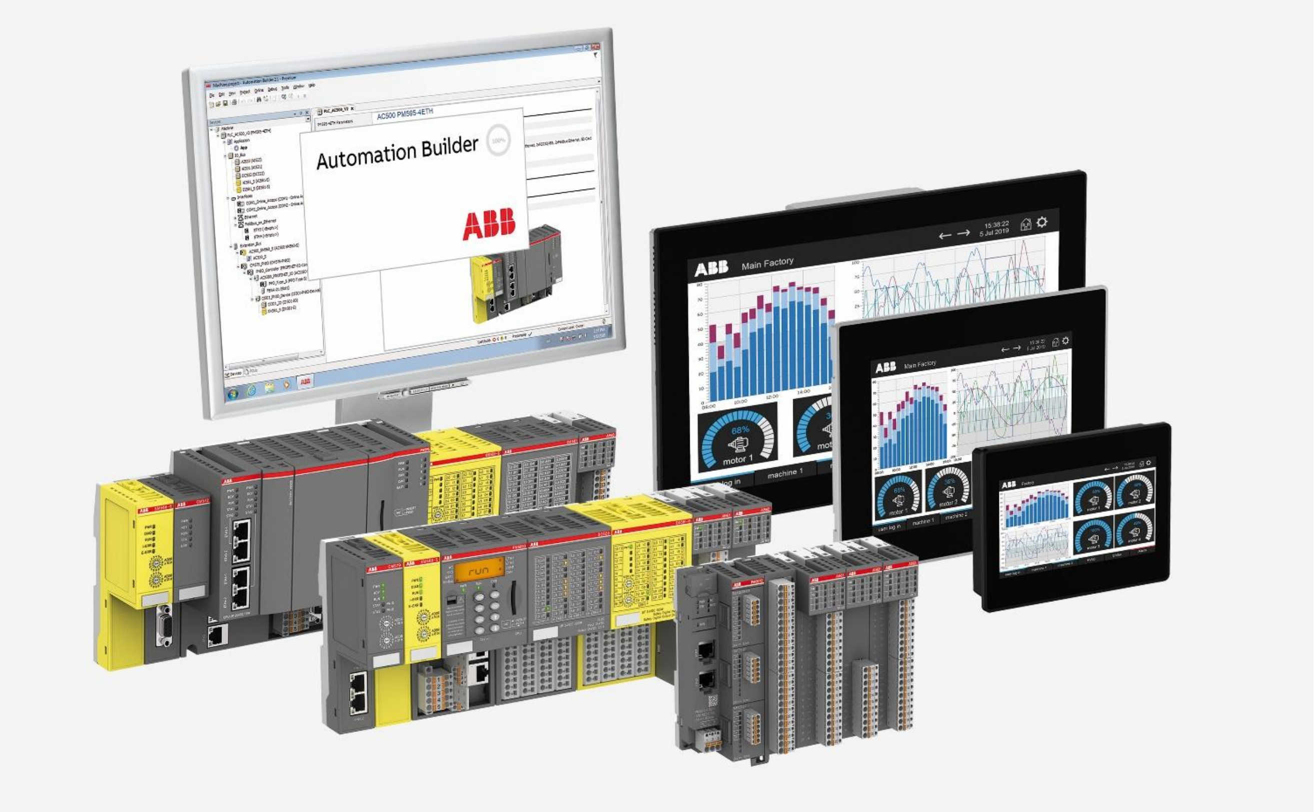
APPLICATION EXAMPLE: **AE-1102**

AC500 UDP

COMMUNICATION VIA UDP PROTOCOL



Contents

1 Disclaimer

A. For customers domiciled outside Germany /

Für Kunden mit Sitz außerhalb Deutschlands

„Warranty, Liability:

The user shall be solely responsible for the use of this products described within this file. ABB shall be under no warranty whatsoever. ABB's liability in connection with application of the products or examples provided or the files included within these products, irrespective of the legal ground, shall be excluded. The exclusion of liability shall not apply in the case of intention or gross negligence. The present declaration shall be governed by and construed in accordance with the laws of Switzerland under exclusion of its conflict of laws rules and of the Vienna Convention on the International Sale of Goods (CISG)."

„Gewährleistung und Haftung:

Der Nutzer ist allein für die Verwendung des in diesem Dokument beschriebenen Produkte und beschriebenen Anwendungsbeispiele verantwortlich.

ABB unterliegt keiner Gewährleistung. Die Haftung von ABB im Zusammenhang mit diesem Anwendungsbeispiel oder den in dieser Datei enthaltenen Dateien - gleich aus welchem Rechtsgrund - ist ausgeschlossen. Dieser Ausschluss gilt nicht im Falle von Vorsatz oder grober Fahrlässigkeit. Diese Erklärung unterliegt Schweizer Recht unter Ausschluss der Verweisungsnormen und des UN-Kaufrechts (CISG)."

B. Nur für Kunden mit Sitz in Deutschland

„Gewährleistung und Haftung:

Die in diesem Dokument beschriebenen Anwendungsbeispiele oder enthaltenen Dateien beschreiben eine mögliche Anwendung der AC500 bzw. zeigen eine mögliche Einsatzart. Sie stellen nur Beispiele für Programmierungen dar, sind aber keine fertigen Lösungen. Eine Gewähr kann nicht übernommen werden.

Der Nutzer ist für die ordnungsgemäße, insbesondere vollständige und fehlerfreie Programmierung der Steuerungen selbst verantwortlich. Im Falle der teilweisen oder ganzen Übernahme der Programmierbeispiele können gegen ABB keine Ansprüche geltend gemacht werden.

Die Haftung von ABB, gleich aus welchem Rechtsgrund, im Zusammenhang mit den Anwendungsbeispielen oder den in dieser Datei enthaltenen Beschreibung wird ausgeschlossen. Der Haftungsausschluss gilt jedoch nicht in Fällen des Vorsatzes, der groben Fahrlässigkeit, bei Ansprüchen nach dem Produkthaftungsgesetz, im Falle der Verletzung des Lebens, des Körpers oder der Gesundheit oder bei schuldhafter Verletzung einer wesentlichen Vertragspflicht. Im Falle der Verletzung einer wesentlichen Vertragspflicht ist die Haftung jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht zugleich ein anderer der in Satz 2 dieses Unterabsatzes erwähnten Fälle gegeben ist. Eine Änderung der Beweislast zum Nachteil des Nutzers ist hiermit nicht verbunden.

Es gilt materielles deutsches Recht unter Ausschluss des UN-Kaufrechts."

2 Introduction

2.1 Scope of the document

This application example shows how the UDP communication between multiple AC500 PLCs can be realized. It contains three AC500 PLCs, two V3 and one V2 PLC. With this setup, it’s possible to show the communication between V2 and V3, as well as V3 to V3. All three communications, unicast, multicast and broadcast are explained. If for example the V2 communication is not needed, the certain device can be deleted from the project and, except of IP settings, there is no further software change needed.

2.2 Compatibility

The application example explained in this document has been used with the below engineering system versions. They should also work with other versions, nevertheless some small adaptations may be necessary, for future versions.

- AC500 V2 and V3 PLC
- Automation Builder 2.7.0 or newer

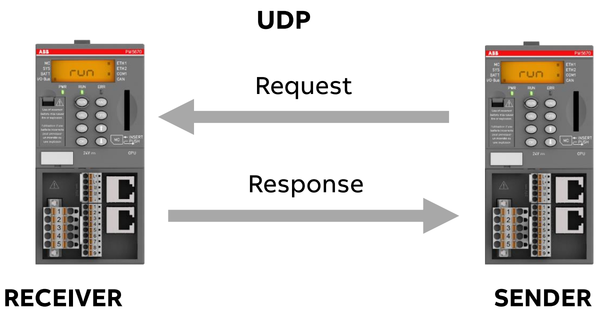
2.3 Acronyms

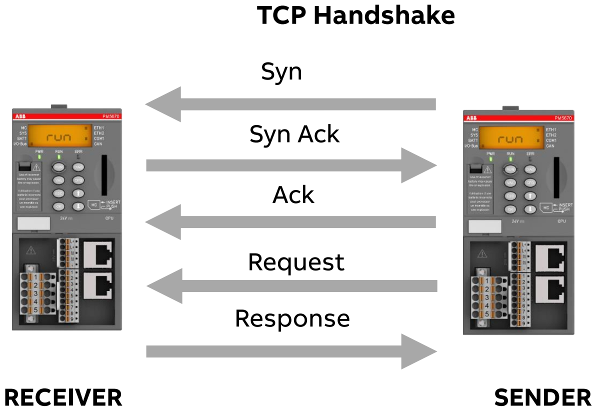
Peer	Specifies a socket on the system which will be needed to communicate via ethernet.
UDP	User Datagram Protocol. Transition protocol which will be used in this example.
OSI	Open system interconnection
TCP	Transmission control protocol
PRG	Program block
ST	Structured text
FB	Function block

3 UDP architecture

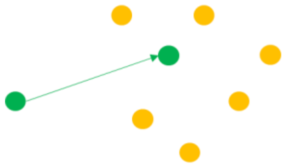
UDP is a transmission protocol like TCP. Both protocols are widely used in network communication. The application purpose of UDP is to transfer data quickly between applications and devices without waiting for the recipient's consent. A big difference also is that the connection in TCP is established by a 3-way handshake. This handshake secures the connection, but is also slower. In UDP there is no real connection to the opponent, the telegram is just sent without observation if it has reached the other side. The opponent is only responding if the request was a command to give a response.

UDP	TCP
Fast data transfer	Slower, bigger Header
Order of packages can differ	Packages will be sent in correct order so they can be put together again easily
Fire and forget Principe, not connection oriented	Successful data transmission is guaranteed, connection is checked
Unicast, multicast, broadcast	Only one receiver

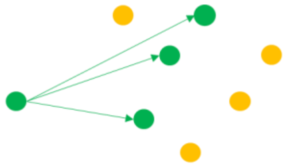




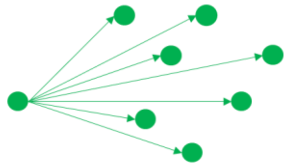
UDP also has three different modes to send its data. Unicast is a direct communication between two devices, Multicast uses groups and broadcast sends data to everyone.



Unicast



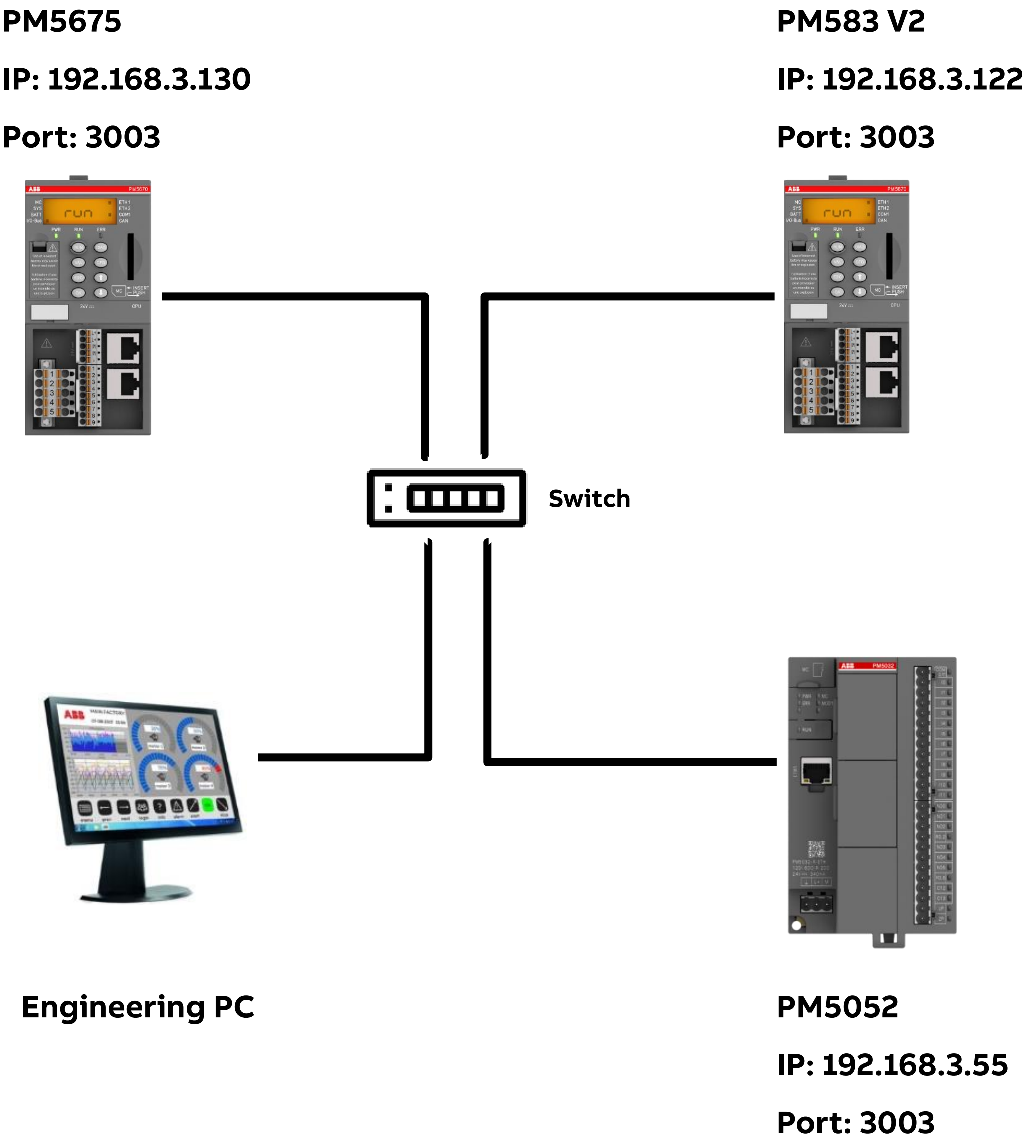
Multicast



Broadcast

4 Hardware Setup

The example Setup contains three AC500 PLCs, two of them are V3 and one is V2.
If, for example, only the V3 communication is needed, the V2 PLC does not need to be connected.



5 Software

In this application example, all three PLC’s use port 3003. It is done to easier switch between PLC’s and send modes. Also, for special usecase “anybody send data to everyone as well as receive data from anyone” all PLC have to use the same port.

Note for multicast and broadcast:

To prevent a PLC to receive its own sent data, the UDPWrapper has a internal filter functionality which checks the source IP in a UDP telegram and deletes the telegram if sender IP and own IP is the same.

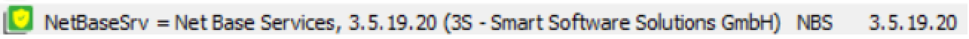
5.1 UDPWrapper function block

In the following usecases, PLC will send and receive data. For every usecase the “UDPWrapper” function block is used. If one PLC should not send but only receive data, there is a separate “EnableSend” input which can be set to FALSE. The send/receive behavior of the function block works as followed:

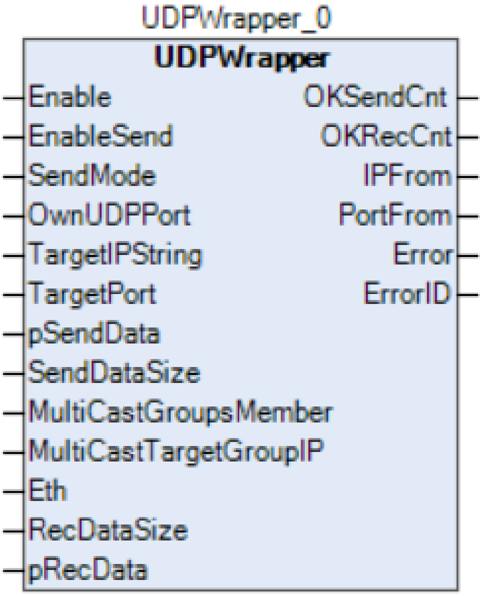
- If “EnableSend” is at FALSE, “UDPWrapper” will just receive. Data can be retrieved from “pRecData” pointer input and “RecDataSize” input.
- If there is no need for receiving data, the “pRecData” and “RecDataSize” simply can be ignored.
- If there is a need to send and receive data, the “EnableSend” can be set to TRUE. The function block will then send then Data additionally to the receiving and after successful sending will flip the input to FALSE again.
- All other In-/ and outputs not related to a certain purpose can be ignored.

Note:

UDPWrapper uses the Net base Services library.



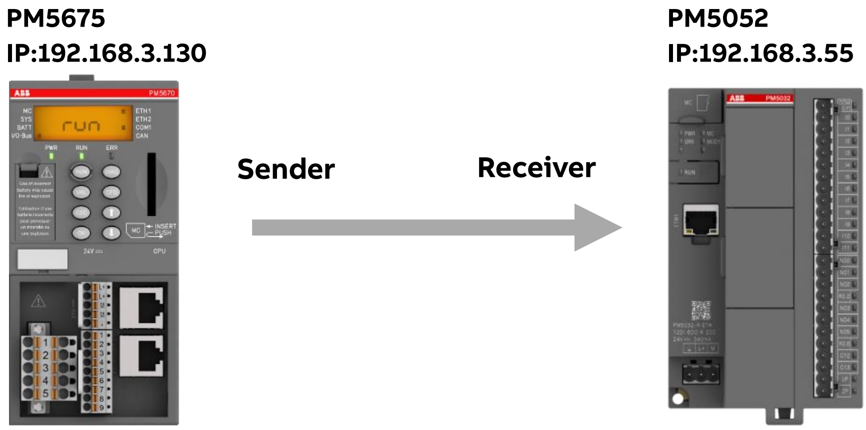
NetBaseSrv = Net Base Services, 3.5.19.20 (3S - Smart Software Solutions GmbH) NBS 3.5.19.20



IN/OUT	Variable	Comment
IN	Enable	Enables the function block, before changing “SendMode” set this imput to FALSE, after change to TRUE again.
IN	EnableSend	If set to TRUE, “pSendData” will be sent.
IN	SendMode	Choose between unicast, multicast and broadcast. All additional steps are handled by the function block.
IN	OwnUDPPort	Specifies the own socket port of the PLC.
IN	TargetIPString	Only Unicast: Target PLC IP.
IN	TargetPort	Unicast,Multicast,broadcast: Target socket port.
IN	pSendData	Pointer to Data which sould be send. When PLC should only receive, ignore this input.
IN	SendDataSize	Size of the Array or data which should be sent.
IN	MultiCastgroupsMember	Multicast only: Dynamic array with multicast group IP’s this device should be in.
IN	MultiCastTargetGroupIP	Multicsat only: The target group to send data.
IN	Eth	Hardware ethernet port
IN	RecDataSize	Size of the received Data.
IN	pRecData	Pointer to received data.
OUT	OKSendCnt	Counter of successful sent data.
OUT	OKRecCnt	Counter of successful received data.
OUT	IPFrom	Remote IP of the opponent device
OUT	PortFrom	Remote Socket port of the opponent device.
OUT	Error	Indicates an error in the FB
OUT	ErrorID	Returns the error code for better investigation

5.2 Usecase 1: Unicast from PM5675 to PM5052 eCo

To use unicast functionality, select “Unicast” in variable “SendMode” when online. To switch to another send mode, set “ProgramEnable” to FALSE first. Then select desired mode and set TRUE afterwards. This will reset the UDPWrapper. “Unicast” is selected per default.



In the first example, one PLC is sending a direct UDP message to the other PLC. The sending interval is 1 sec which is triggered from the “Timer” FB. The PM5052 project is almost the same, but the “Timer” FB is not used, so it will not send anything. The parameterization is done in “Unicast” action below the main PRG.

Note:

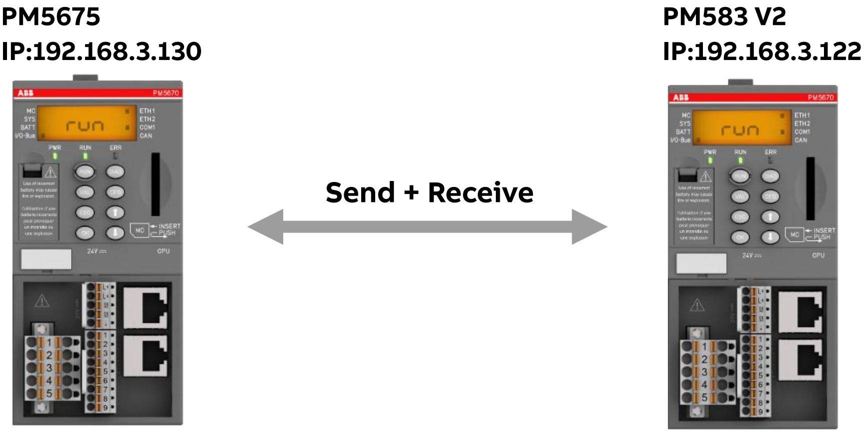
Altough the PM5675 is sending only, the reading section is still implemted. If not necessary, just delete the reading section in action unicast.

5.3 Usecase 2: Unicast between PM5675 and PM583 V2

Communication between V2 and V3 works similar than in V3, only the project on the PM583 is structured differently. To work with this usecase, select “unicast” in “SendMode” and change the “UDP.TargetIPString” to the V2 PLC IP. Then set “ProgramEnable” to TRUE.

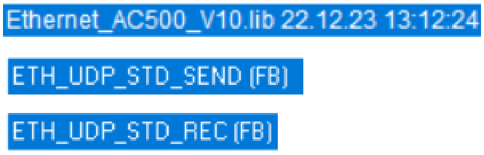
Note:

In V2 there is no multicast functionality available, only unicast and broadcast can be done.



Note:

PM583 UDP functionality uses library Ethernet_AC500_V10.lib with function blocks ETH_UDP_STD_SEND(FB) and

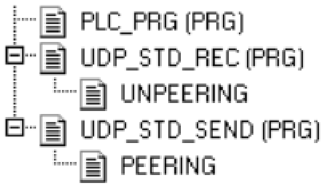

Ethernet_AC500_V10.lib 22.12.23 13:12:24

ETH_UDP_STD_SEND (FB)

ETH_UDP_STD_REC (FB)

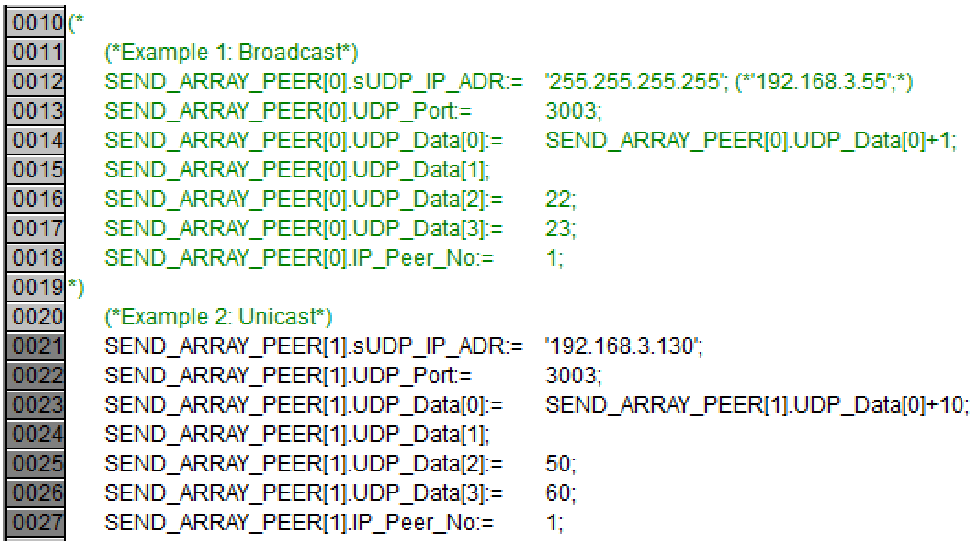
The sending and receiving functionality in the PM5675 V3 is the same like in usecase 1.

The V2 project consists of two parts, the “UDP_STD_SEND” for sending and “UDP_STD_REC” for receiving. Both work in the same way, the function block itself handles the send / receive functionality and a separate “peering” / “unpeering” action takes care of the data handling.


PLC_PRG (PRG)
UDP_STD_REC (PRG)
UNPEERING
UDP_STD_SEND (PRG)
PEERING

5.3.1 Sending UDP message in V2

The V2 send process works as follows. In “PEERING”, one or more datasets can be defined. In this example, there will be only one unicast. The broadcast dataset is commented, but can be used.



```
(*
    (*Example 1: Broadcast*)
    SEND_ARRAY_PEER[0].sUDP_IP_ADR:=   '255.255.255.255'; (*'192.168.3.55';*)
    SEND_ARRAY_PEER[0].UDP_Port:=      3003;
    SEND_ARRAY_PEER[0].UDP_Data[0]:=   SEND_ARRAY_PEER[0].UDP_Data[0]+1;
    SEND_ARRAY_PEER[0].UDP_Data[1];
    SEND_ARRAY_PEER[0].UDP_Data[2]:=   22;
    SEND_ARRAY_PEER[0].UDP_Data[3]:=   23;
    SEND_ARRAY_PEER[0].IP_Peer_No:=    1;
*)
    (*Example 2: Unicast*)
    SEND_ARRAY_PEER[1].sUDP_IP_ADR:=   '192.168.3.130';
    SEND_ARRAY_PEER[1].UDP_Port:=      3003;
    SEND_ARRAY_PEER[1].UDP_Data[0]:=   SEND_ARRAY_PEER[1].UDP_Data[0]+10;
    SEND_ARRAY_PEER[1].UDP_Data[1];
    SEND_ARRAY_PEER[1].UDP_Data[2]:=   50;
    SEND_ARRAY_PEER[1].UDP_Data[3]:=   60;
    SEND_ARRAY_PEER[1].IP_Peer_No:=    1;
```

It is possible, to perform a broadcast, as well as multiple unicasts. For this, uncomment the broadcast and multiply the unicast dataset. The sending functionality will send every second the next dataset in “SEND_ARRAY_PEER”

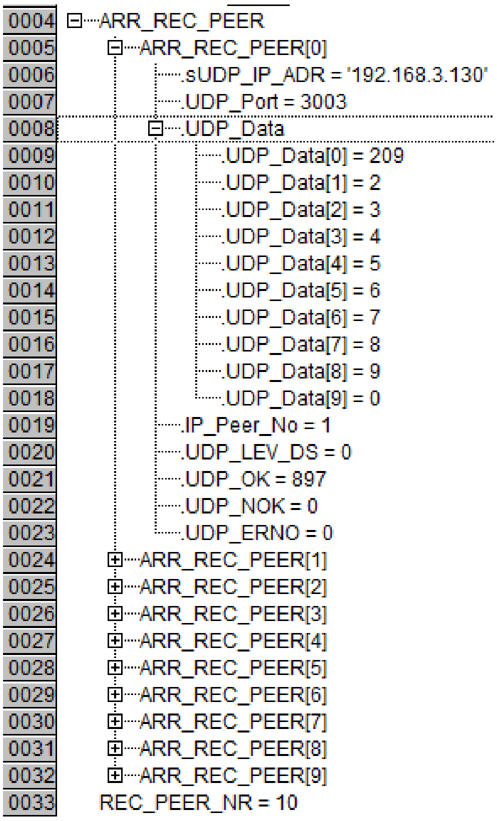
Note:

While multiplying the unicast datasets, it is important to increase also the index of “SEND_ARRAY_PEER”.

5.3.2 Receiving UDP message in V2

The receiving functionality of the PM583 V2 works similarly to the sending functionality. The “UDP_STD_REC” will receive data and “UNPEERING” will extract data from the receiving functionality and will store this data in an seperate array of struct “ARR_REC_PEER[10]”.
All received data will be checked if an array entry with the same sender IP already exist. If the senders IP is already availible in this array, following data will update those in this index.
This way, for every remote IP there is only one index created. The receive array is stored in the “Global_Variables”.

The following picture shows the array of struct with already one created index as the PM5675 already had sent data to the PM583 V2 PLC.



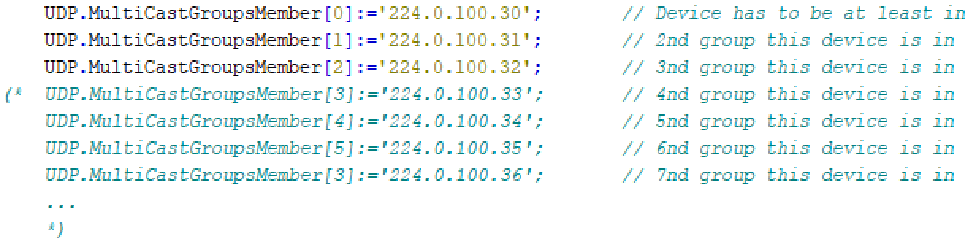
```
⊟ARR_REC_PEER
    ⊟ARR_REC_PEER[0]
        .sUDP_IP_ADR = '192.168.3.130'
        .UDP_Port = 3003
        ⊟.UDP_Data
            .UDP_Data[0] = 209
            .UDP_Data[1] = 2
            .UDP_Data[2] = 3
            .UDP_Data[3] = 4
            .UDP_Data[4] = 5
            .UDP_Data[5] = 6
            .UDP_Data[6] = 7
            .UDP_Data[7] = 8
            .UDP_Data[8] = 9
            .UDP_Data[9] = 0
        .IP_Peer_No = 1
        .UDP_LEV_DS = 0
        .UDP_OK = 897
        .UDP_NOK = 0
        .UDP_ERNO = 0
    ⊞ARR_REC_PEER[1]
    ⊞ARR_REC_PEER[2]
    ⊞ARR_REC_PEER[3]
    ⊞ARR_REC_PEER[4]
    ⊞ARR_REC_PEER[5]
    ⊞ARR_REC_PEER[6]
    ⊞ARR_REC_PEER[7]
    ⊞ARR_REC_PEER[8]
    ⊞ARR_REC_PEER[9]
REC_PEER_NR = 10
```

5.4 Usecase 3: Multicast from PM5675 to PM5052 eCo

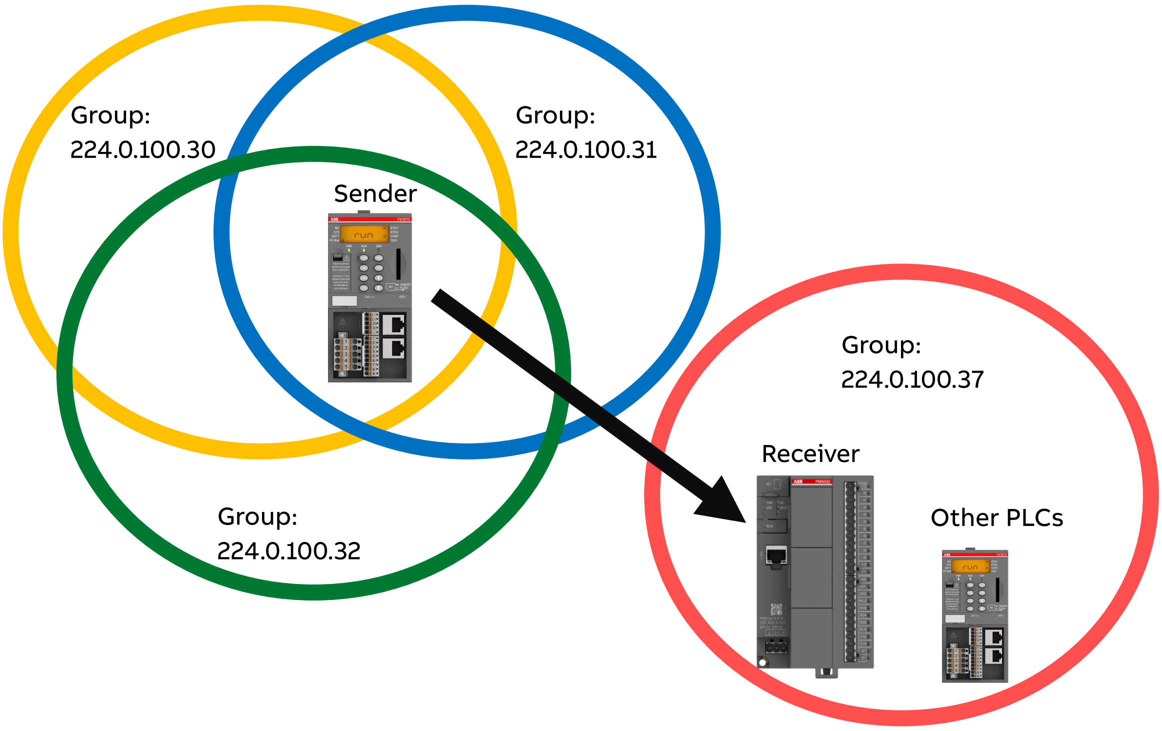
To use multicast functionality, select “Multicast” in variable “SendMode” when online. To switch to another send mode, set “ProgramEnable” to FALSE first. Then select desired mode and set TRUE afterwards this will reset the UDPWrapper. “Unicast” is selected per default.

> Note:
>
> The multicast group address looks like a normal IP address, but it is not an IP address. It only has a similar format. Group IP addresses can be from 224.0. 0.0 to 239.255. 255.255. As there are also other group addresses reserved, it is important to stay in this IP address range.

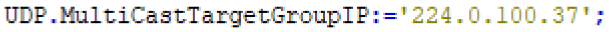
For better demonstration, following is a schematical setup which differs slightly from the setup in this application example. In the schematical setup, the sending PLC is member of three groups and sending data to one target group including one (or more) PLC’s. For this “MultiCastGroupsMember” array should be filled as followed:



```
    UDP.MultiCastGroupsMember[0]:='224.0.100.30';      // Device has to be at least in
    UDP.MultiCastGroupsMember[1]:='224.0.100.31';      // 2nd group this device is in
    UDP.MultiCastGroupsMember[2]:='224.0.100.32';      // 3nd group this device is in
(*  UDP.MultiCastGroupsMember[3]:='224.0.100.33';      // 4nd group this device is in
    UDP.MultiCastGroupsMember[4]:='224.0.100.34';      // 5nd group this device is in
    UDP.MultiCastGroupsMember[5]:='224.0.100.35';      // 6nd group this device is in
    UDP.MultiCastGroupsMember[3]:='224.0.100.36';      // 7nd group this device is in
    ...
    *)
```



The multicast target is group 224.0.100.32, which should be defined in the UDPWrapper.MultiCastTargetGroupIP.



```
UDP.MultiCastTargetGroupIP:='224.0.100.37';
```

This way. The receiver PLC as well as all other PLCs in this group will receive this multicast message.

The sender in the the application example uses only one group and its target group consits of only one eCo PLC.

5.5 Usecase 4: Broadcast from PM5675 to PM5052 eCo

To use broadcast functionality, select “Broadcast” in variable “SendMode” when online. To switch to another send mode, set “ProgramEnable” to FALSE first. Then select desired mode and set TRUE afterwards. This will reset the UDPWrapper. “

> Note:
>
> When sending broadcast messages, all devices in one network can receive this messages. A switch or router has to be configured correctly to forward broadcast messages as this is not enabled by default to prevent sending data to too much devices.

In this example, PM5675 will send a broadcast message every second, triggered by “Timer” Function block. All devices within the same network will receive this message. Important to know is, that the broadcast message also has a destination port. This one has to match with the target deviec. Here, PM5052 is also in the same network and will therefore receive the message.

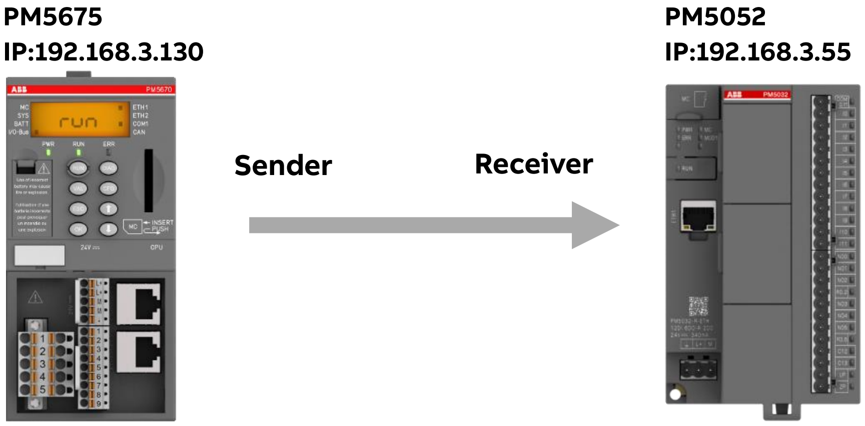


ABB AG

Contact:
https://access.motion.abb.com/contact/contact

Homepage:
www.abb.com/plc