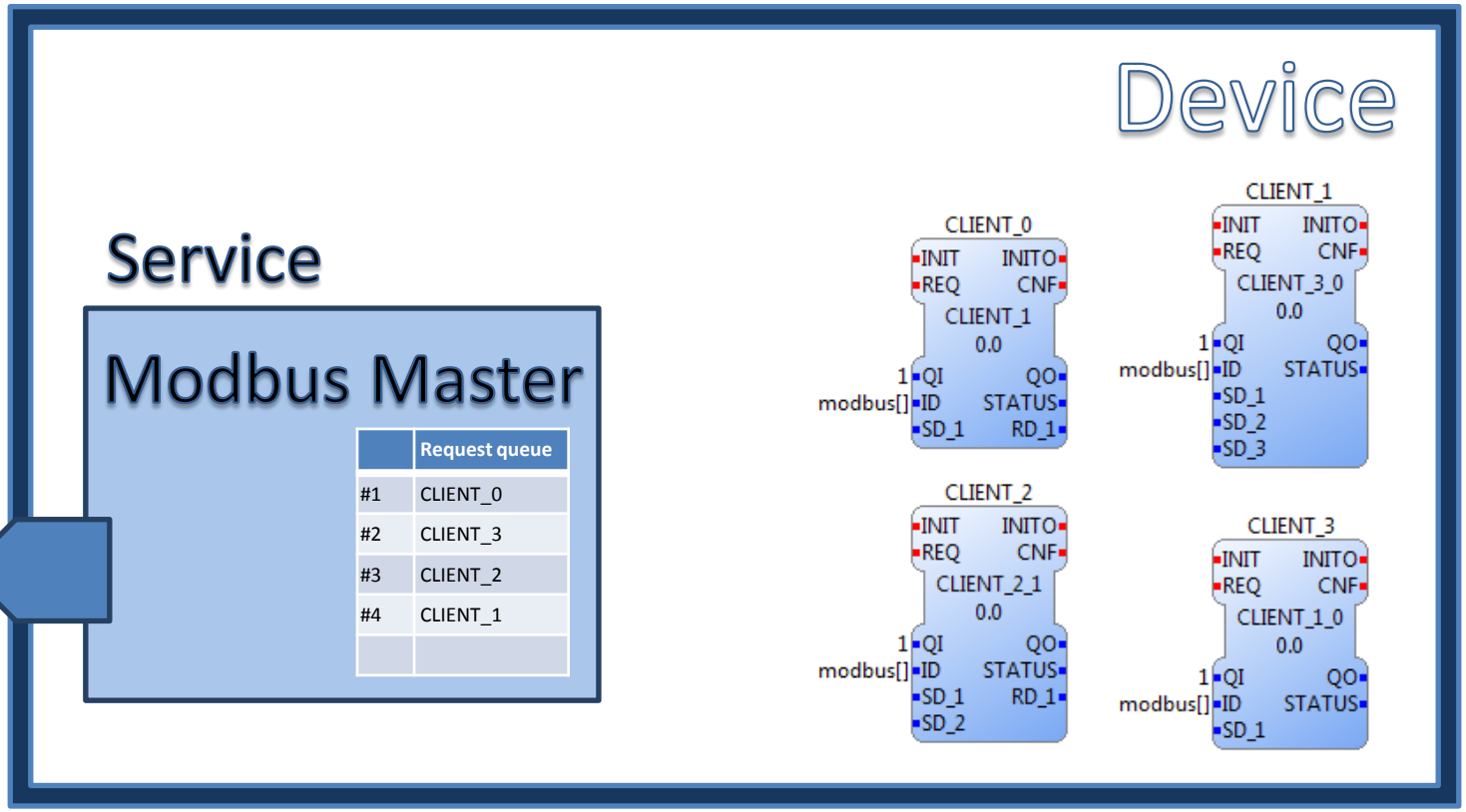
Each time a CLIENT Function Block receives a REQ event its request is appended to the Request queue of the Modbus Client Service.

Modbus Protocol in 4DIAC



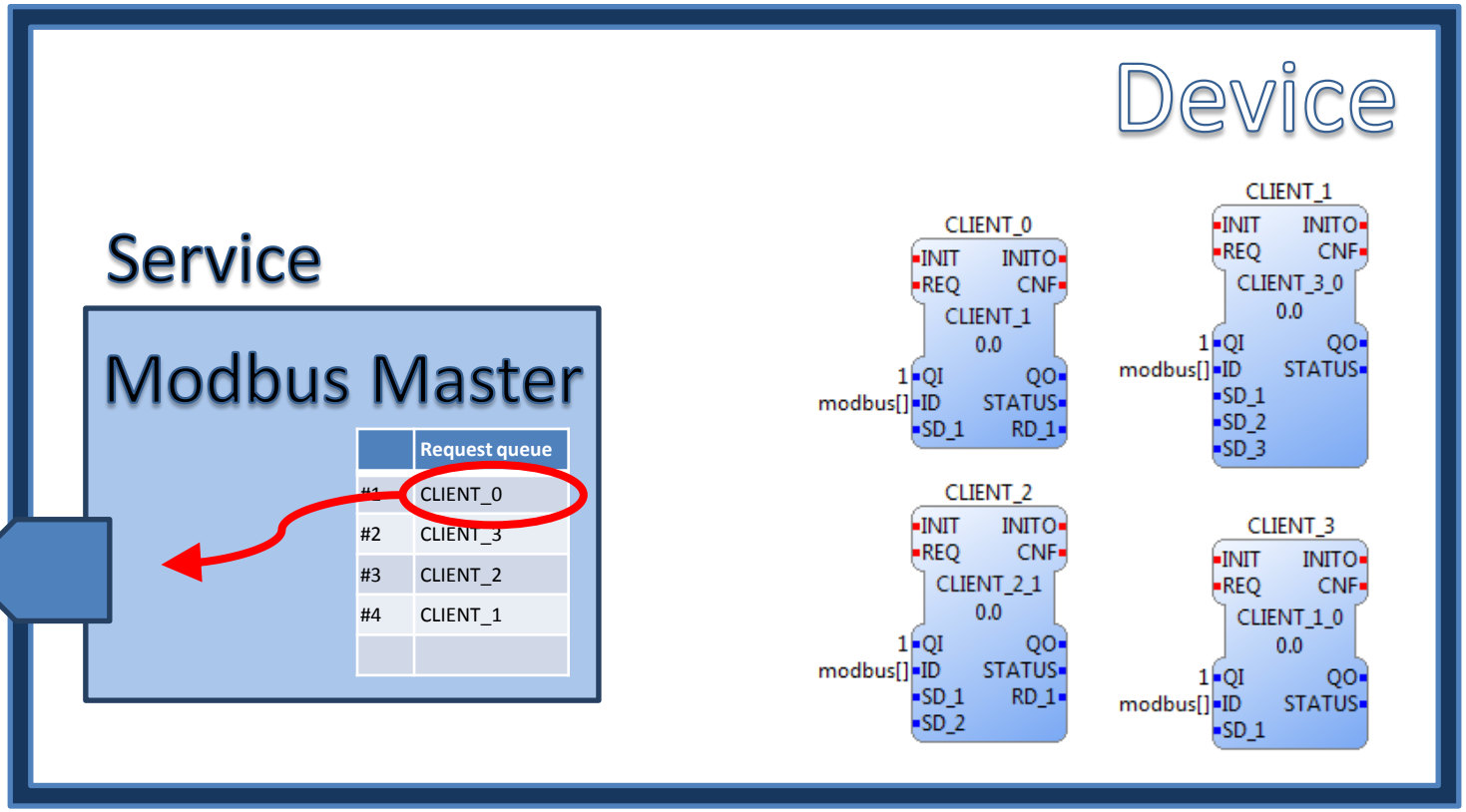
Each time a CLIENT Function Block receives a REQ event its request is appended to the Request queue of the Modbus Client Service.

Modbus Protocol in 4DIAC



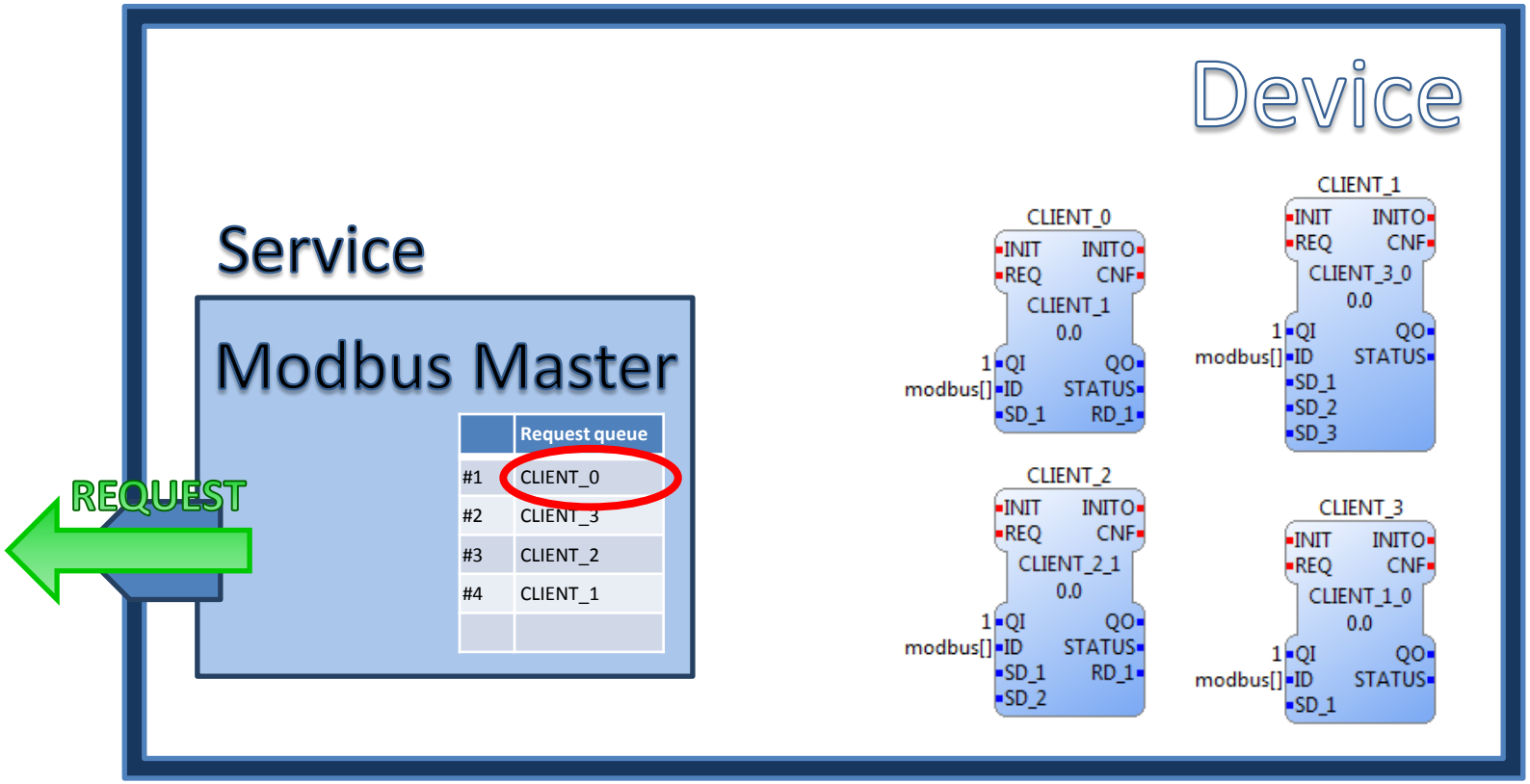
The Modbus Master can send only one request at a time so it picks the top of the Request queue and serves it.

Modbus Protocol in 4DIAC



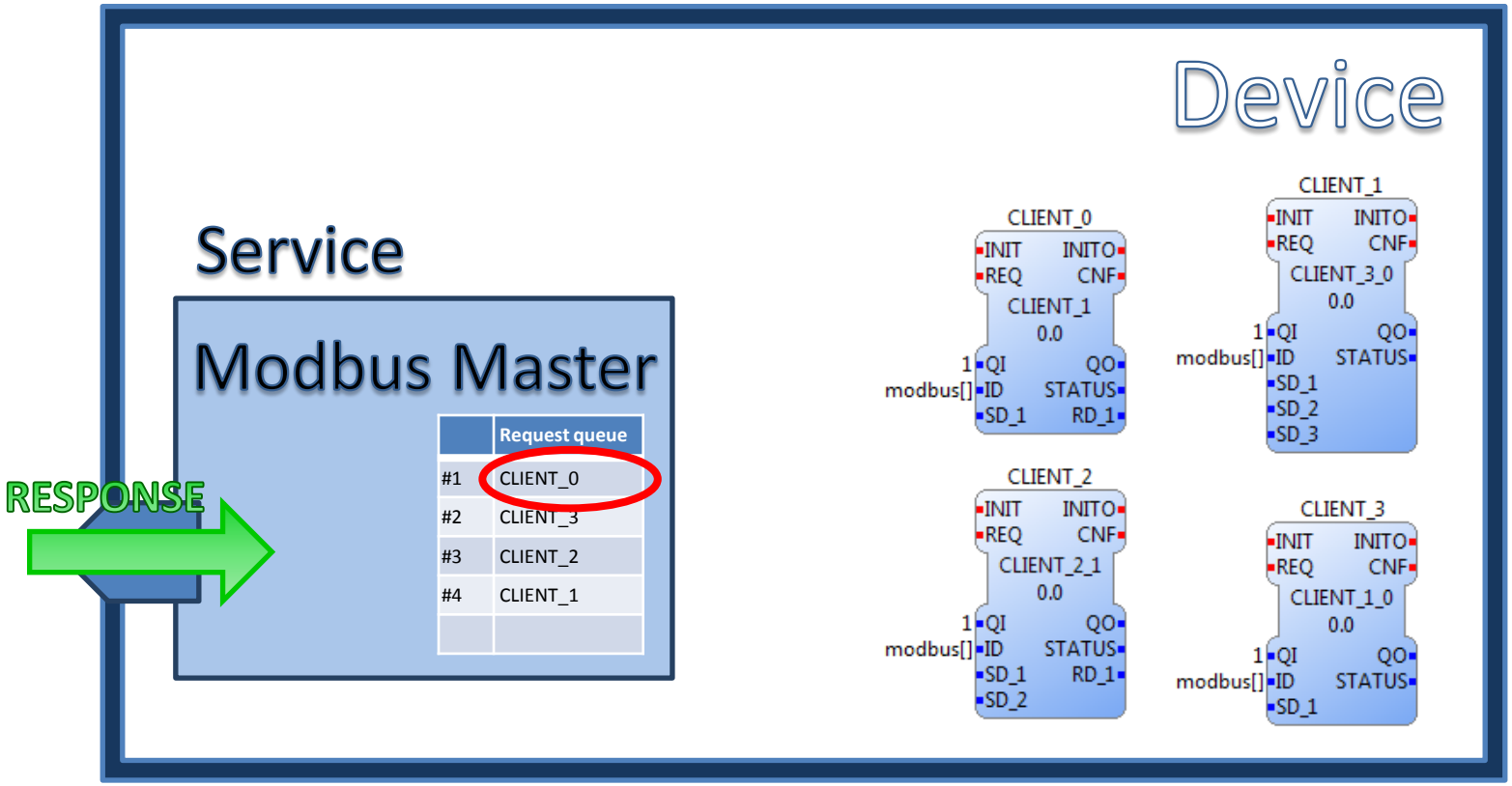
The top of the Request queue is picked.

Modbus Protocol in 4DIAC



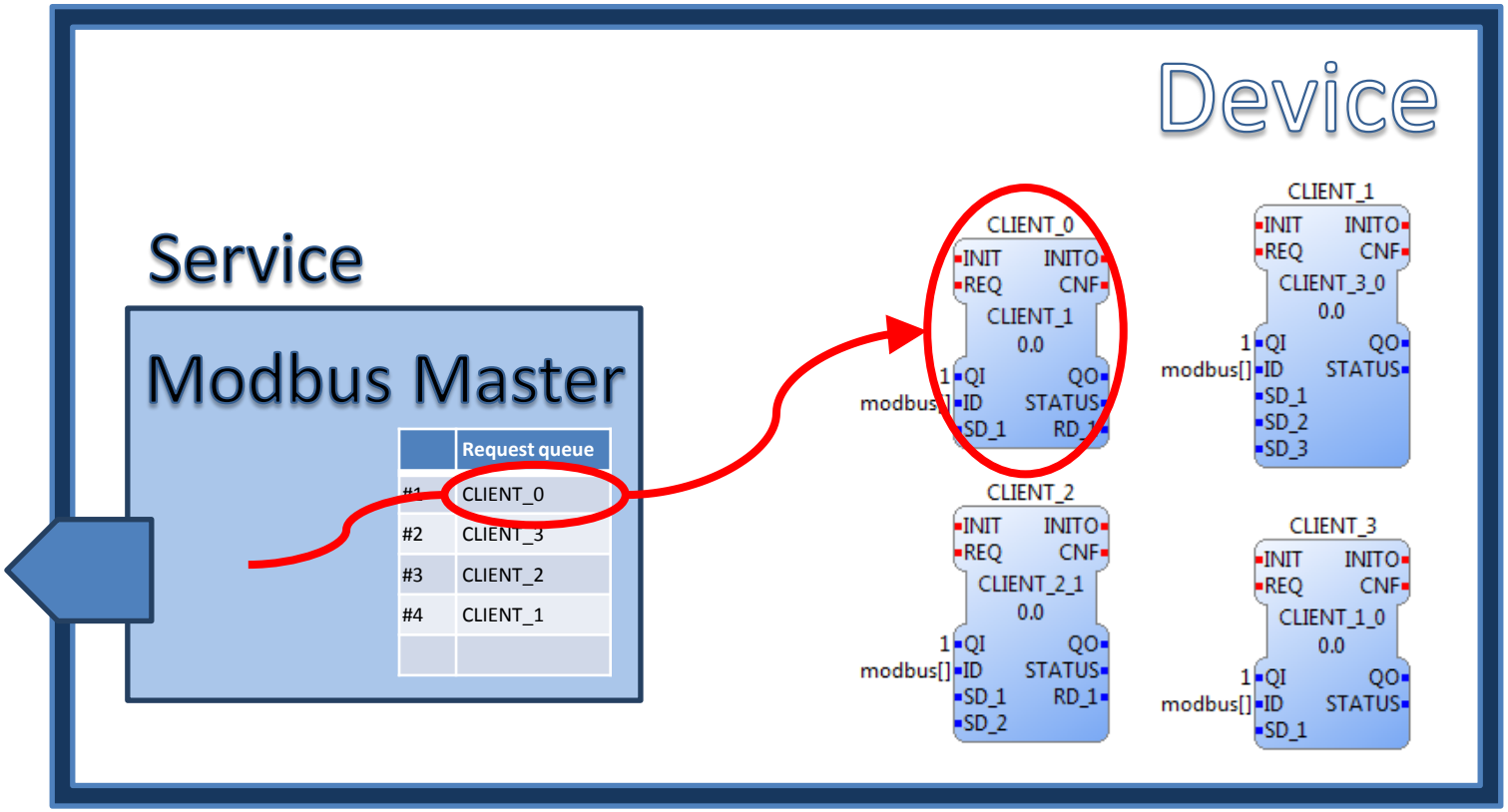
The top of the Request queue is picked. The request is sent.

Modbus Protocol in 4DIAC



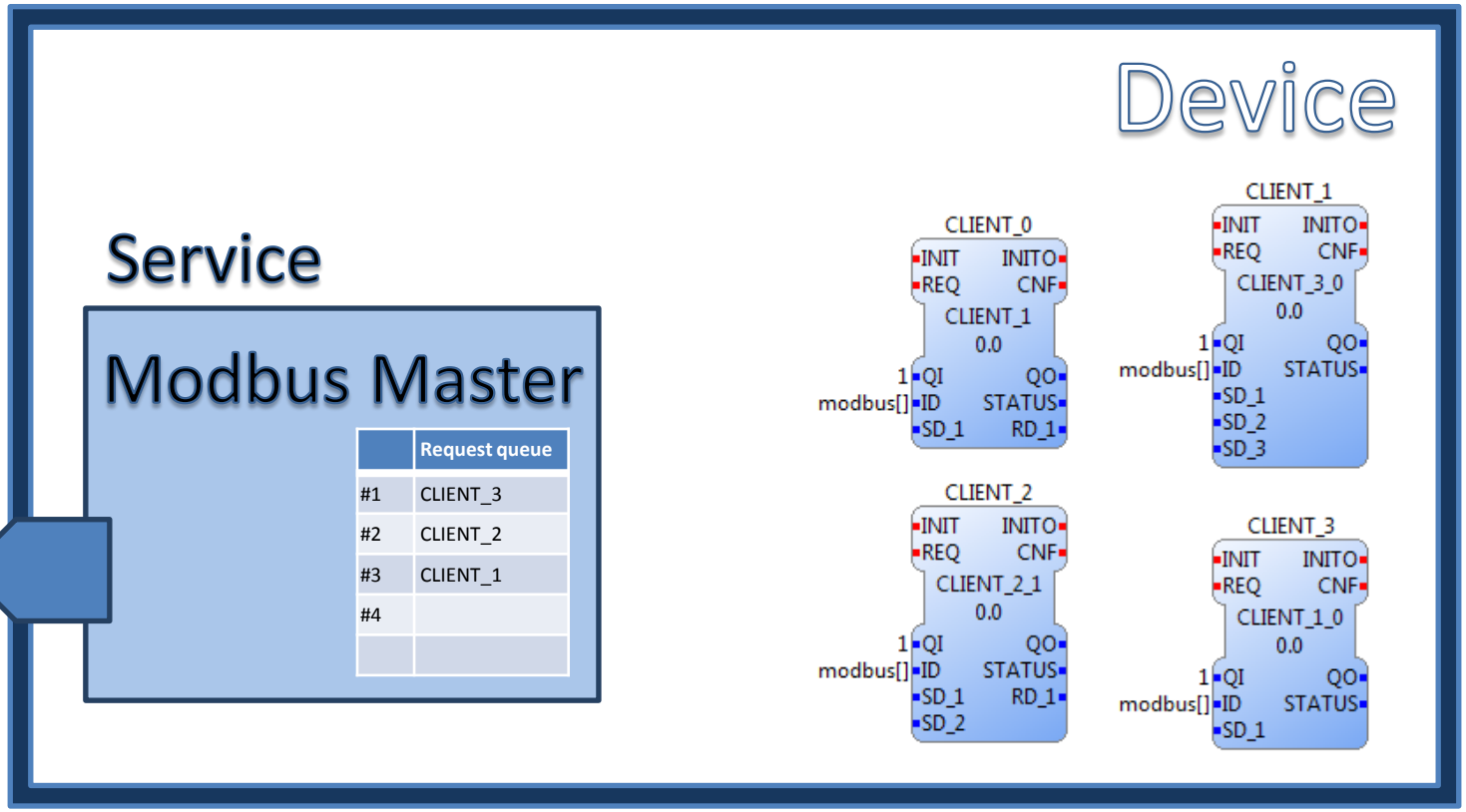
The top of the Request queue is picked. The request is sent. A response is received.

Modbus Protocol in 4DIAC



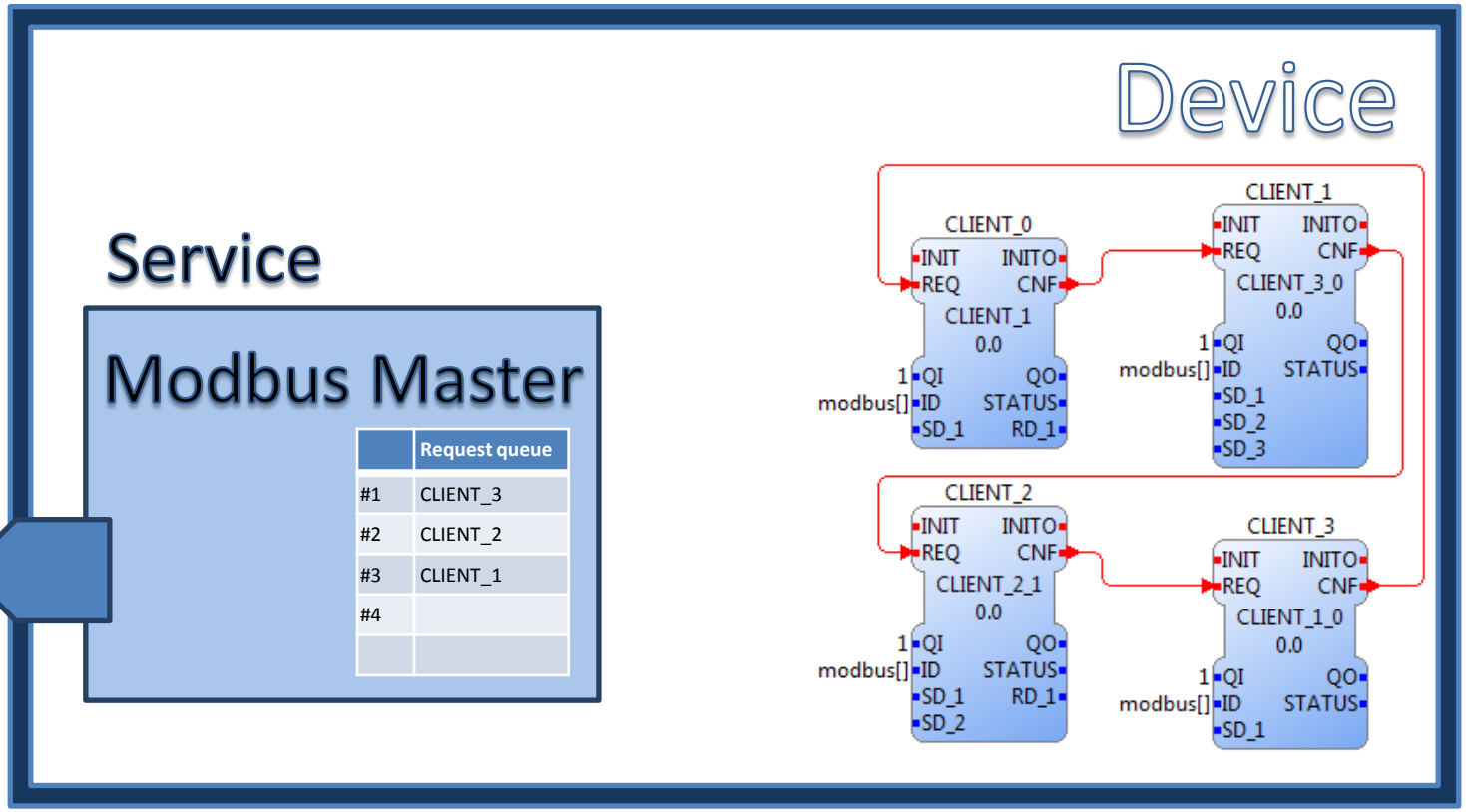
The top of the Request queue is picked. The request is sent.
A response is received. The corresponding CLIENT FB is informed.

Modbus Protocol in 4DIAC



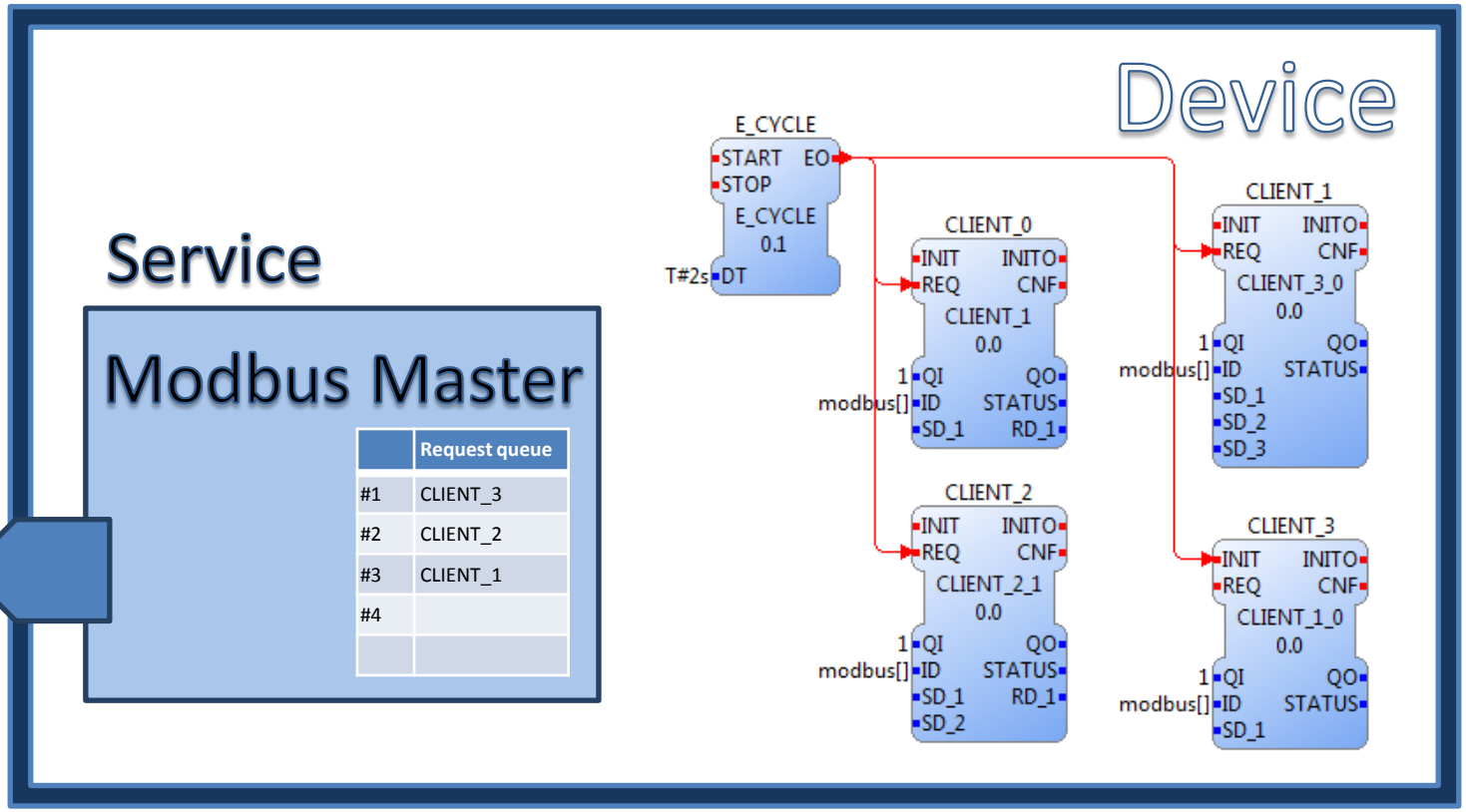
The top of the Request queue is picked. The request is sent.
A response is received. The corresponding CLIENT FB is informed.
The request is removed from the queue.

Modbus Protocol in 4DIAC



The IEC 61499 Application must take care to avoid long delays or the endless increase of the queue by using either a freewheel cascading scheme …

Modbus Protocol in 4DIAC



The IEC 61499 Application must take care to avoid long delays or the endless increase of the queue by using either a freewheel cascading scheme or the most common cyclic polling scheme.

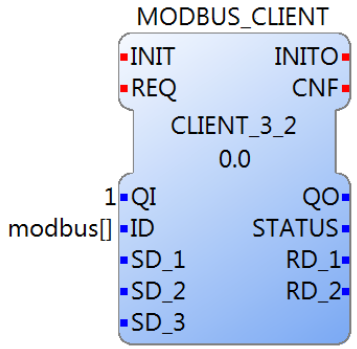
Modbus Protocol in 4DIAC

The advantages of this multiple CLIENT scheme are:

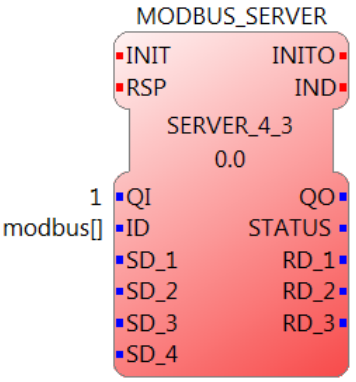
- A more flexible implementation within the IEC 61499 Application.
- Support of all Modbus data types.
- Devices capable of sending both read and write Modbus requests.
- Ability to have many CFBs to send Modbus requests from many places within a IEC 61499 Application.
- Time management of the Modbus Master from within the IEC 61499 Application using a freewheel, cyclic polling or other schedule scheme.

Modbus Protocol in 4DIAC

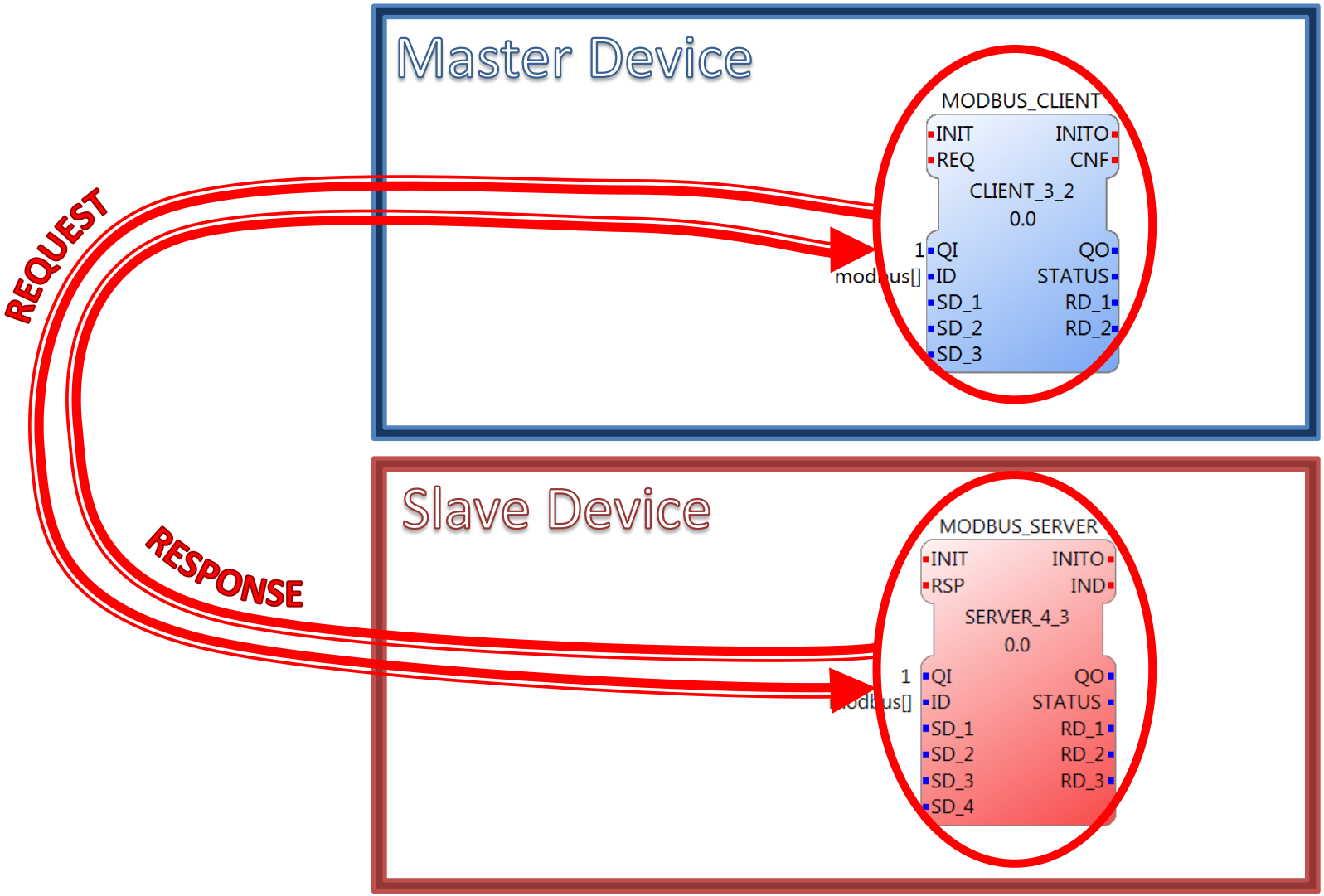
Master Device



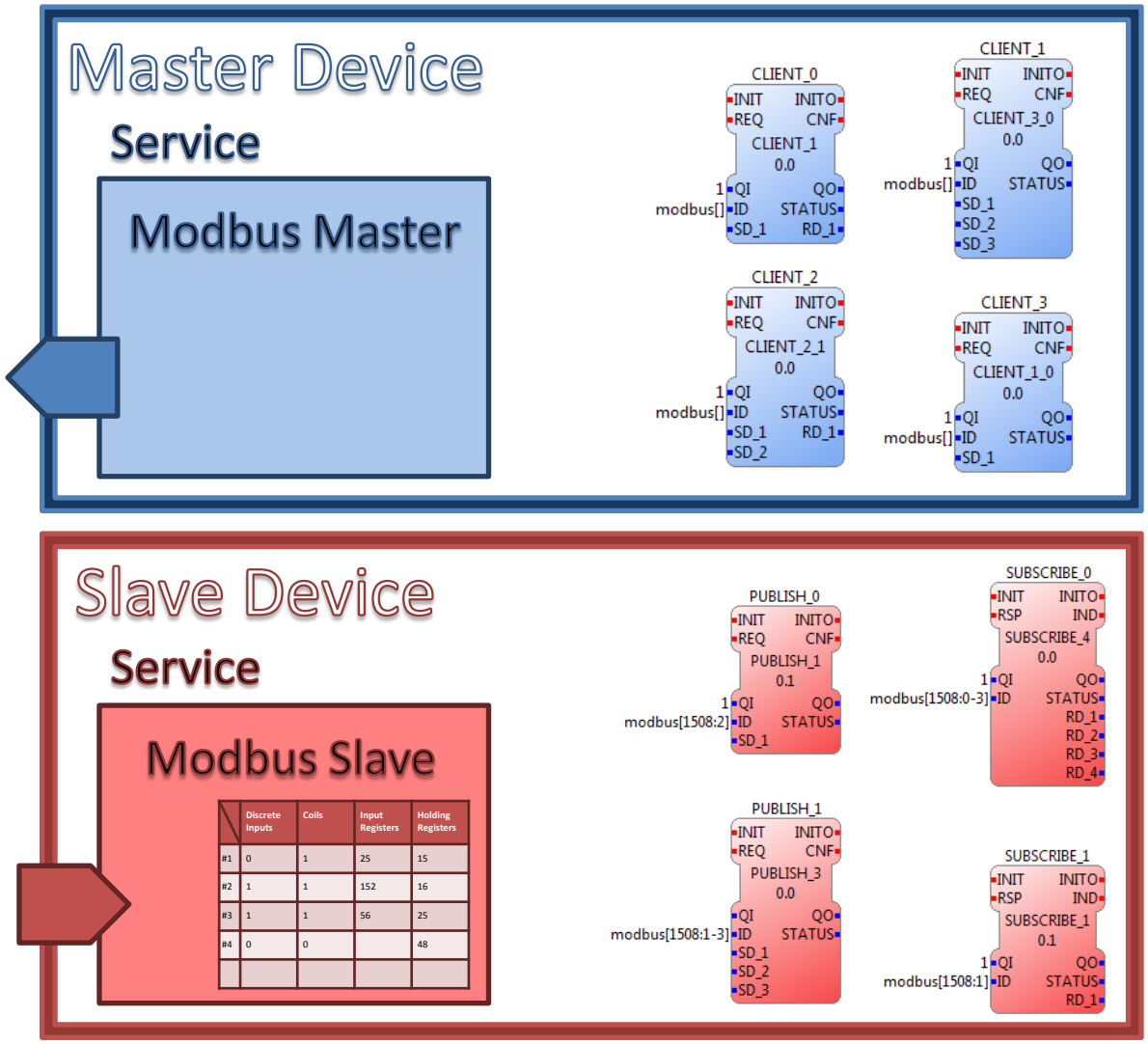
Slave Device



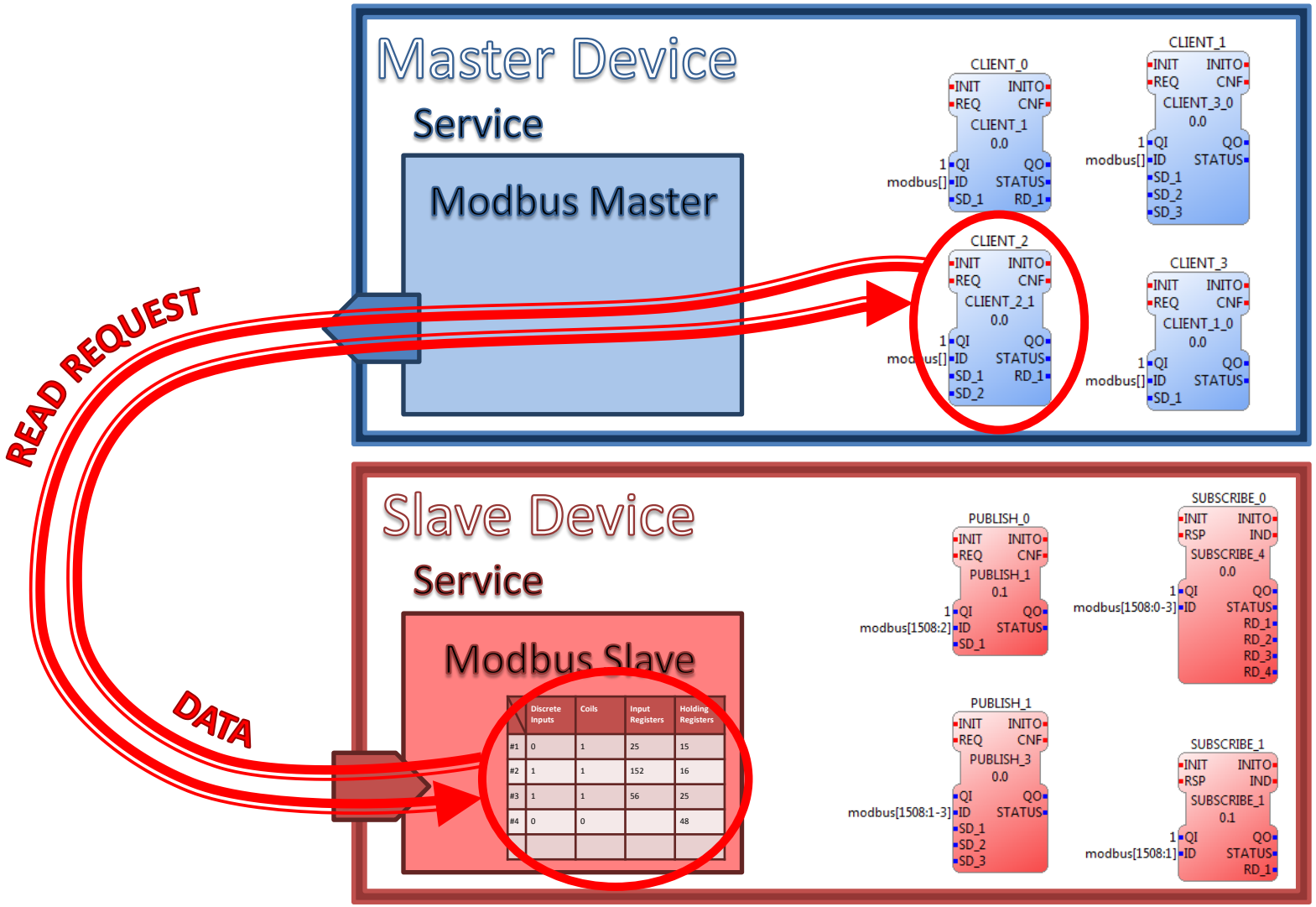
Modbus Protocol in 4DIAC



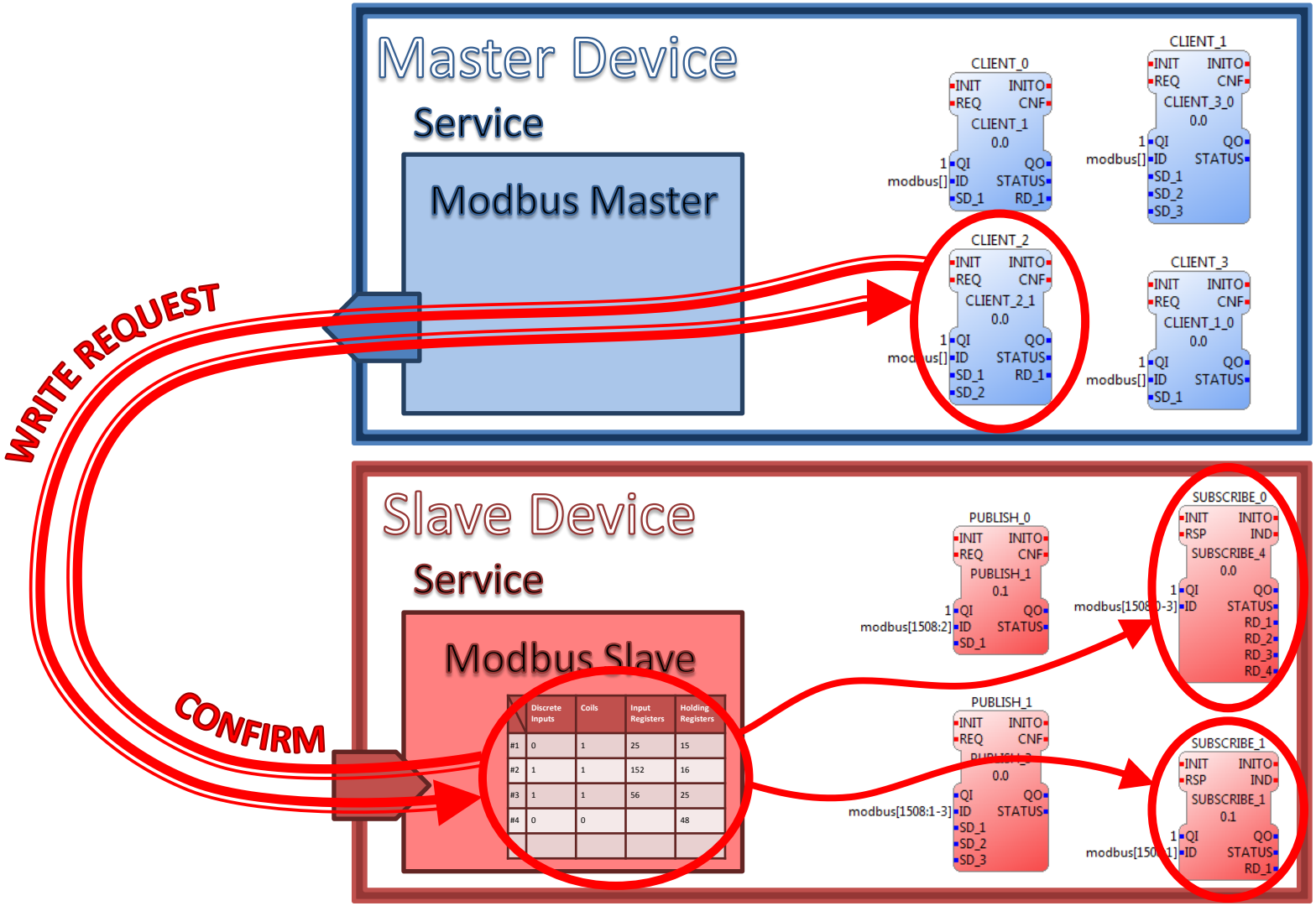
Modbus Protocol in 4DIAC



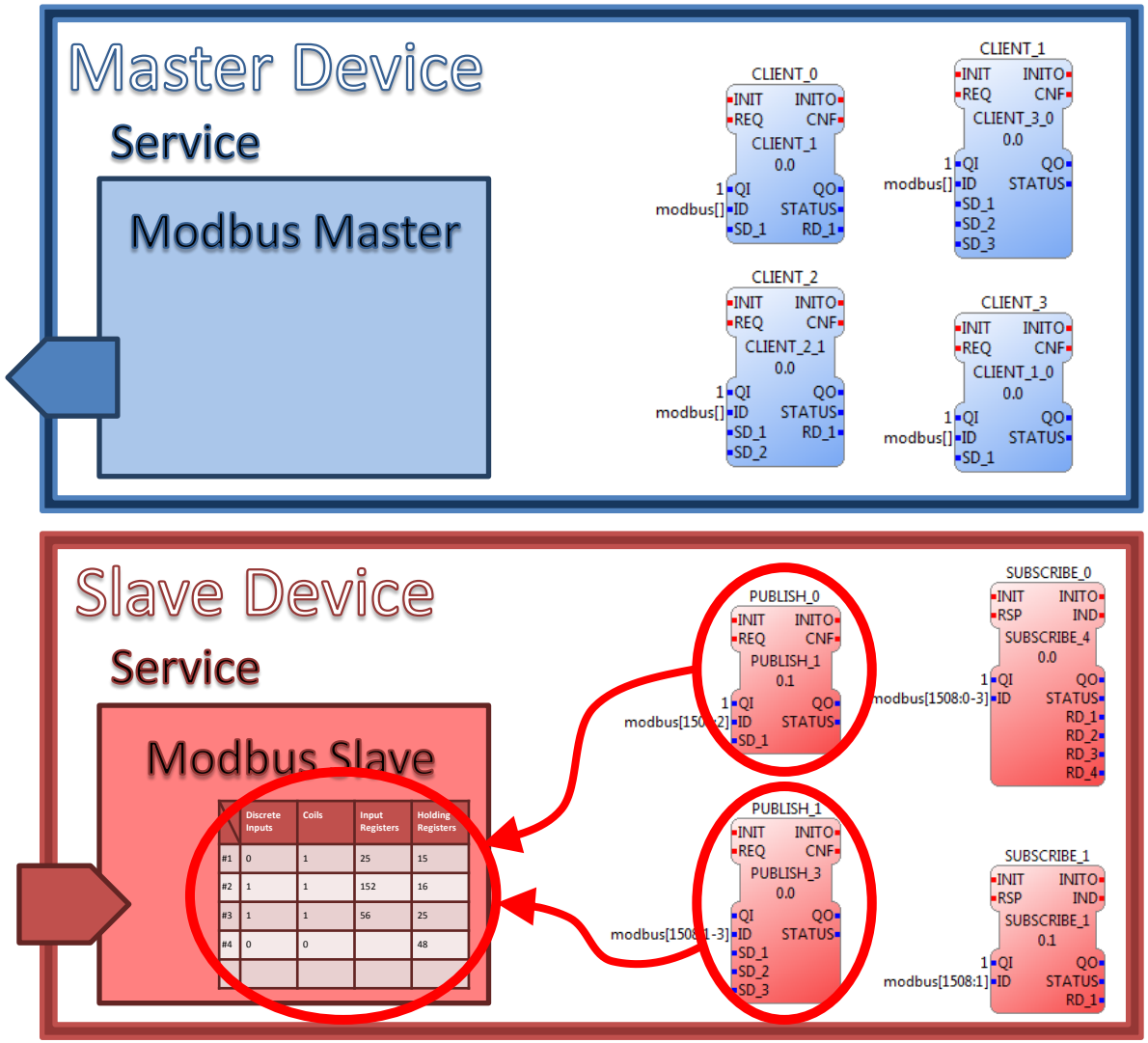
Modbus Protocol in 4DIAC



Modbus Protocol in 4DIAC



Modbus Protocol in 4DIAC



Modbus Protocol in 4DIAC

The application of the Modbus protocol with IEC 61499 Function Blocks in the proposed way:

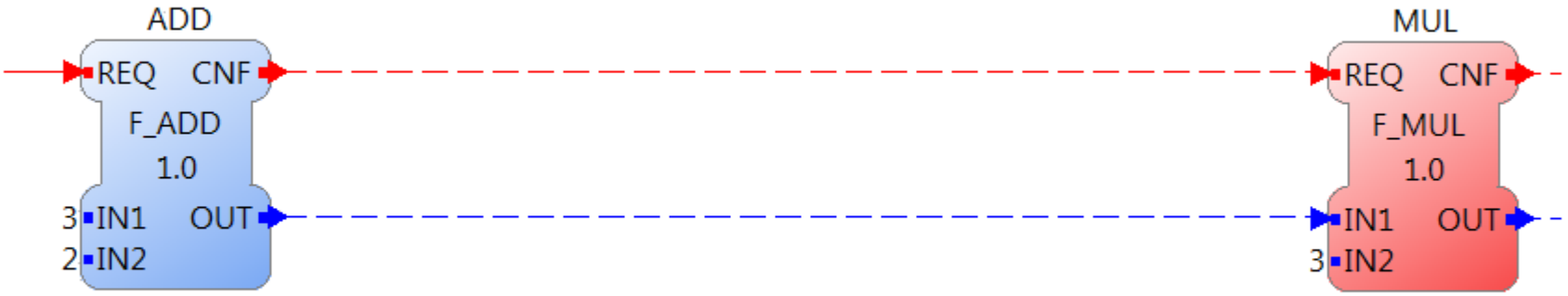
- is straightforward.
- allows multiple interaction points within the IEC 61499 Application.
- allows full exploitation of the Modbus's Master/Slave scheme.
- hides the exact details of communication protecting the inexperienced user.

Modbus Protocol in 4DIAC

The proposed method of implementation can be used in other Master/Slave communication protocols as well giving a universal solution to their IEC 61499 or 4DIAC's integration.

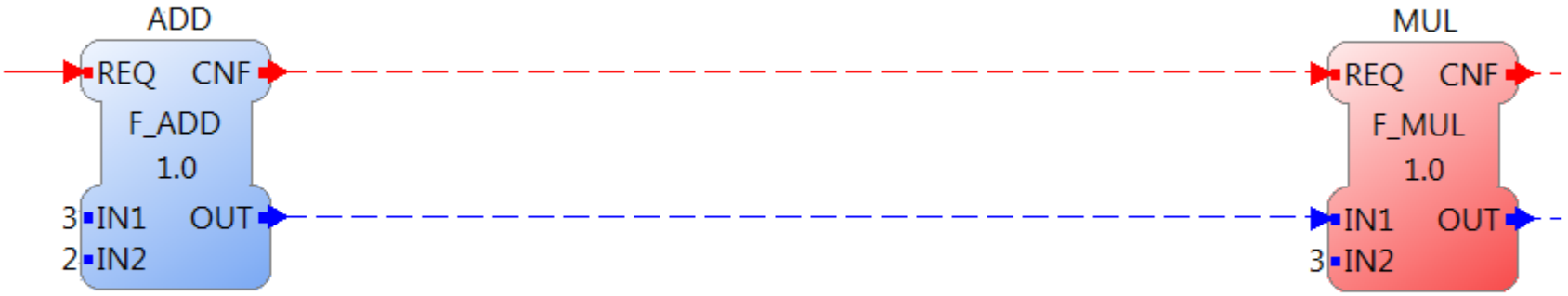
The increased complexity however calls for an automation option in the usage of communication protocols within the IEC 61499 Applications, something rather suggested by the standard itself.

Automated Communication Scheme



Assume that we want two Function Blocks from different devices to communicate with each other.

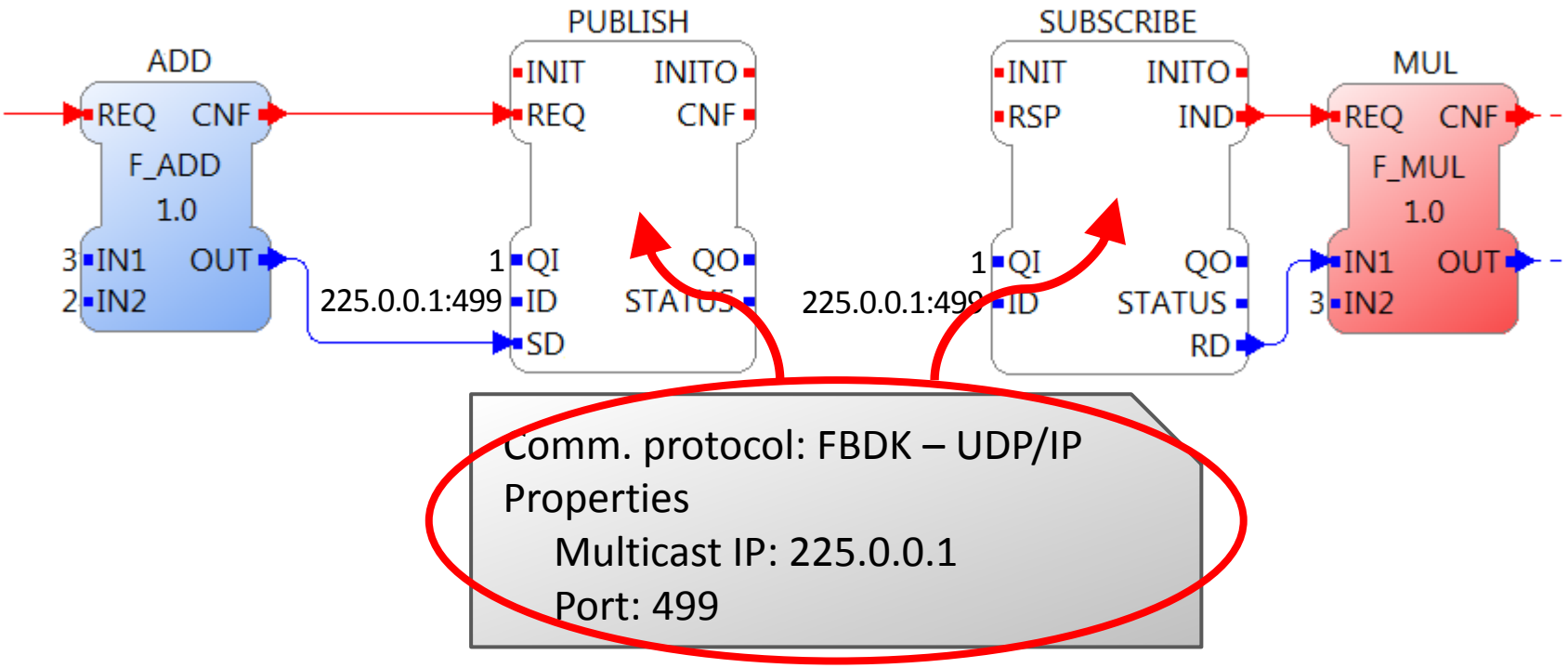
Automated Communication Scheme



Comm. protocol: FBDK – UDP/IP
Properties
Multicast IP: 225.0.0.1
Port: 499

The IEC 61499 Application programmer should be able to just set some properties of the preferred communication protocol.

Automated Communication Scheme

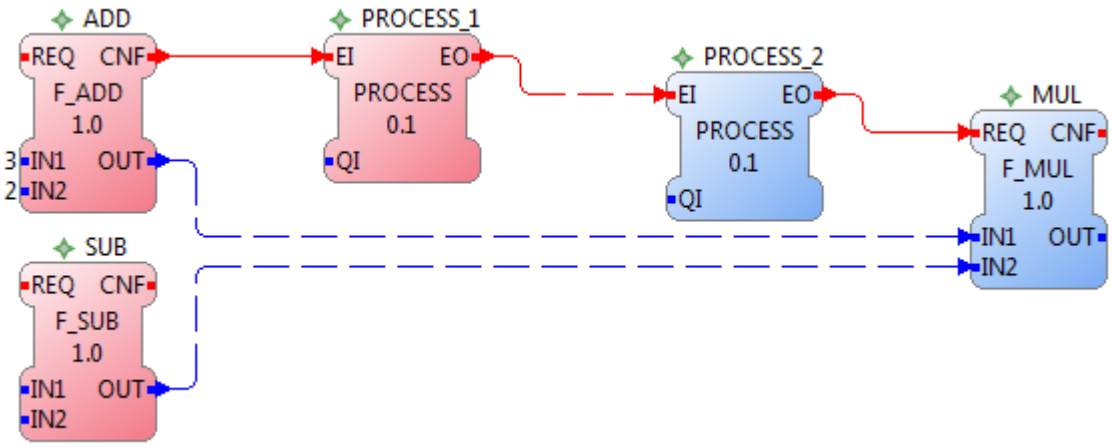


The development environment should offer an option to add and set the Communication Function Blocks automatically utilizing the programmers options.

Automated Communication Scheme

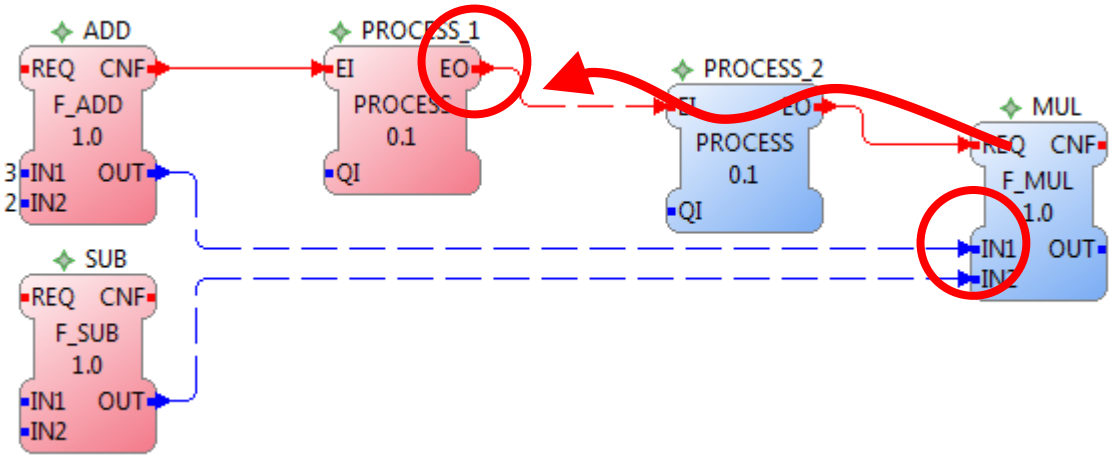
To implement such feature for the FBDK – UDP/IP protocol we would have to automatically add a pair of PUBLISH/SUBSCRIBE FBs for every device interconnection.

Automated Communication Scheme



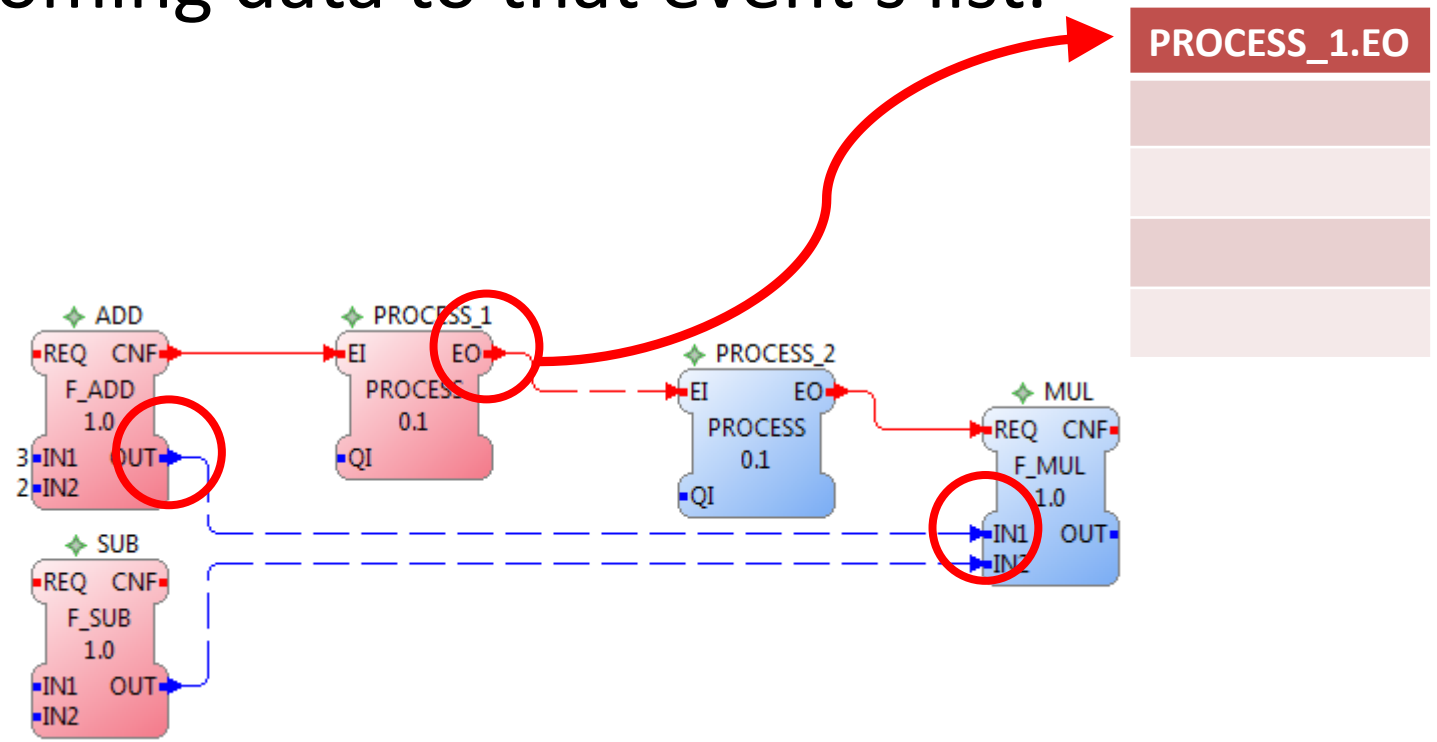
Automated Communication Scheme

For each incoming data we trace back the related outcoming event …



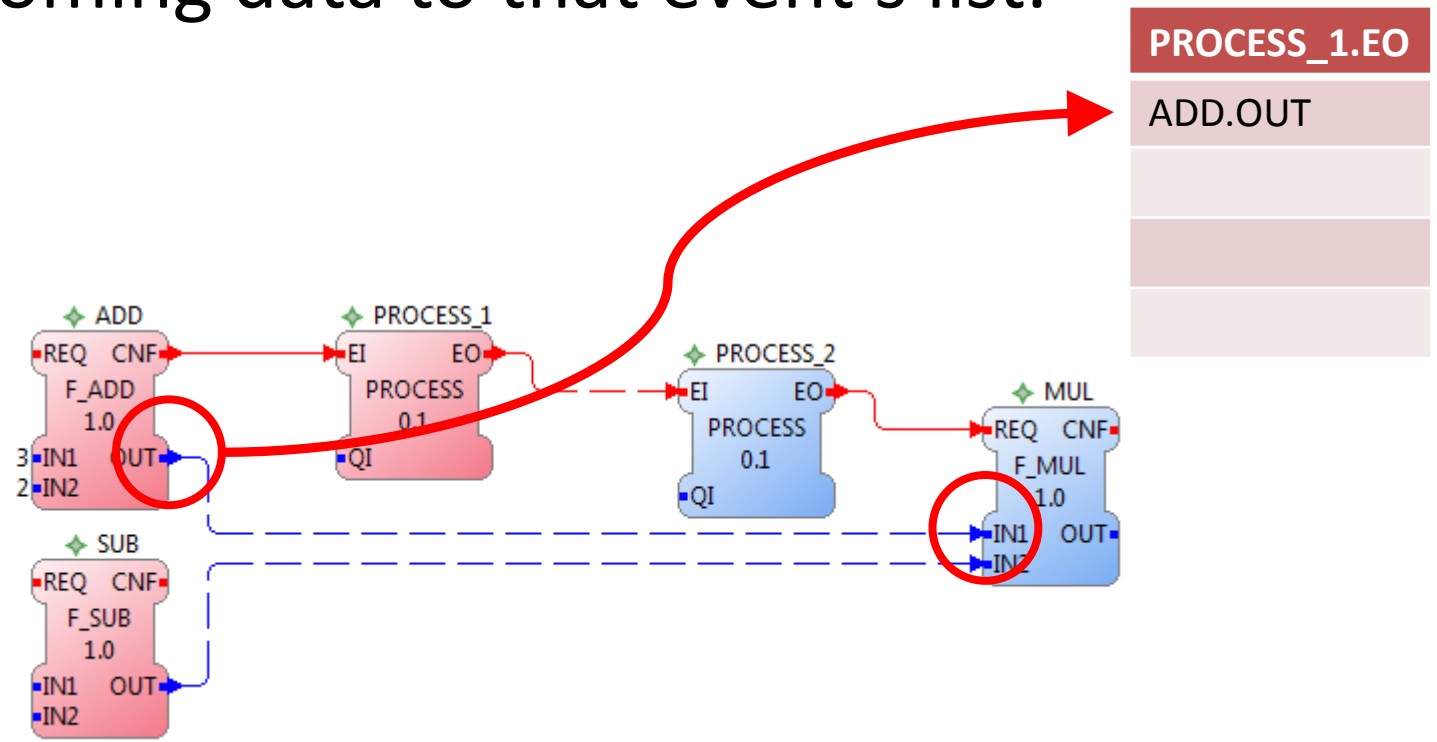
Automated Communication Scheme

For each incoming data we trace back the related outcoming event and append the outcoming data to that event's list.



Automated Communication Scheme

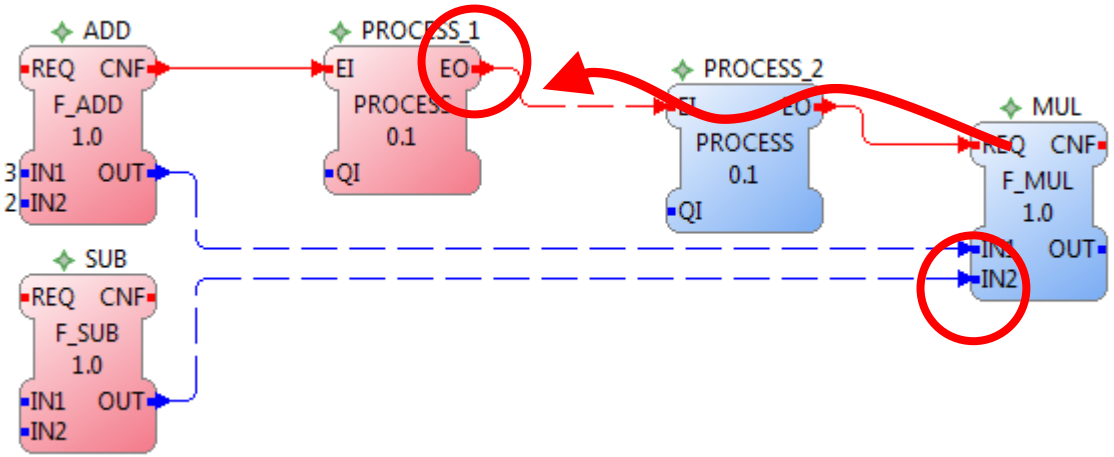
For each incoming data we trace back the related outcoming event and append the outcoming data to that event's list.



Automated Communication Scheme

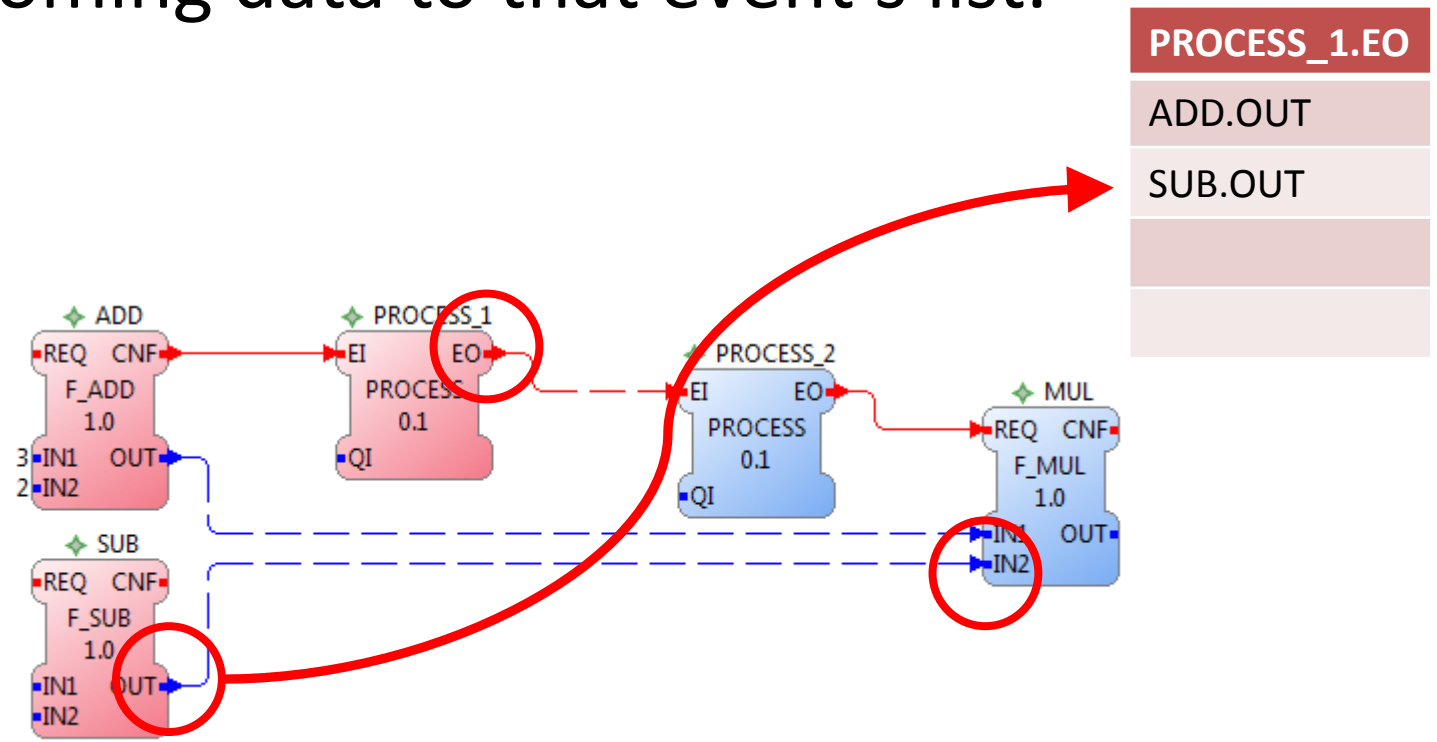
For each incoming data we trace back the related outcoming event and append the outcoming data to that event's list.

PROCESS_1.EO
ADD.OUT



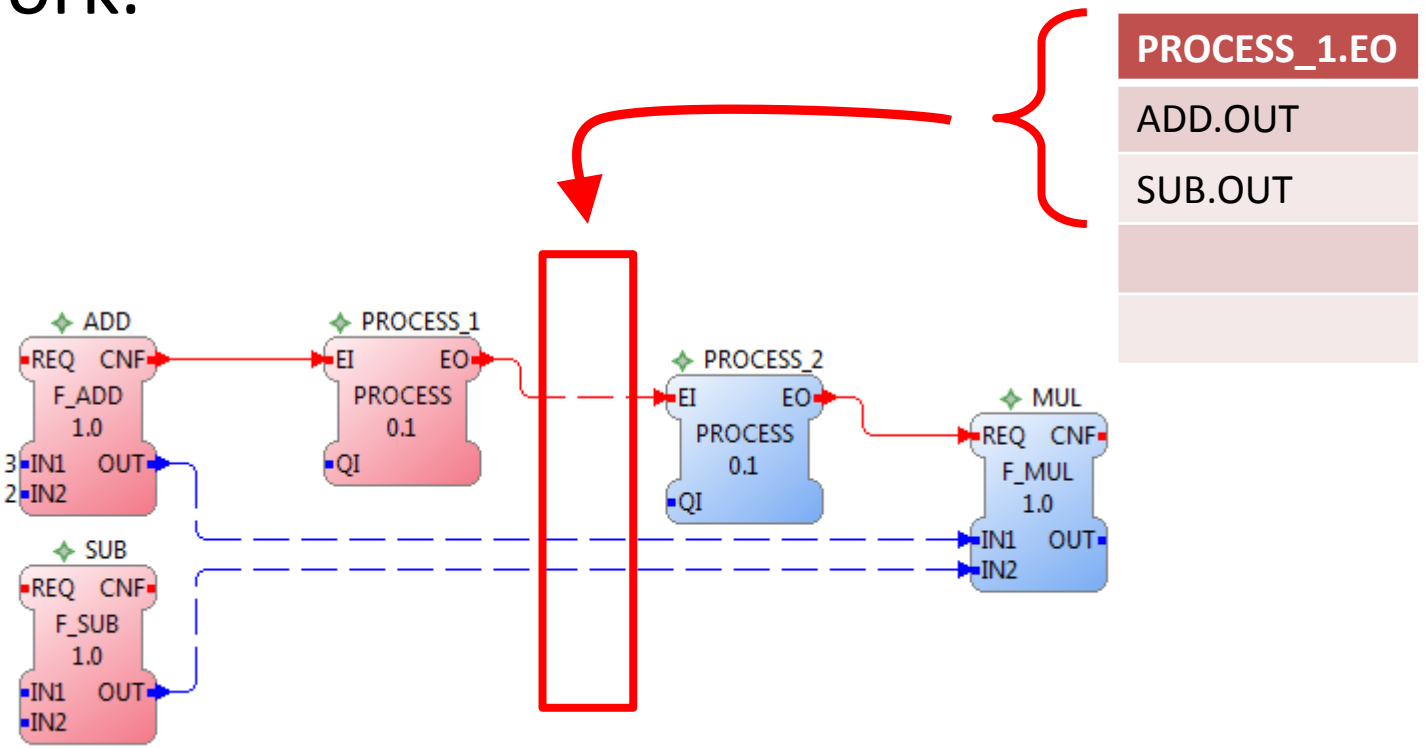
Automated Communication Scheme

For each incoming data we trace back the related outcoming event and append the outcoming data to that event's list.

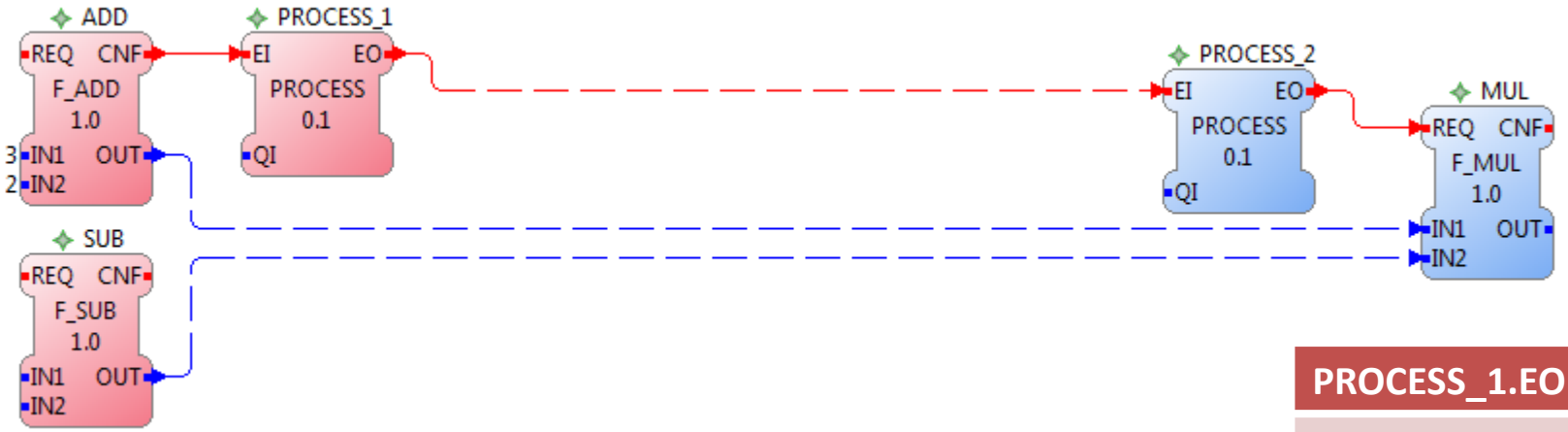


Automated Communication Scheme

The event list corresponds to a point of device interconnection in the Function Block network.



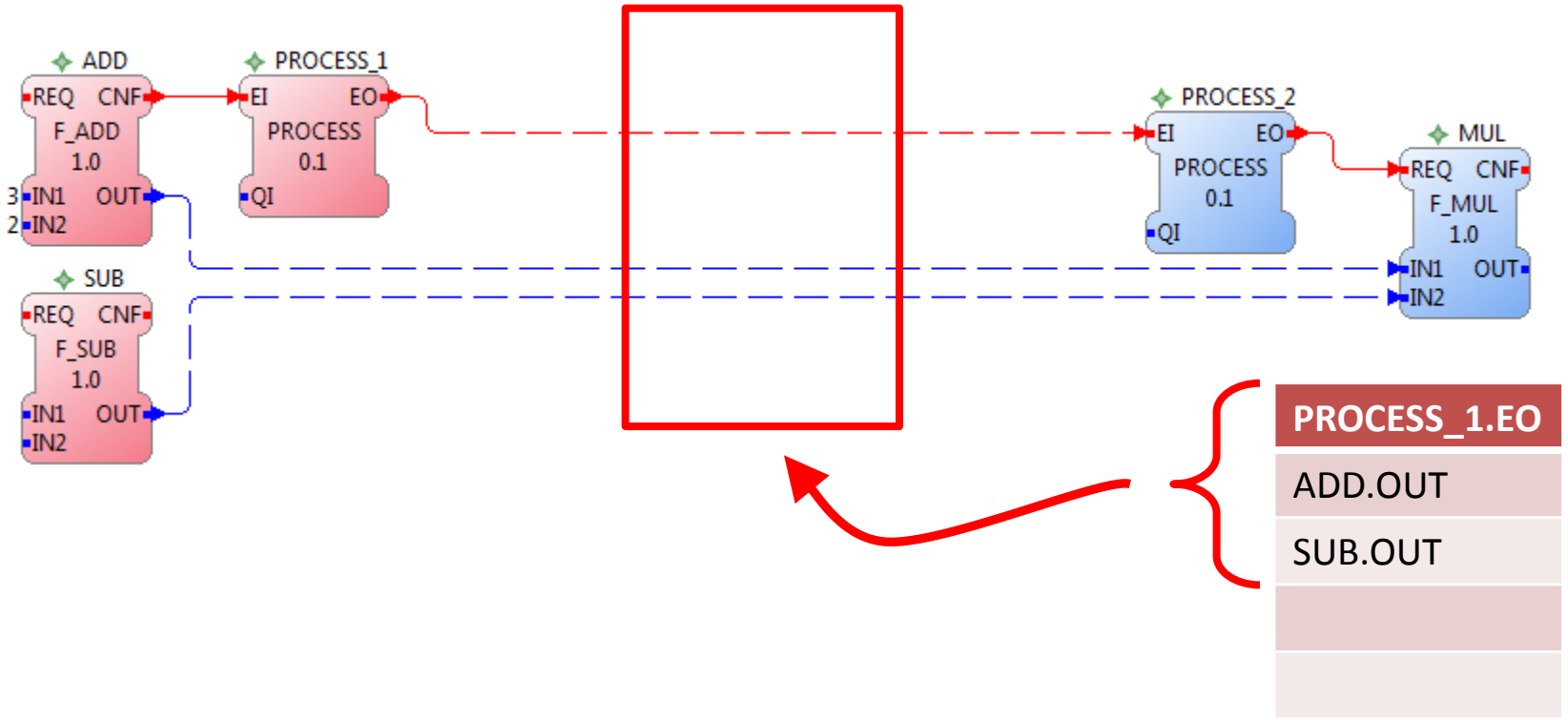
Automated Communication Scheme



PROCESS_1.EO
ADD.OUT
SUB.OUT

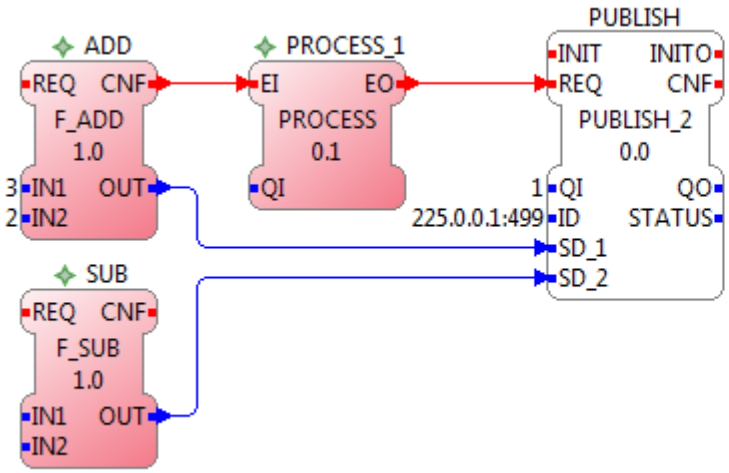
We use this list to add a PUBLISH/SUBSCRIBE pair.

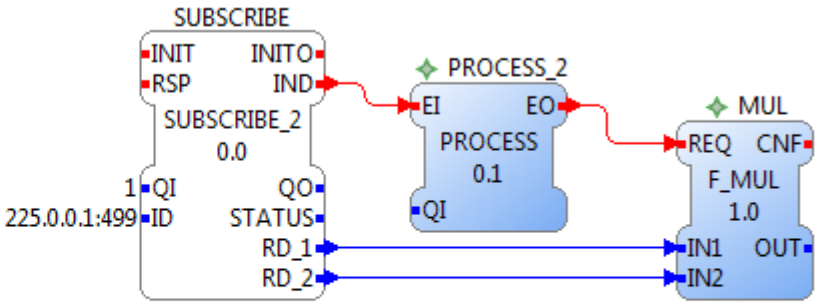
Automated Communication Scheme



We use this list to add a PUBLISH/SUBSCRIBE pair.

Automated Communication Scheme

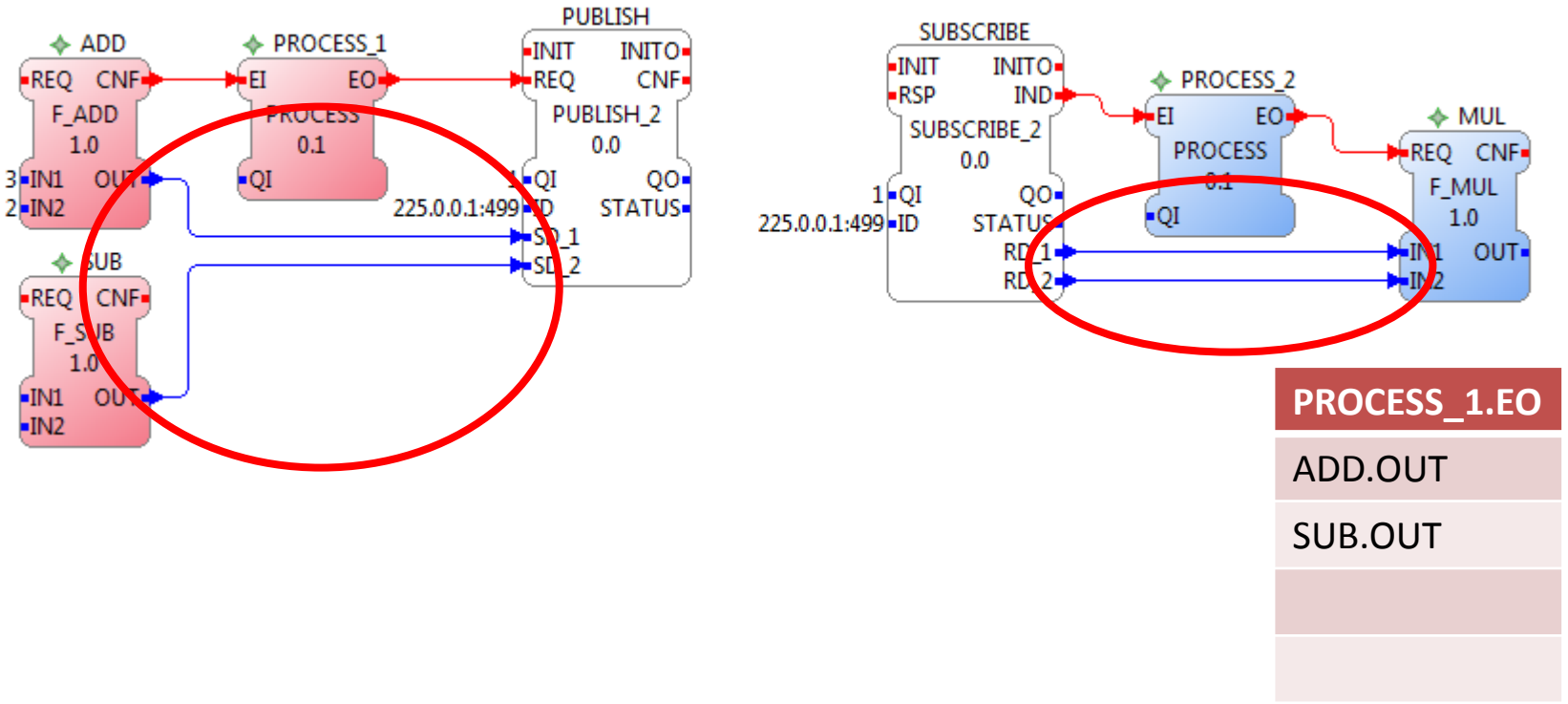




PROCESS_1.EO
ADD.OUT
SUB.OUT

We use this list to add a PUBLISH/SUBSCRIBE pair.

Automated Communication Scheme



PROCESS_1.EO
ADD.OUT
SUB.OUT

Thus a single PUBLISH/SUBSCRIBE pair handles all transferred data related to the same event.

Automated Communication Scheme

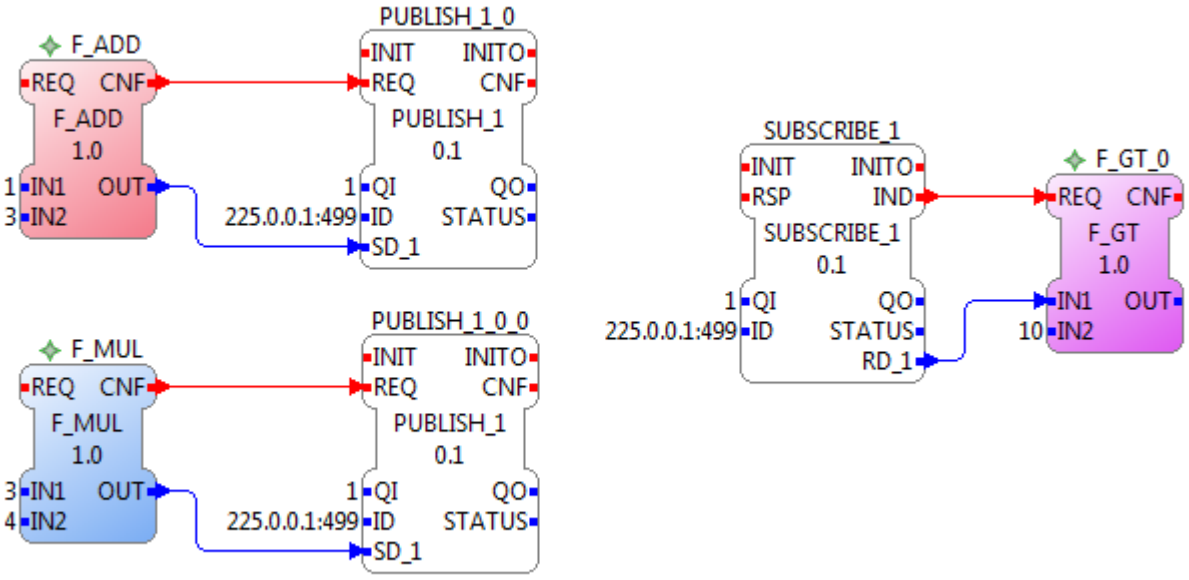
Overcoming some peculiarities the insertion of PUBLISH/SUBSCRIBE FBs can be automated for the FBDK – UDP/IP protocol in a way that:

- allows all possible Function Block configurations
- guarantees data is received before related events
- merges all data transactions related to a single event to a single UDP transmission
- exploits the UDP advantage of having a single PUBLISH and multiple SUBSCRIBE FBs

Automated Communication Scheme

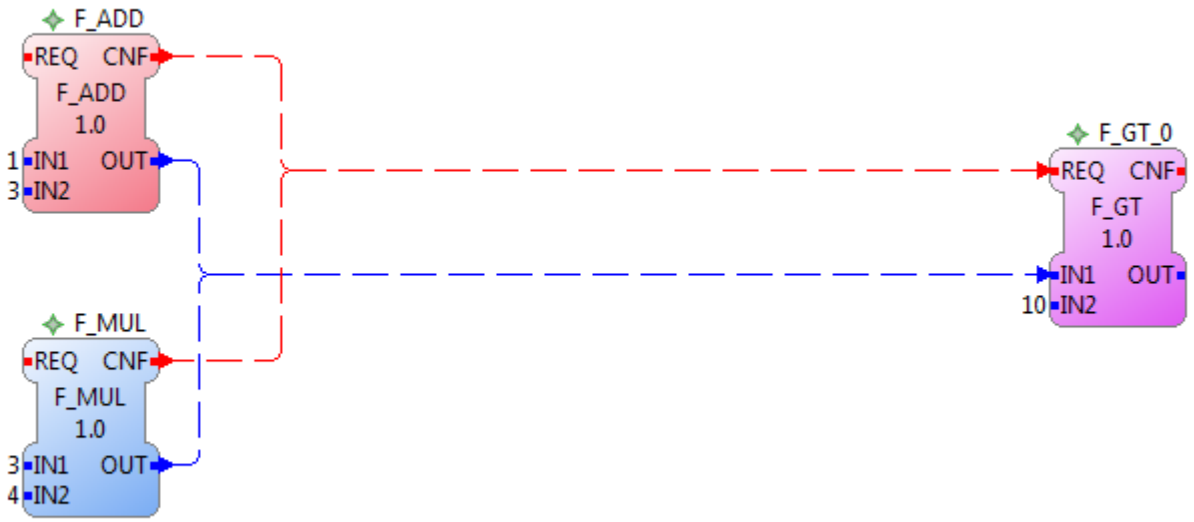
However it is impossible for an IEC 61499 Application to exploit the multiple PUBLISH – single SUBSCRIBE option of the UDP protocol in an automated manner.

Automated Communication Scheme



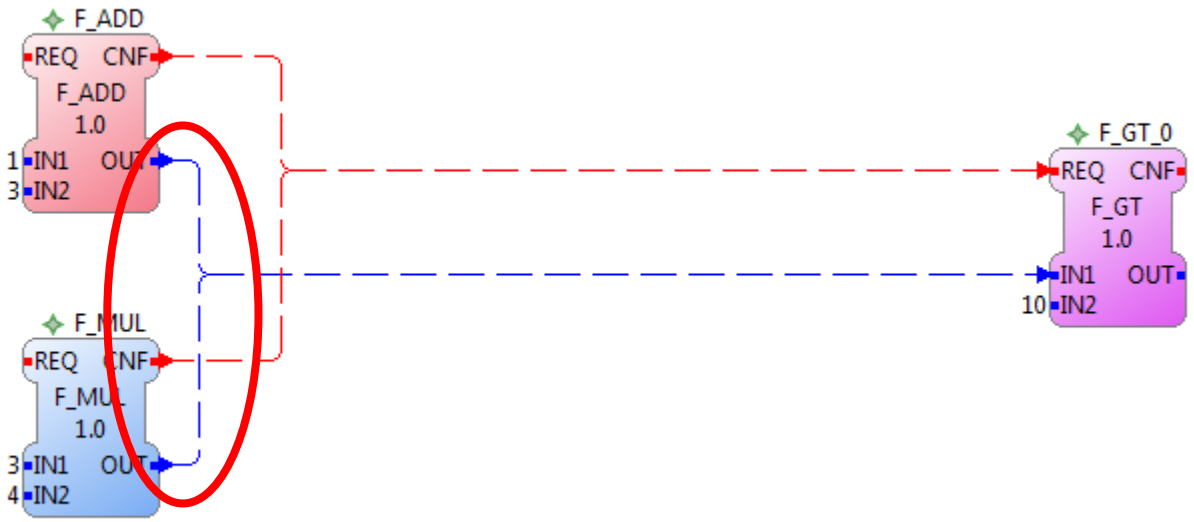
This configuration is legitimate according to IEC 61499 standard and allows a single SUBSCRIBE FB to receive data from two different PUBLISH FBs.

Automated Communication Scheme



However to allow such configuration to result automatically the original design wouldn’t be a legitimate IEC 61499 configuration.

Automated Communication Scheme



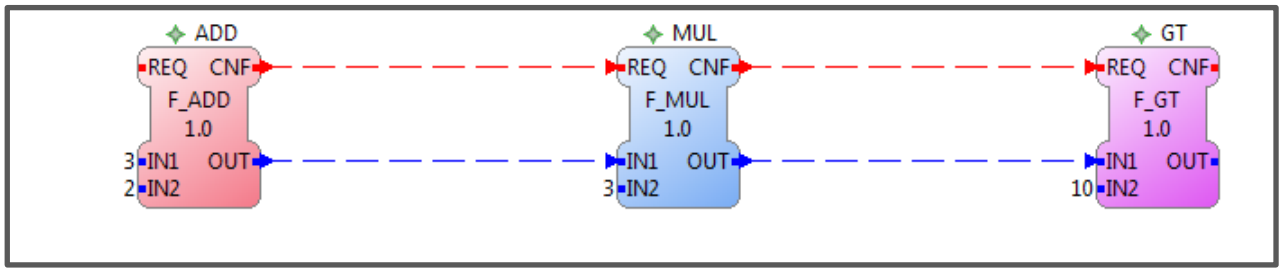
However to allow such configuration to result automatically the original design wouldn’t be a legitimate IEC 61499 configuration.

Automated Communication Scheme

4DIAC's Integrated Development Environment gives a convenient way to support the proposed automated scheme.

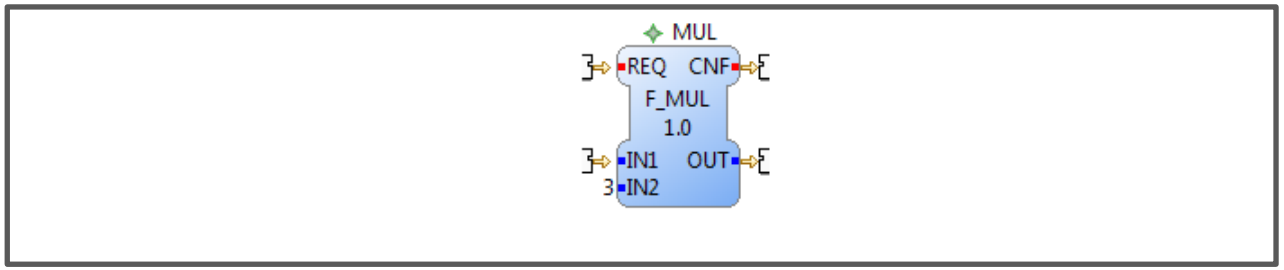
4DIAC IDE provides two views of Function Block networks, one of the Application's network and one of each Device's network.

Automated Communication Scheme



Application

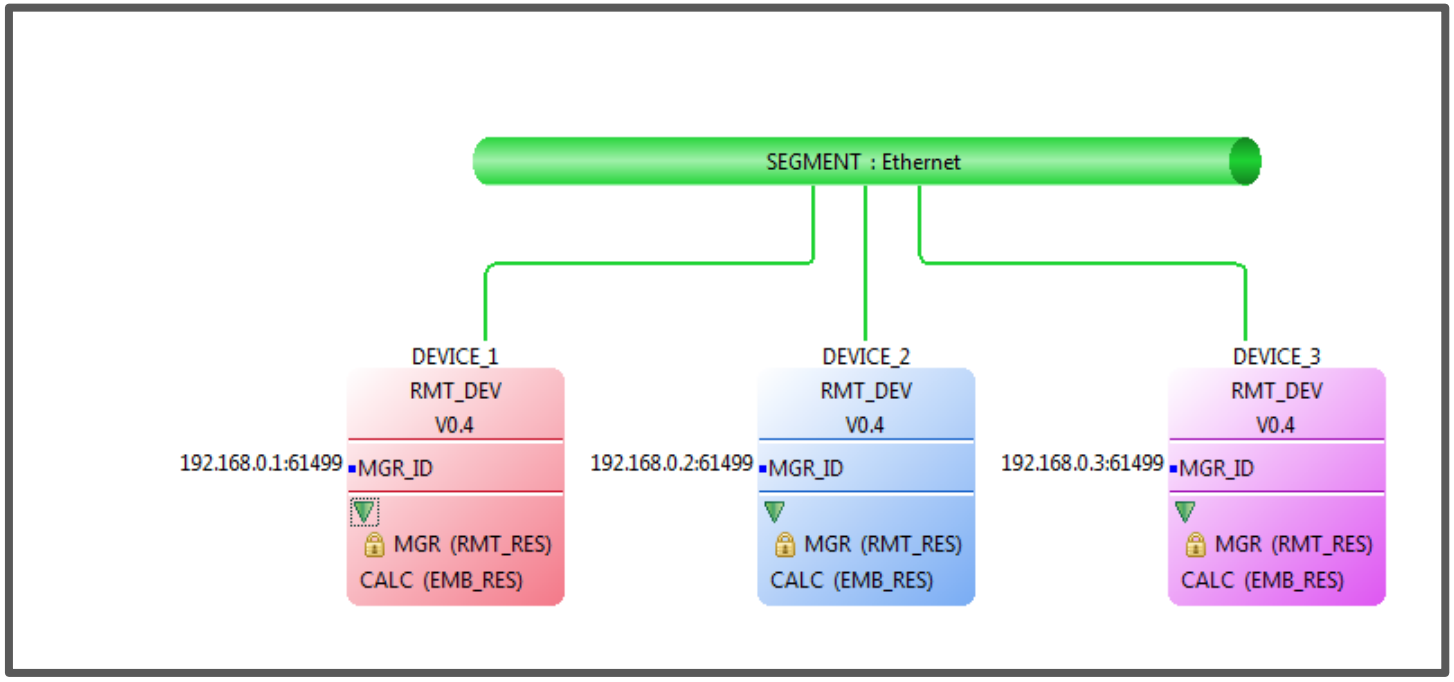
Device 1



Device 2

Device 3

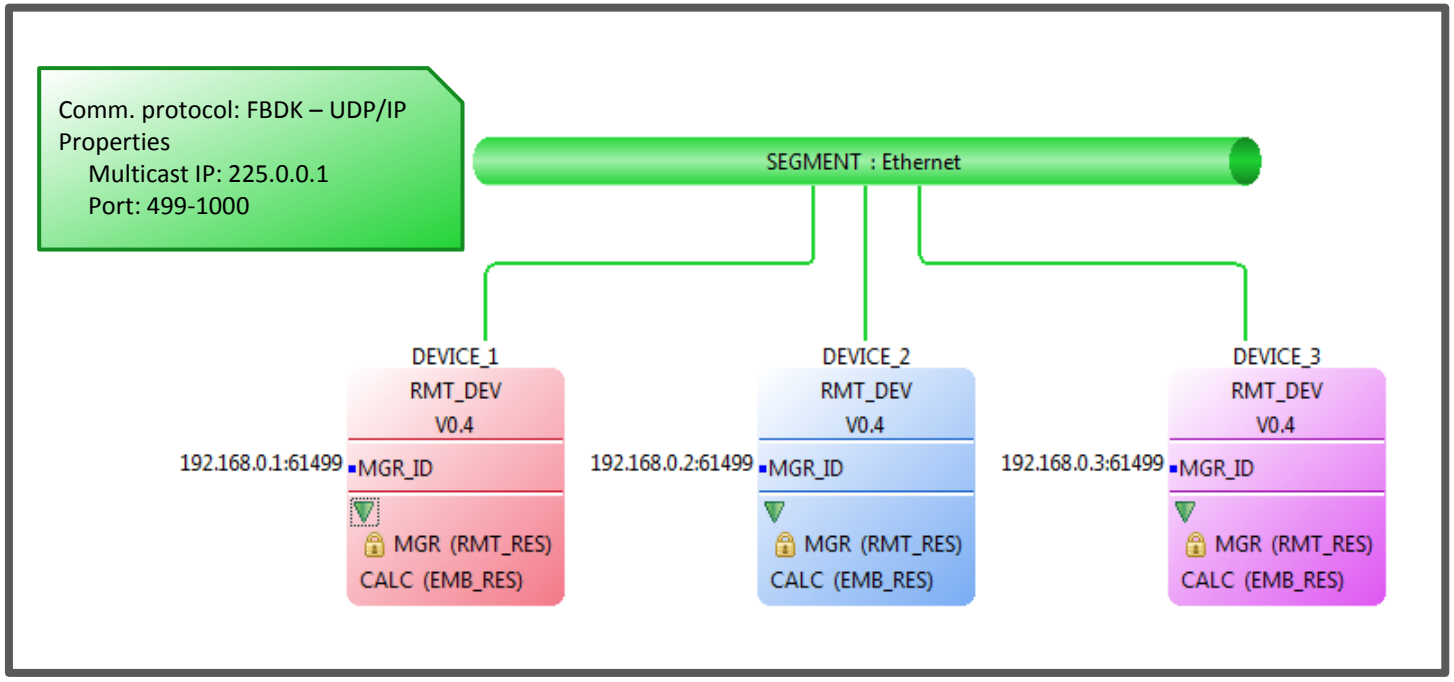
Automated Communication Scheme



System

Another view shows our system of devices. Here the user should be able to add a network segment and the properties of his preferred communication protocol.

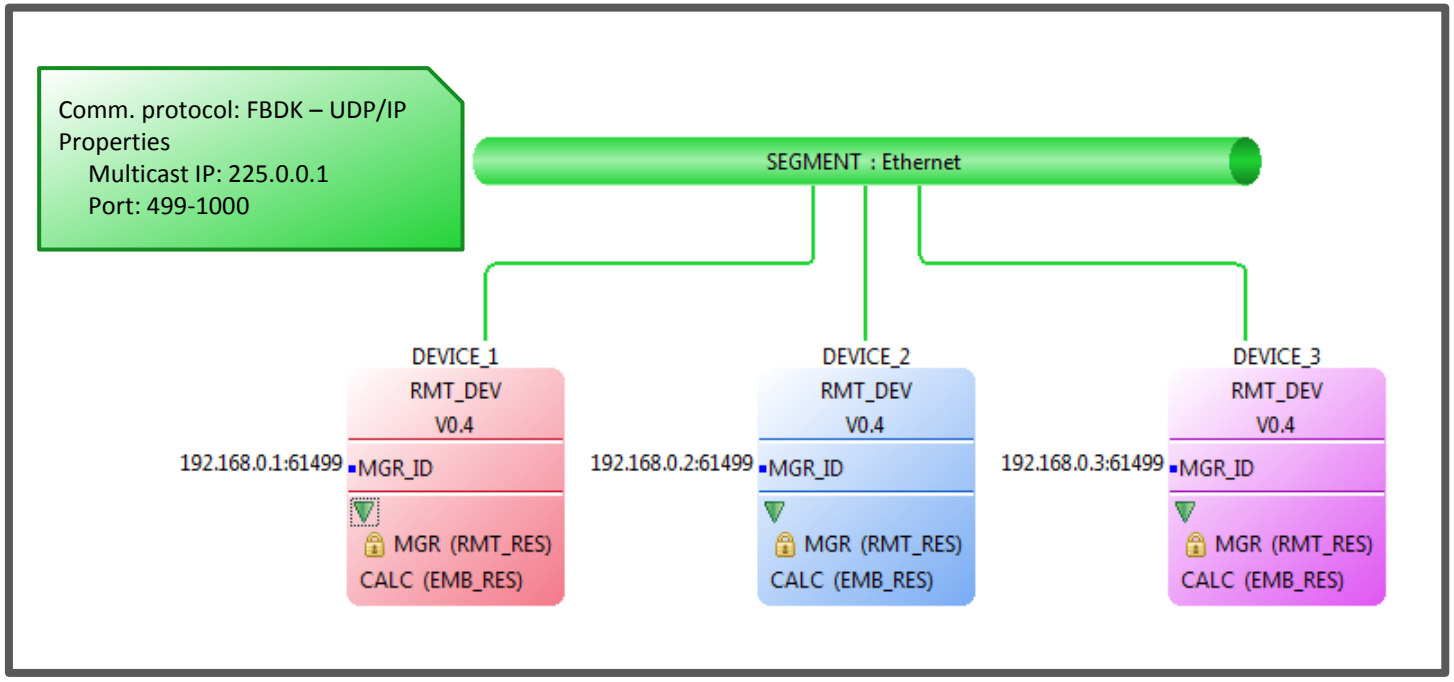
Automated Communication Scheme



System

Another view shows our system of devices. Here the user should be able to add a network segment and the properties of his preferred communication protocol. e.g. FBDK – UDP/IP.

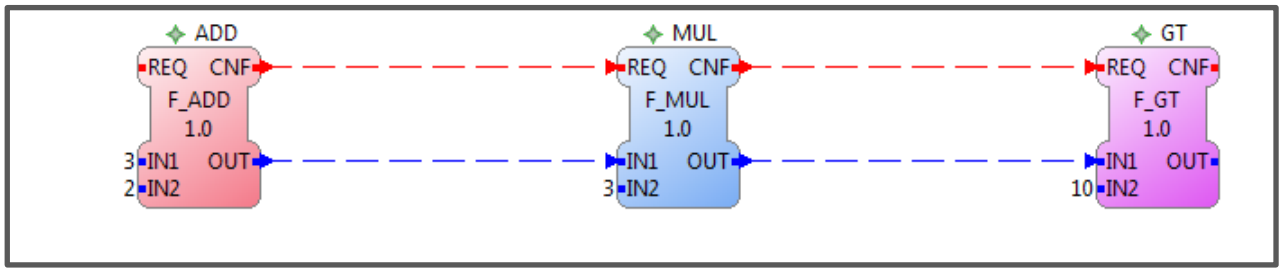
Automated Communication Scheme



System

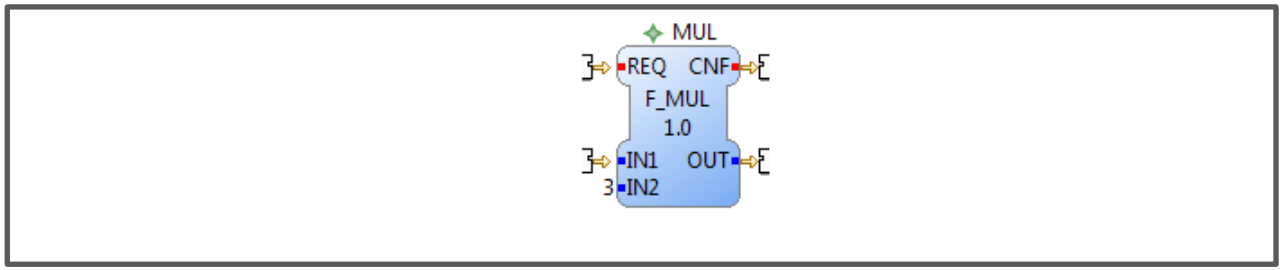
Another view shows our system of devices. Here the user should be able to add a network segment and the properties of his preferred communication protocol. e.g. FBDK – UDP/IP.
The CFBs would be added automatically to each device.

Automated Communication Scheme



Application

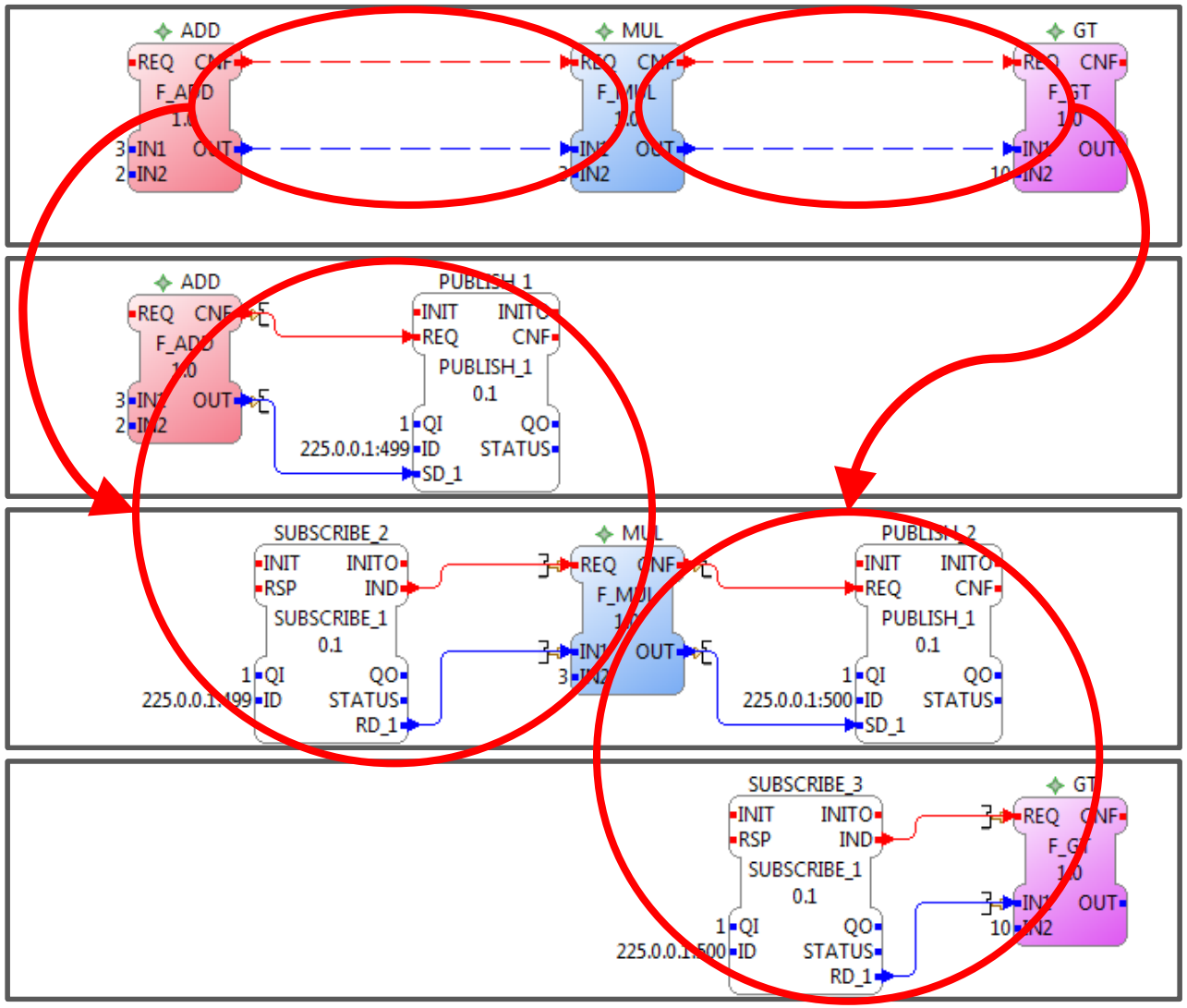
Device 1



Device 2

Device 3

Automated Communication Scheme



Application

Device 1

Device 2

Device 3

Automated Communication Scheme

The proposed automated scheme for the FBDK – UDP/IP protocol is fairly simple and poses few limitations.

However this shouldn’t be expected of other protocols that have a more complicated implementation.

In particular an attempt to create such a scheme for the Modbus protocol would have to resolve the issue of having a single Master device to handle all communication between Slave devices.

Conclusions

The resolution of the implementation of the Modbus protocol and the automated FBDK – UDP/IP scheme comes to the conclusion that the IEC 61499 standard requires a new perspective to be acquired.

Then more sophisticated solutions can arise to deal with the IEC 61499 implementation's particular problems.

Conclusions

4DIAC project on the other hand introduces a very efficient model for implementing, applying and using the IEC 61499 standard.

Getting improved with each new feature it has the potential of becoming a powerful, easy to use tool for microcontroller programming.