Modbus Training

What is Modbus ?

- An open data communication protocol
- Published by Modicon
- http://www.modicon.com
- Open structure
- Flexible
- Widely known
- Supplied by many SCADA and HMI software
- 2 serial transmission modes:
  - ASCII ➔ 10 bits
  - RTU (Binary) ➔ 11 bits
- Communication interface
  - RS-232/485
  - Ethernet (TCP/IP)

- Modbus Organization (http://www.modbus.org/default.htm)

Types of Modbus:

- A leading industrial open control protocol.
- Several different types, depending upon the transported media
- Modbus RTU - Original Modbus used over RS-232 and RS-485
- Modbus ASCII - Similar to Modbus RTU, data is in ASCII instead of raw binary. This version is mainly used over radio links.
- Modbus/TCP - Used over Ethernet - similar to Modbus RTU, but uses the Ethernet check-sum rather than the RTU check sum.
- Modbus over Ethernet - This is a vague term used by some vendors, but is not an officially recognised name. It is often used to refer to tunnelling Modbus RTU over Ethernet between two points using special hardware. N*ot* part of the Modbus standard
- Modbus/UDP - similar to Modbus/TCP but uses UDP Ethernet sockets instead of TCP sockets. *not* part of the Modbus standard.
- Modbus+ - This is a proprietary protocol doesn't follow the Modbus communications standard. This is rarely encountered.

Modbus Types of communication

- Modbus serial
  1) Modbus on RS232(EIA/TIA-232)
  2) Modbus on RS422
  3) Modbus on RS485(EIA/TIA-485)
  4) Transfer method ：RTU/ASCII

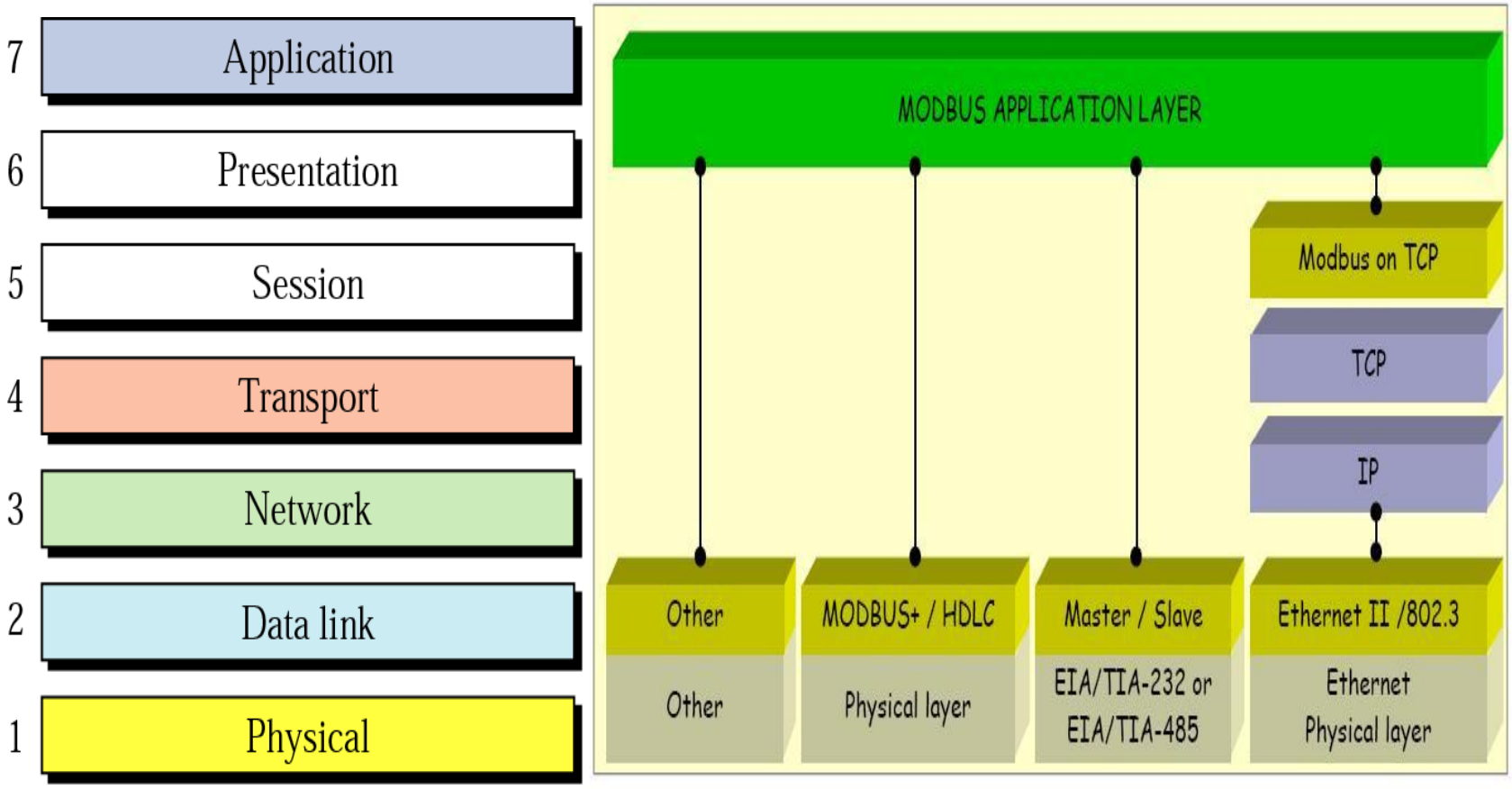
     RTU(Remote Terminal Unit)

     ASCII

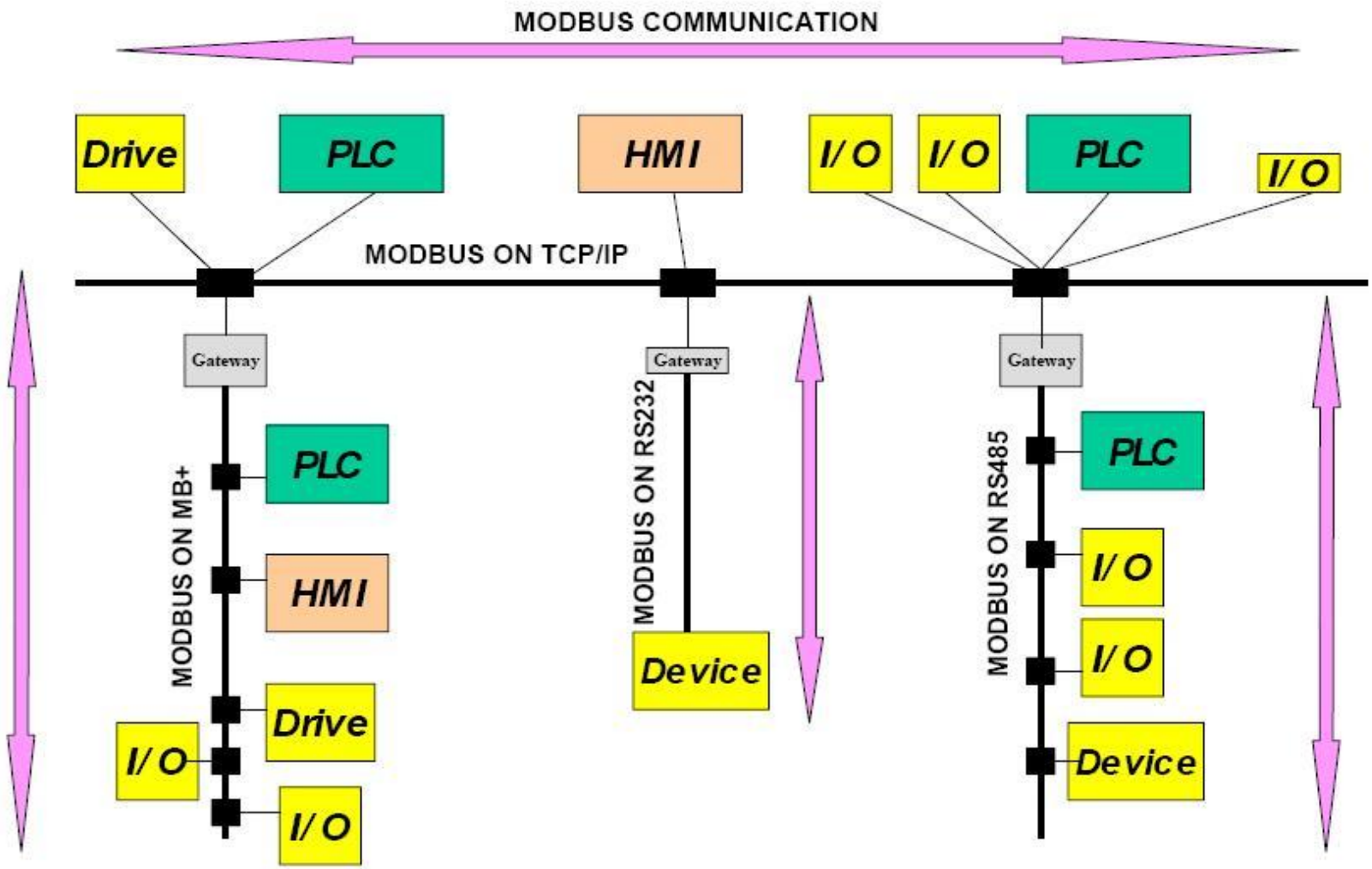
     (American Standard Code For Information Interchange)

- Modbus plus
- Modbus TCP/IP

Modbus Communication Stack



Modbus network Architecture



Modbus protocol & system application

Modbus serial

Modbus Serial Features I

1. Master-Slave Protocol

   -Master: At the same time, only one can be connected only

   -Slave ： Up to 247 can be connected to

2. Master Request 2 modes

   -Unicast mode

   Master: request(Query)

   Slave ：reply(Response)

   Address: 1 ~ 247

   -Broadcast mode

   Master sends a Request to all Slaves

   Address: 0

3. Address : 248 ~ 256(reserved)

Modbus Serial Features II

4. Communication speed

   1200, 2400, 4800, 9600, 19200 bps, 56Kbps, 115.2kbps

5. Max communication distance: 1000 m

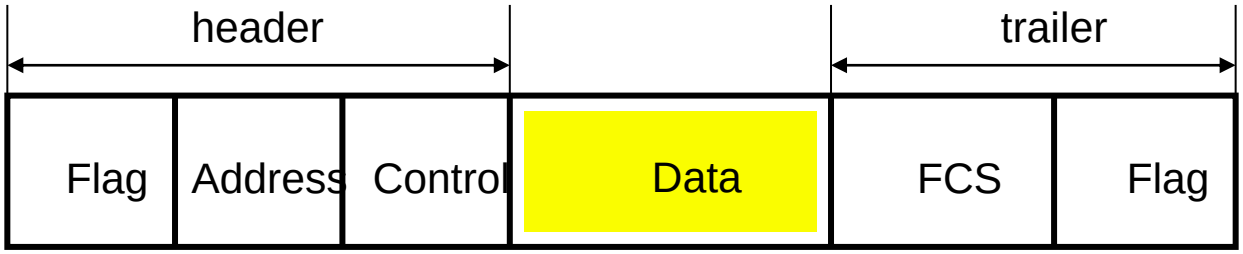
6. Termination: 150 Ohms / 0.5W

7.1979 years developed by Modicon

8. Token passing by way communication

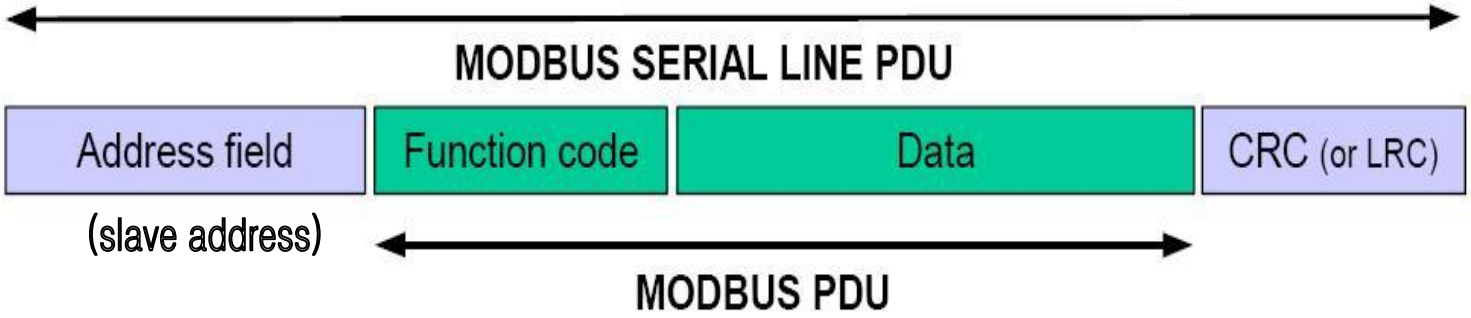
General communication frame

- **(header) :** in front of the data field
  - **flag, address, control field**
- **(trailer) :** located behind data field
  - **FCS and Flag**
- **(frame check sequence : FCS)**



Modbus Serial

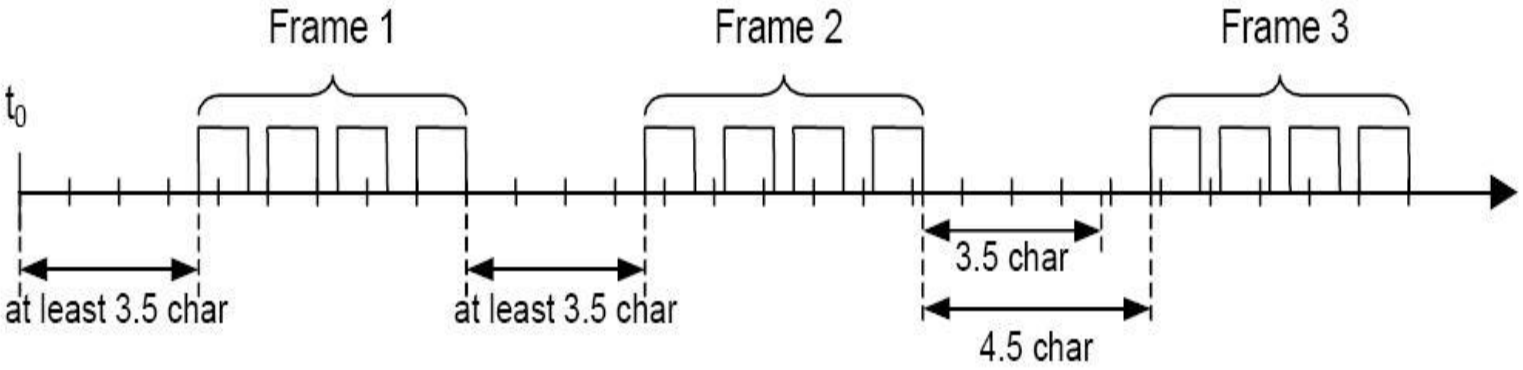
Modbus Frame



- PDU : Protocol Data Unit
- Address Field : Slaves Address
- Function Code : Indicating the command code is performed
- CRC/LRC : Frame Error Check

Modbus Serial

RTU(Remote Terminal Unit) Mode



MODBUS message

Start	Address	Function	Data	CRC Check	End
≥ 3.5 char	8 bits	8 bits	N x 8 bits	16 bits	≥ 3.5 char

Modbus Serial

ASCII Mode(Frame inside the characters that apply)

Start	Address	Function	Data	LRC	End
1 char :	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

FRAME ERROR CHECKING

1. CRC (Cyclical Redundancy Checking) => RTU
2. LRC (Longitudinal Redundancy Checking)=>ASCII

Modbus protocol frame element

Primary tables	Object type	Type of access	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

Function code Definition

Public Function Code Definition

				Function Codes			
				code	*Sub code*	*(hex)*	*page*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	13
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	11
			Write Single Coil	**05**		**05**	18
			Write Multiple Coils	**15**		**0F**	30
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	16
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	15
			Write Single Register	**06**		**06**	19
			Write Multiple Registers	**16**		**10**	32
			Read/Write Multiple Registers	**23**		**17**	40
			Mask Write Register	**22**		**16**	39
			Read FIFO queue	**24**		**18**	43
	File record access		Read File record	**20**	**6**	**14**	42
			Write File record	**21**	**6**	**15**	44
Diagnostics			Read Exception status	**07**		**07**	20
			Diagnostic	**08**	**00-18**		21
			Get Com event counter	**11**		**OB**	26
			Get Com Event Log	**12**		**0C**	28
			Report Slave ID	**17**		**11**	34
			Read device Identification	**43**	**14**	**2B**	46
Other			Encapsulated Interface Transport	**43**		**2B**	44

Modbus Frame

(0x02) Read Discrete inputs

Request

Function code	1 Byte	**0x01**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of coils	2 Bytes	1 to 2000 (0x7D0)

Response

Function code	1 Byte	**0x01**
Byte count	1 Byte	**N***
Coil Status	**n** Byte	n = N or N+1

***N** = Quantity of Outputs / 8, if the remainder is different of 0 ⇒ N = N+1

Error

Function code	1 Byte	**Function code + 0x80**
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to read discrete outputs 20–38:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**01**	Function	**01**
Starting Address Hi	**00**	Byte Count	**03**
Starting Address Lo	**13**	Outputs status 27-20	**CD**
Quantity of Outputs Hi	**00**	Outputs status 35-28	**6B**
Quantity of Outputs Lo	**13**	Outputs status 38-36	**05**

Modbus Frame

(0x03) Read Holding Registers

Request

Function code	1 Byte	0x03
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	0x03
Byte count	1 Byte	2 x N*
Register value	N* x 2 Bytes	

*N = Quantity of Registers

Error

Error code	1 Byte	0x83
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to read registers 108 – 110:

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	03	Function	03
Starting Address Hi	00	Byte Count	06
Starting Address Lo	6B	Register value Hi (108)	02
No. of Registers Hi	00	Register value Lo (108)	2B
No. of Registers Lo	03	Register value Hi (109)	00
		Register value Lo (109)	00
		Register value Hi (110)	00
		Register value Lo (110)	64

Master Query

QUERY

Field Name	Example (Hex)	ASCII Characters	RTU 8–Bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function	03	0 3	0000 0011
Starting Address Hi	00	0 0	0000 0000
Starting Address Lo	6B	6 B	0110 1011
No. of Registers Hi	00	0 0	0000 0000
No. of Registers Lo	03	0 3	0000 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	17	8

Slave Response

RESPONSE

Field Name	Example (Hex)	ASCII Characters	RTU 8–Bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function	03	0 3	0000 0011
Byte Count	06	0 6	0000 0110
Data Hi	02	0 2	0000 0010
Data Lo	2B	2 B	0010 1011
Data Hi	00	0 0	0000 0000
Data Lo	00	0 0	0000 0000
Data Hi	00	0 0	0000 0000
Data Lo	63	6 3	0110 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	23	11

Modbus transaction(exception response)

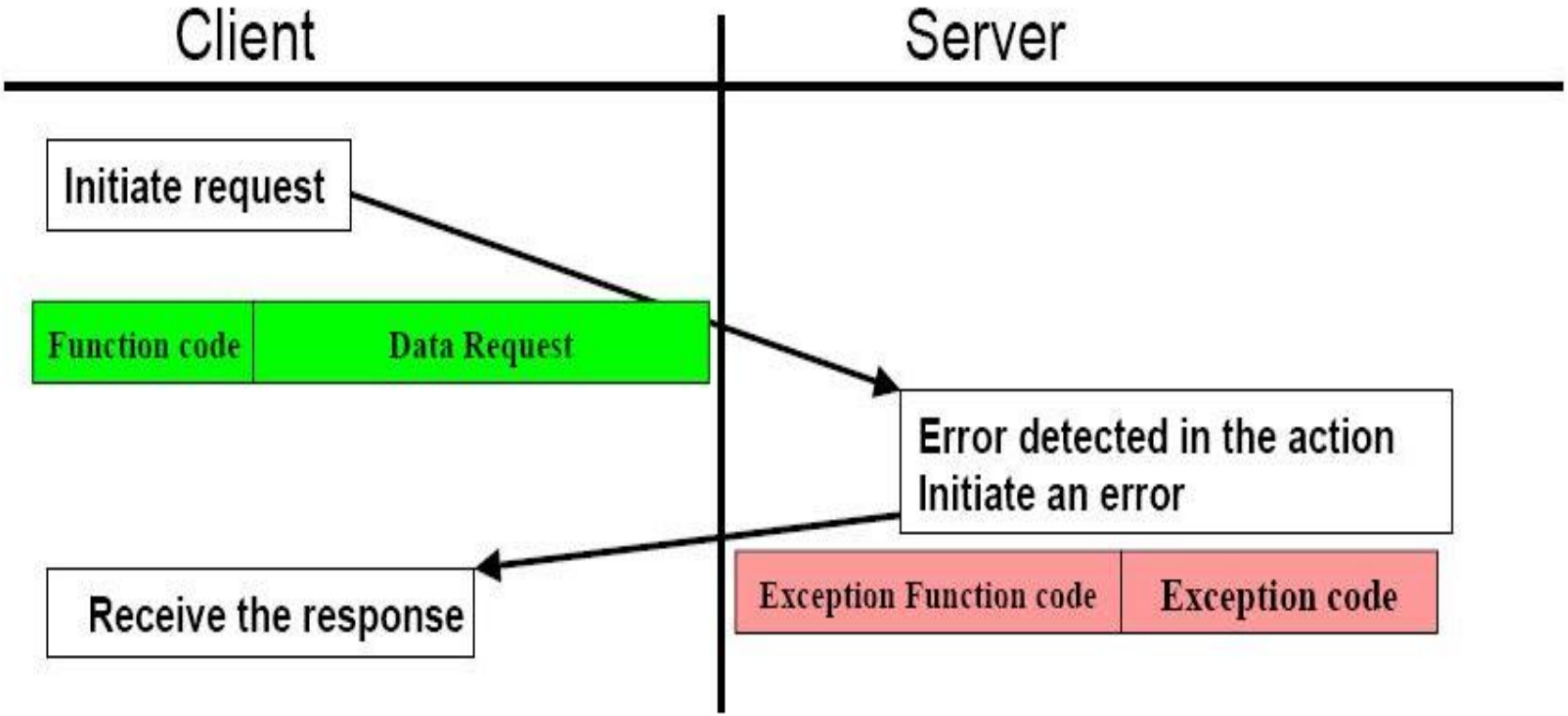


Figure 5: MODBUS transaction (exception response)

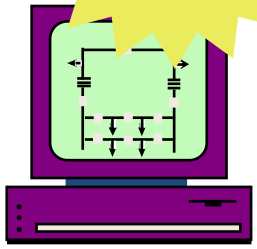
Modbus Master AutoSense Feature

Slave

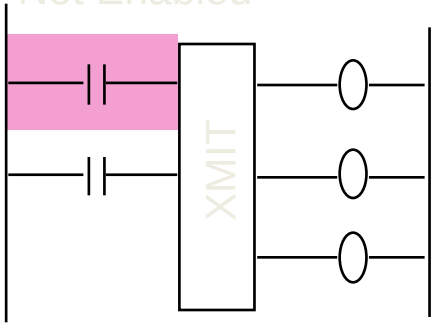
Query

Response

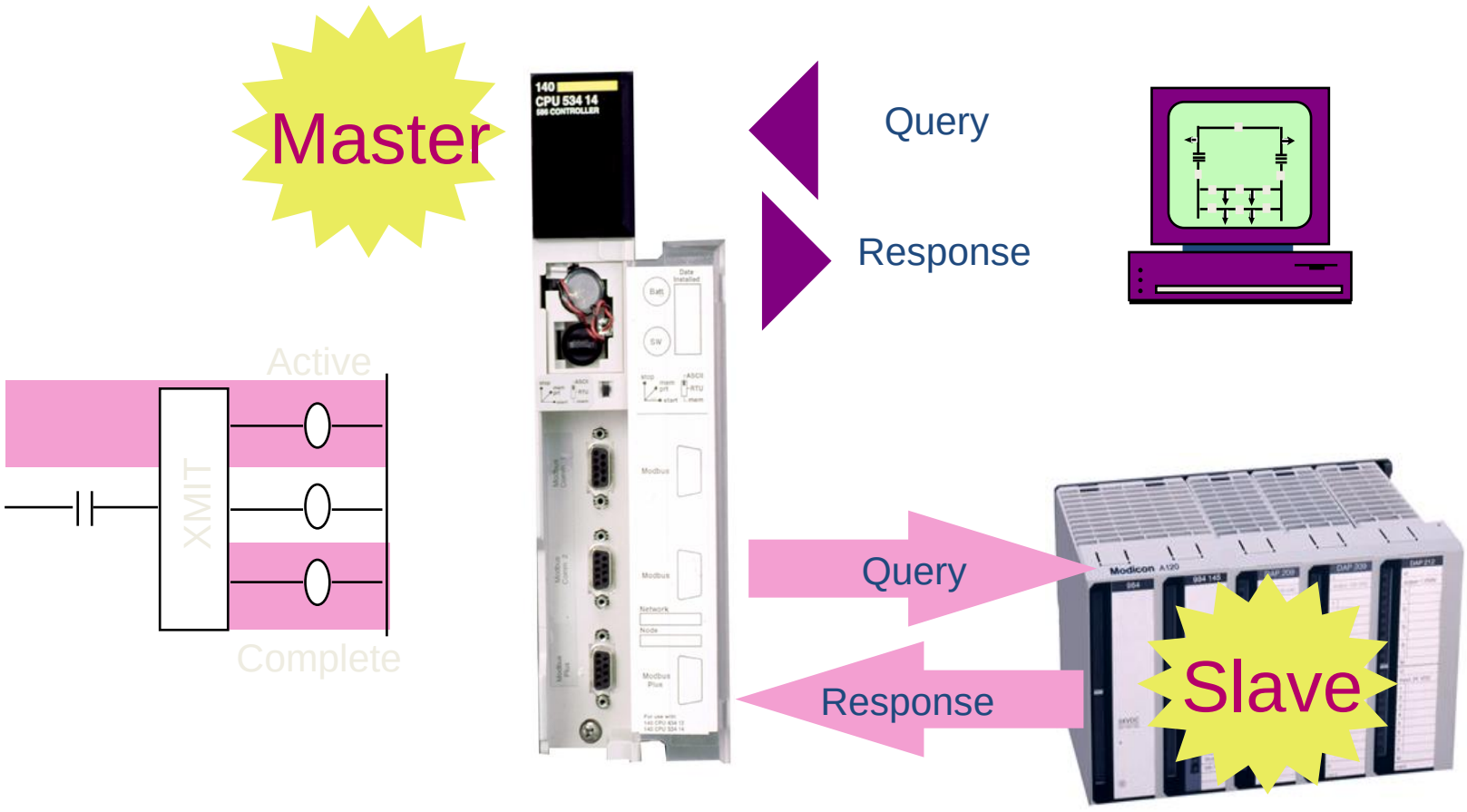
Master



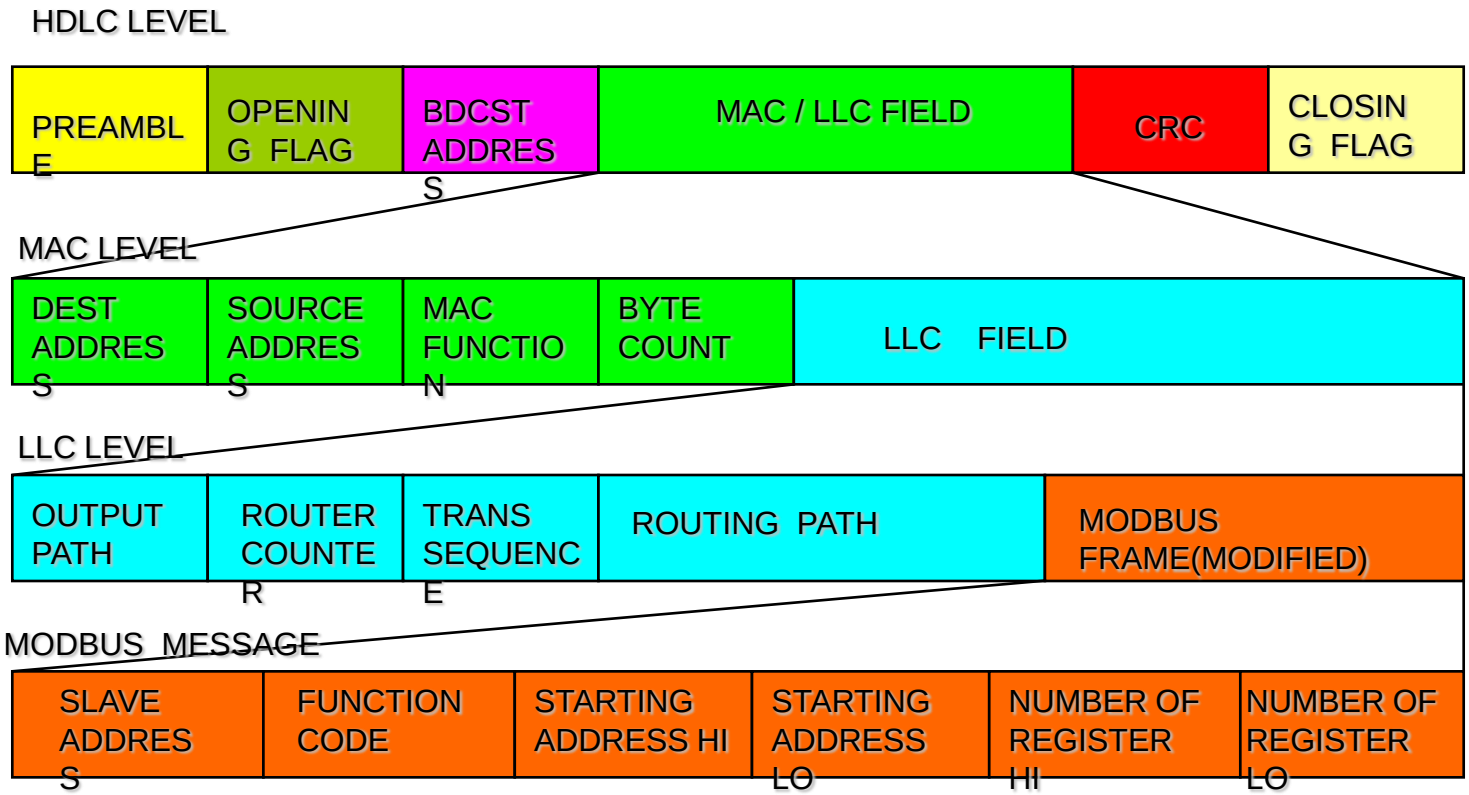
Not Enabled



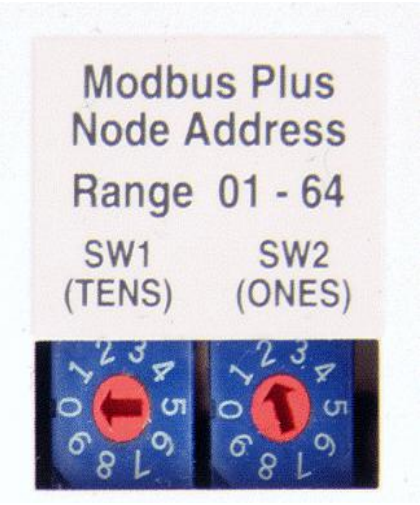
Modbus Master AutoSense Feature



Modbus Plus Frame



The flexibility of Modbus Plus



1. Data Communications
   - Modbus Application solution to enhance more
   - High-speed connectivity of the host, and improved operator system
   - Event handling of peer to peer communication
   -Controller for distributed control interlock between the easy and reliable
   - Bridges and repeaters can be configured in a flexible
   - Schneider's products and products with a variety of connectivity

2. Programming

Program to upload, download and can be Verify.

- The network connection is available in the eight programmer productivity was increased.

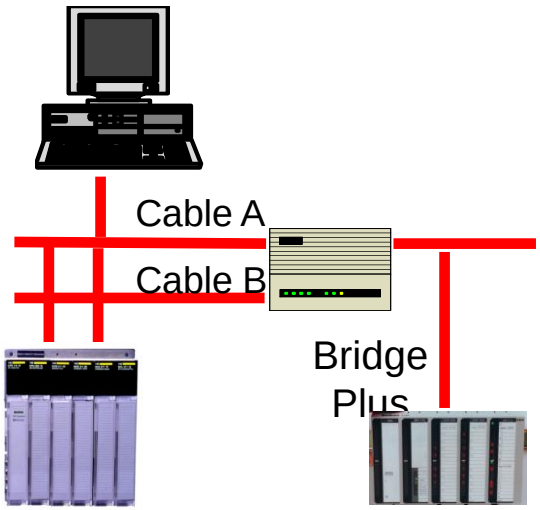
Modbus Plus System

Single network with dual cabling, all network traffic is carried over both cables simultaneously

Available on

- Standard on Quantum CPU controller
- Quantum and NOM modules
- Quantum DIO adapters
- AT and VME bus adapters
- Some PanelMates
- Bridge Multiplexer and Bridge Plus

Excellent for process and safety critical applications



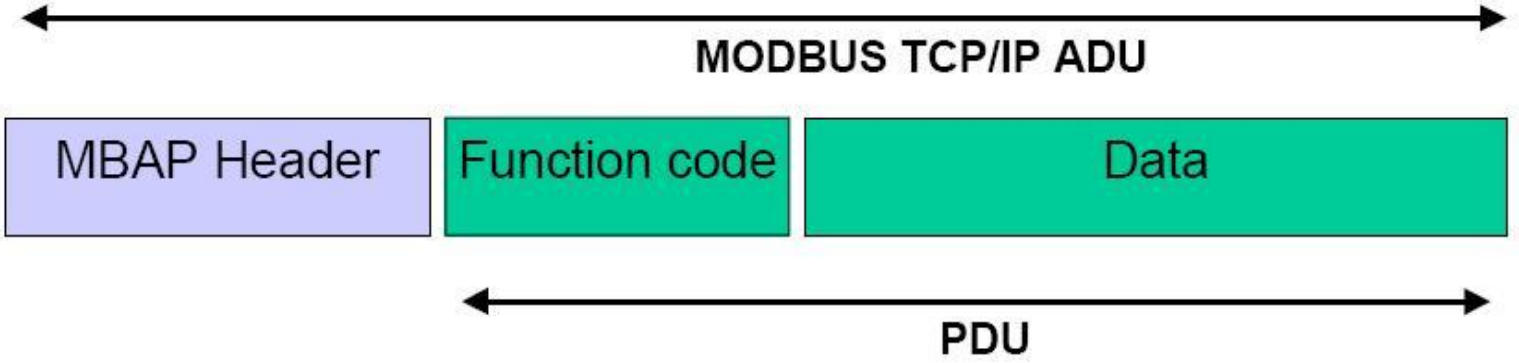
Dual cable Modbus Plus Network

Single cable Modbus Plus Network

Modbus protocol & system application

Modbus TCP/IP

Modbus TCP/IP ADU

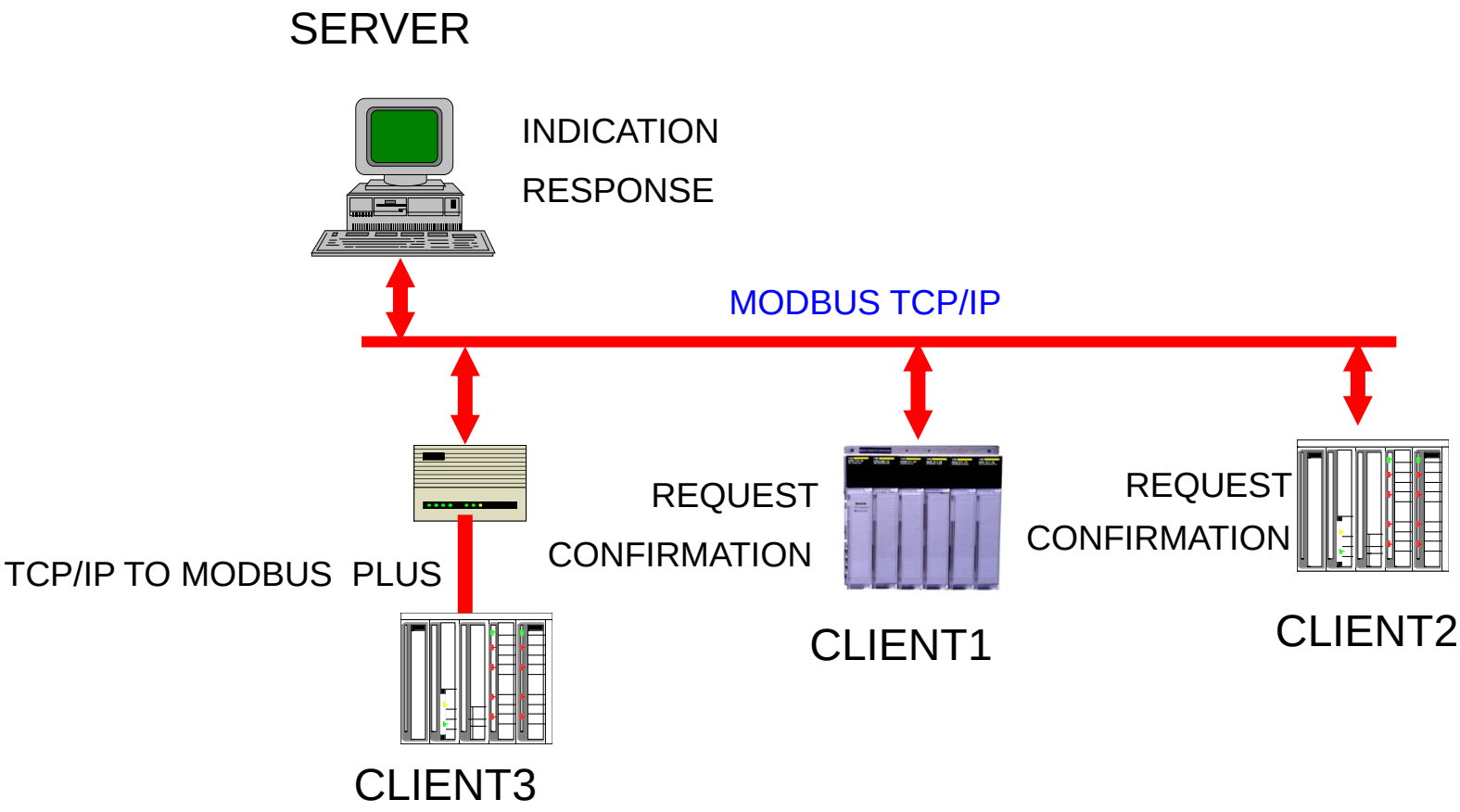


- ADU : Application Data Unit
- PDU : Protocol Data Unit
- MBAP : Modbus Application Protocol
- Function Code :What kind of a code indicating the command is performed

MBAP Header

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

SERVER/CRIENT MODEL



Communication on and off

- Communication Settings
  - Modbus message, the program performs a new communications link for exchanging data with other devices in order to Port no 502 should provide a listening socket.
  - The local port must be greater than 1024 and each one client to another are Different.
  - Connecting the client and the server exceeds the number of allowable features

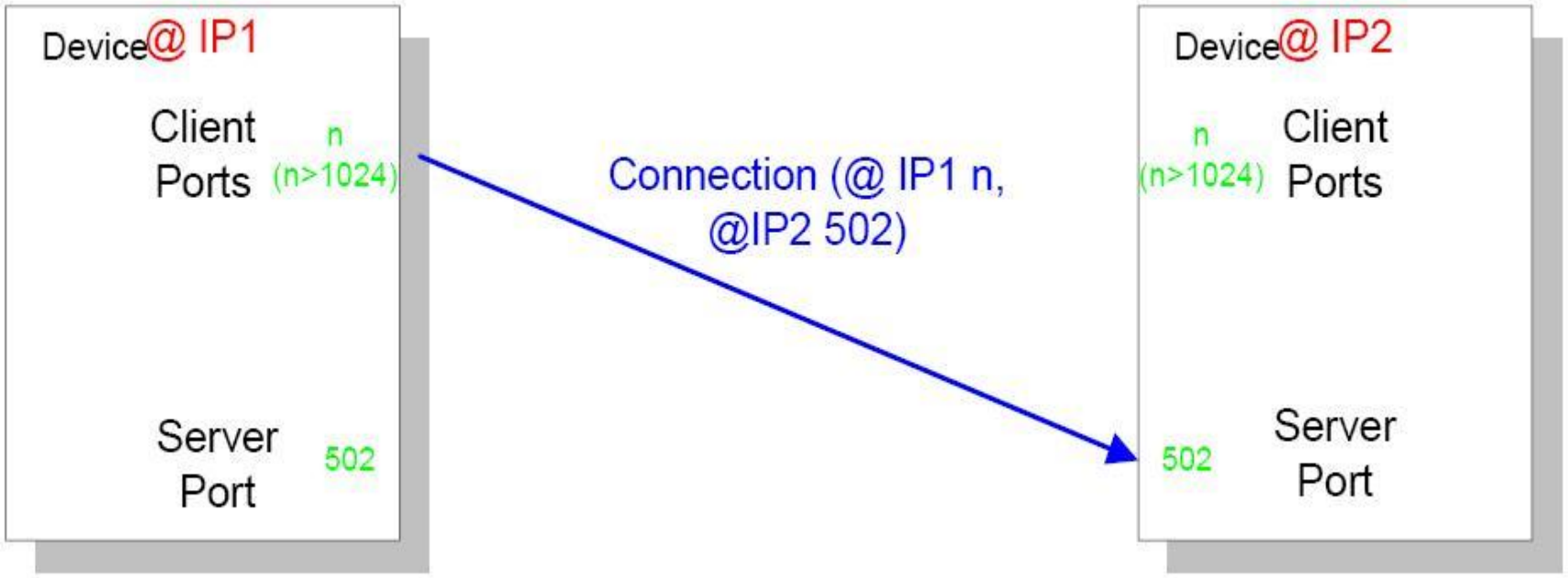
  Has not been used for a long time, it is the most closed.

Two. Communications closing (Closing)

- The communication between the client and the server, the client ends

  Was used to initiate the connection to close the connection.

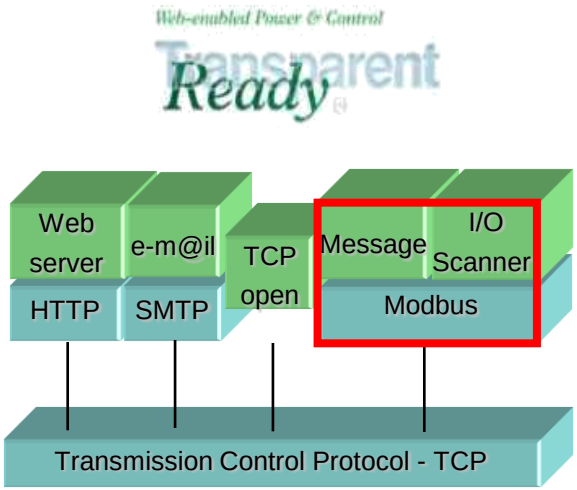
Communication on and off



Modbus TCP/IP Features

1. Slave address field of a Modbus Serial line on the MBAP Header Located in the unit identifier is replaced by a single byte.
2. Unit identifier is to support multiple independent Modbus unit Using the IP address of one of the bridges, routers, and devices such as gateways are used to communicate.
3. Modbus requests and responses are all recipients of the message Ended been designed so that they can be confirmed.
4. To be performed on Modbus TCP when the length of the accompanying information Recipient of the message is unknown, so the range is performed at MBAP.
5. CRC-32 error check code is used.
6. All Modbus / TCP ADU are a registed port 502 over TCP Sent.

Modbus TCP/IP mFeatures



Modbus TCP/IP function codes		dec	hexa
Bits access	Read of n input bits	02	02
	Read of n output bits	01	01
	Exceptional read status	07	07
	Write 1 output bit	05	05
	Write of n output bits	15	0F
	Read of 1 input word	04	04
	Read of n input words	03	03
	Write 1 output word	06	06
	Write of n output words	16	10
	Read device ID	43/14	2B/0E
	Acces CANopen interface	43/13	2B/0D

- Modbus TCP
  Read and write data
  Update I / O

Benefit: Modbus Device connectivity with the other is easy. Host computers and OPC servers directly, without the PLC to communicate with other devices can send and receive. .
Transparent link to Modbus Serial Devices

Results: The cost is reduced. (Special equipment is not needed).
Simply build a communication system.

Modbus TCP/IP Features

Modbus-IDA
the architecture for distributed automation

" Modbus Accepted as Chinese Standard"

" Modbus TCP Accepted as IEC Publicly Available Specification"

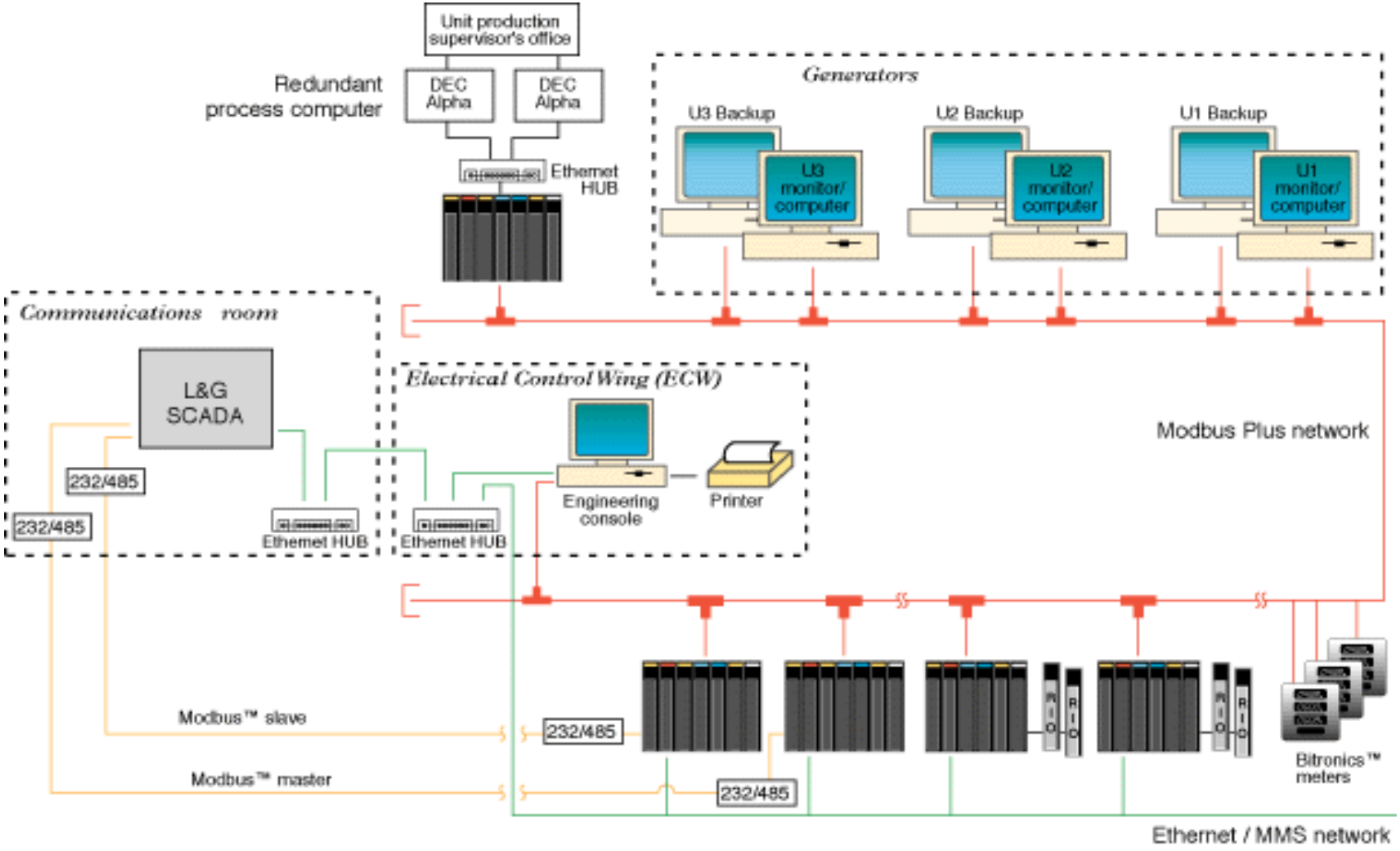
" ARC Advisory Study Places Modbus TCP in the Lead"

- MODBUS ® is the world's most widely used industrial protocol. Modbus Ethernet (Modbus TCP) is the best of the Industrial Ethernet. By Modbus-IDA is a free, open protocols.
  Communication system, which is easy to use, simply
  Independent media; Ethernet, RS-232/485 serial links, wireless, fiber optic, radio, cellular, etc.
  There are hundreds of MODBUS device.
  Modbus is a industrial protocol (IP) standard to comply with. (Port 502)

Modbus protocol & system application

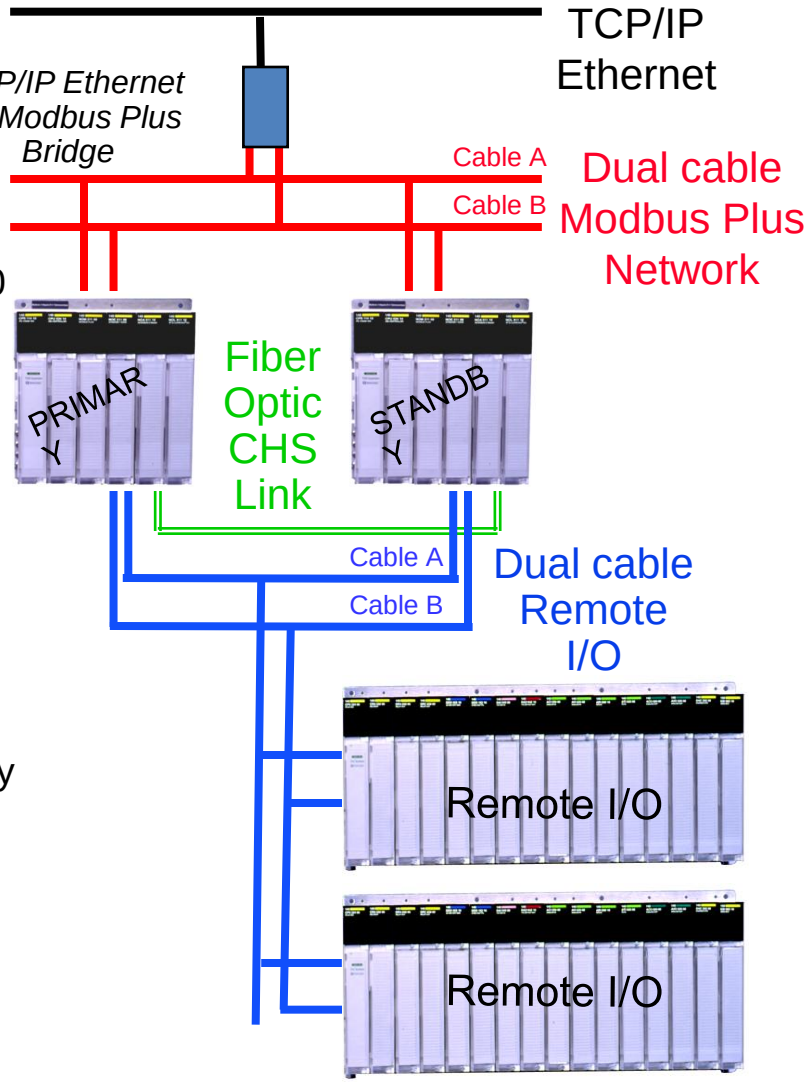
SYSTEM APPLICATION

Modbus Plus network

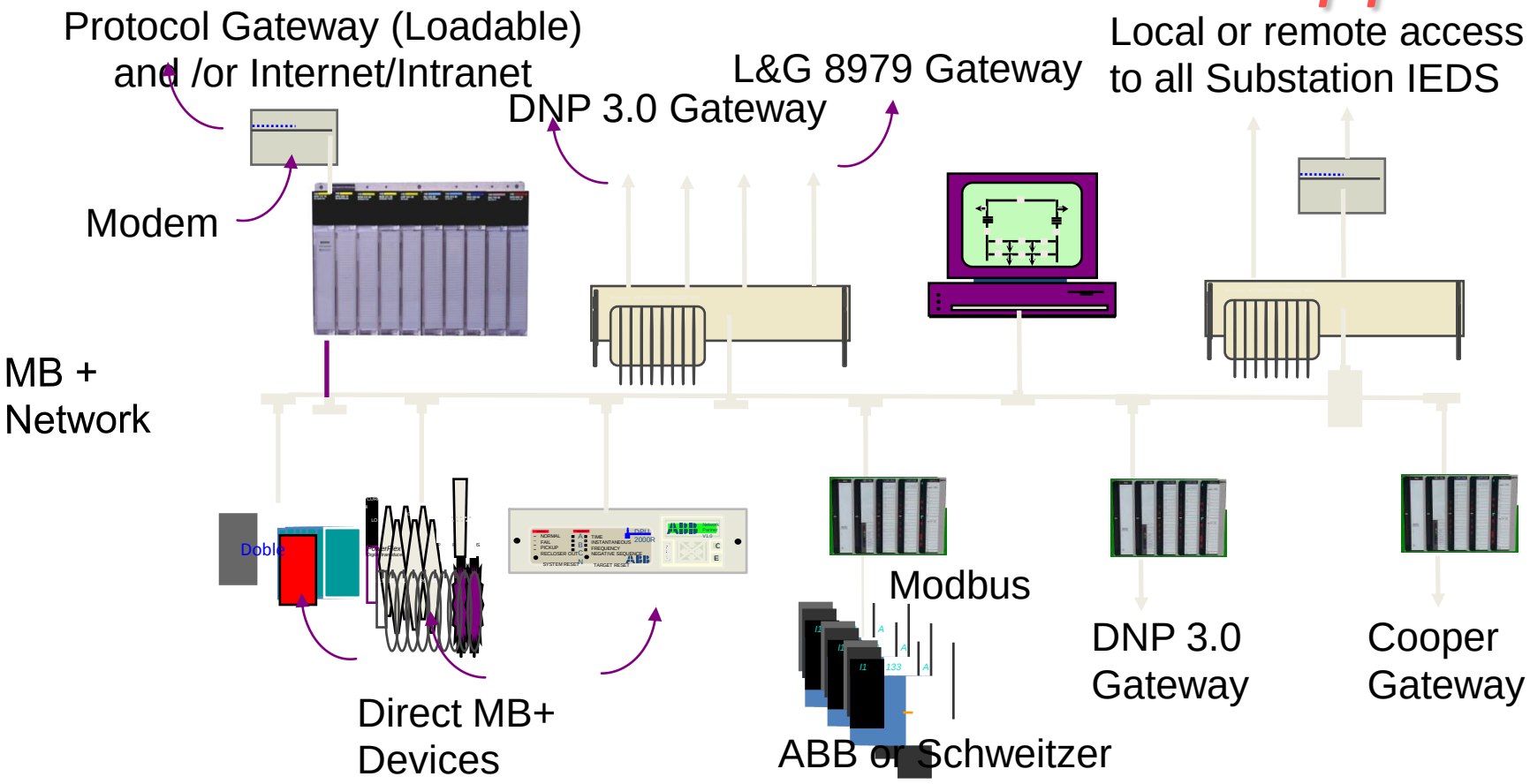


Quantum Hot Standby, the ultimate in high availability

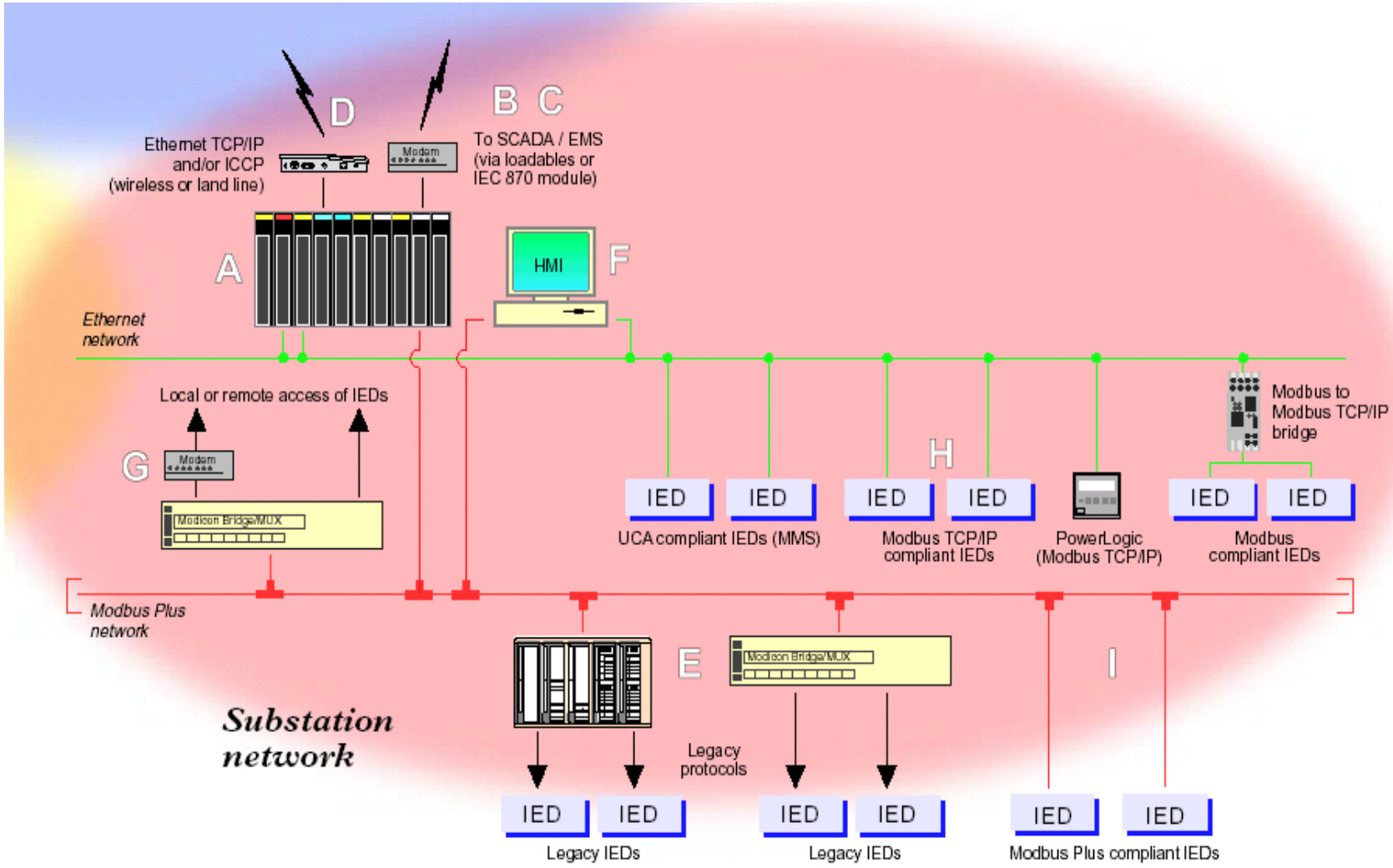
- Redundant
  - CPUs140cpu671 60
  - Remote I/O processors
  - Hot standby processors
- Redundant cabling
  - Modbus Plus
  - Remote I/O
- Redundant power supplies
  - Primary and standby controllers
  - Remote I/O drops
- Fiber optic CHS processor link



Electrical Utility Infrastructure
Complete Substation Data Hub
Multi - vendor communications support



System Communication



Modbus TCP/IP

- Ethernet TCP/IP and Web technologies
- Modbus, industrial & Internet standard
- Openness and partnership

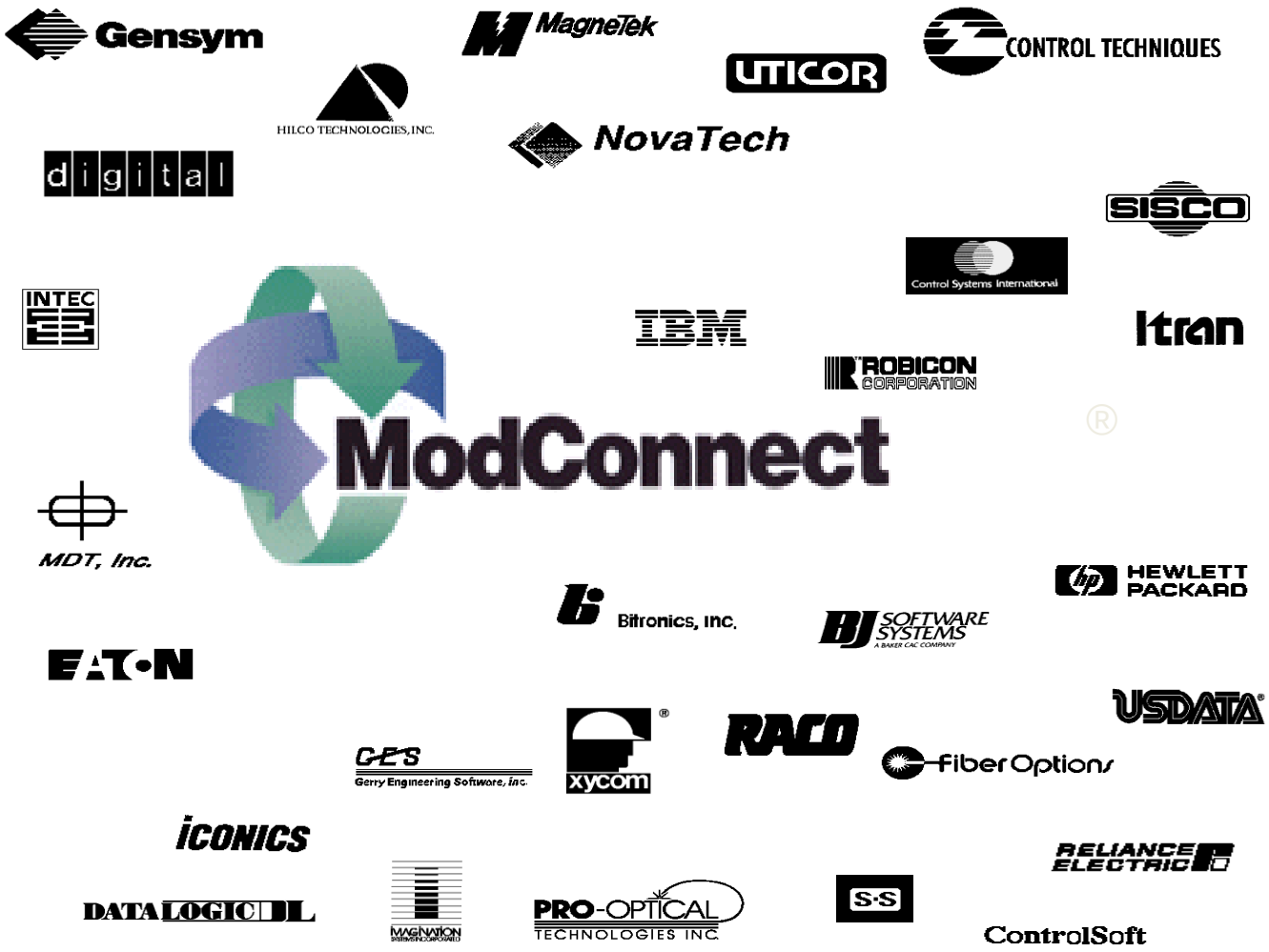
Electrical Distribution

- Reducing energy costs
- Increasing energy availability and quality
- Optimizing electrical equipment utilization

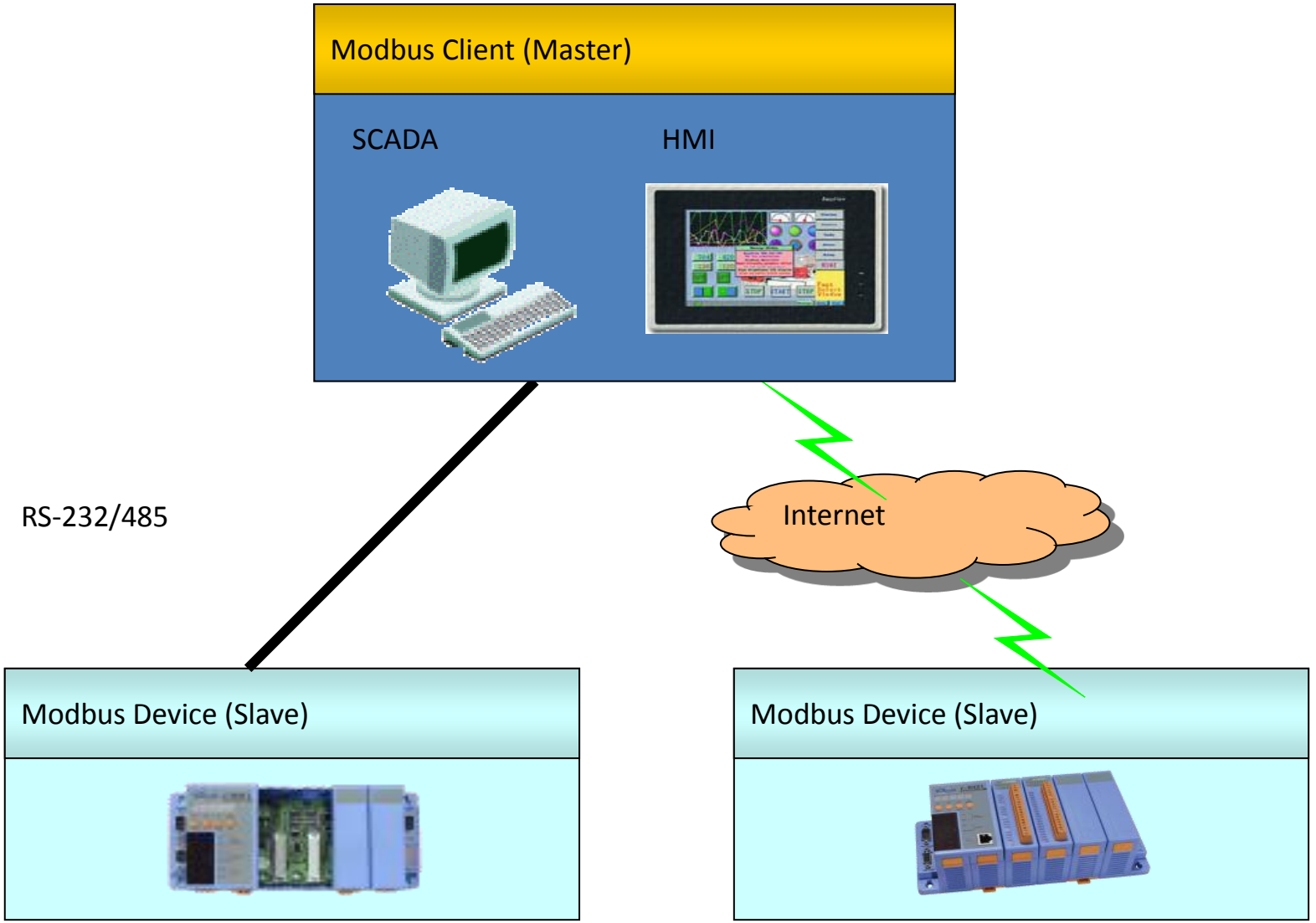
Automation & Control

- Ingenuity of collaboration
- Openness of Ethernet TCP/IP universal network
- Simple HMI, with Web technologies

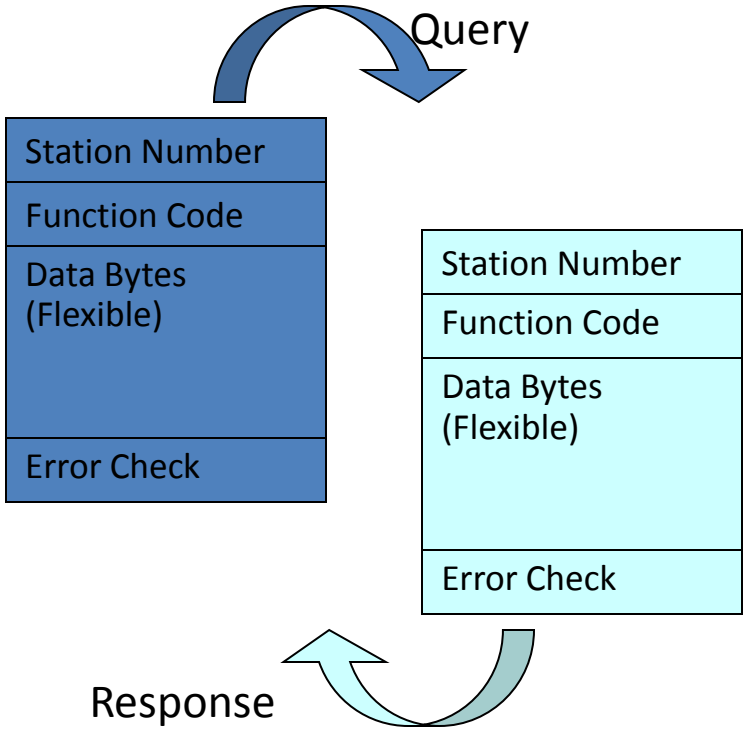
Third party connectivity



Application Structure (general)



Query-Response Cycle



Communication
Client/Server or Master/Slave:

- Modbus uses a *client/server* type protocol. Sometimes referred to as *master/slave*. A "master" = "client", while a "slave" = a "server".
- The master/slave terms originated in industry, while the client/server terms originated in the computer industry. Typically, the client is the PLC or controller, while the server is a field device such as a valve bank or sensor block.
- A "client" sends a *request* to a "server". The server decodes the request and sends back a *response* with the requested data or an acknowledgement.
- Eg. when you use a web browser to view a web page on the internet, your web browser sends a “page request” to the web server. The web server decodes the request and sends back a web page as a “response”. Your e-mail client program fetches your e-mail in the same way from a mail server.

Unit ID & Message ID

- **Unit ID:**
- A Modbus message includes the *unit ID*. A unit ID - from 0 and 255
  - used to identify the server (slave) address in RS-232 or RS-485 networks.
  - Each server (slave) is assigned a "slave ID" number and listens for messages which contain this number in the unit ID field.
- Modbus/TCP also has the unit ID in its messages, but the Ethernet TCP/IP address is used to decide where to actually delivery the message. Many or most server devices will ignore the unit ID. However, some will use the unit ID to decide whether to forward the message out a built-in serial port. This message forwarding allows older RS-485 devices to be used on newer Ethernet networks. Support for this feature is only found in a few devices.
- **Message ID:**
- When a Modbus message sends a request, it includes a *message ID* number.
  - from 0 to 65,535. It is incremented by the client for each request (and will roll over to 0 again when it overflows).
  - This message ID is echoed back by the server. The client can use this message ID number to determine if any messages are being lost or delayed in transmission.

Hardware Classification

- Station Device: 0 ~ 255
- Digital input module
  - 1xxxx: 4 digits for hexadecimal address (0000 ~ FFFF)
  - 1xxxxx: 5 digital for decimal address (0 ~ 65535)
- Digital output module
  - 0xxxx: 4 digits for hexadecimal address (0000 ~ FFFF)
  - 0xxxxx: 5 digital for decimal address (0 ~ 65535)
- Analog input module
  - 3xxxx: 4 digits for hexadecimal address (0000 ~ FFFF)
  - 3xxxxx: 5 digital for decimal address (0 ~ 65535)
- Analog output module
  - 4xxxx: 4 digits for hexadecimal address (0000 ~ FFFF)
  - 4xxxxx: 5 digital for decimal address (0 ~ 65535)

- Begining of Address
  - From 0: VLC
  - From 1: InduSoft, iFix

Two Serial Transmission Modes

- ASCII Mode
  - Data system
    ASCII character, '0'~'9','A'~'F'
  - Bits per data unit

1 Start Bit	7 Data Bits	1 Parity Bit (Even/Odd)	1 Stop Bit

1 Start Bit	7 Data Bits	2 Stop Bit

  - Error Check Field
    Longitudinal Redundancy Check (LRC)

- RTU Mode
  - Data system
    8-bit Binary, 00~FF
  - Bits per data unit

1 Start Bit	8 Data Bits	1 Parity Bit (Even/Odd)	1 Stop Bit

1 Start Bit	8 Data Bits	2 Stop Bit

  - Error Check Field
    Cyclical Redundancy Check (CRC)

Modbus Message Packet

- ASCII Mode

Start	Station Number	Function Code	Data	Error Check	End
1 Char	2 Chars	2 Chars	n Chars	2 Chars	2 Chars
:				LRC	CR,LF

- RTU Mode

Start	Station Number	Function Code	Data	Error Check	End
3.5 Char	1 Char	1 Char	n Chars	2 Chars	3.5 Chars
Silence				CRC	Silence

- Modbus Plus network

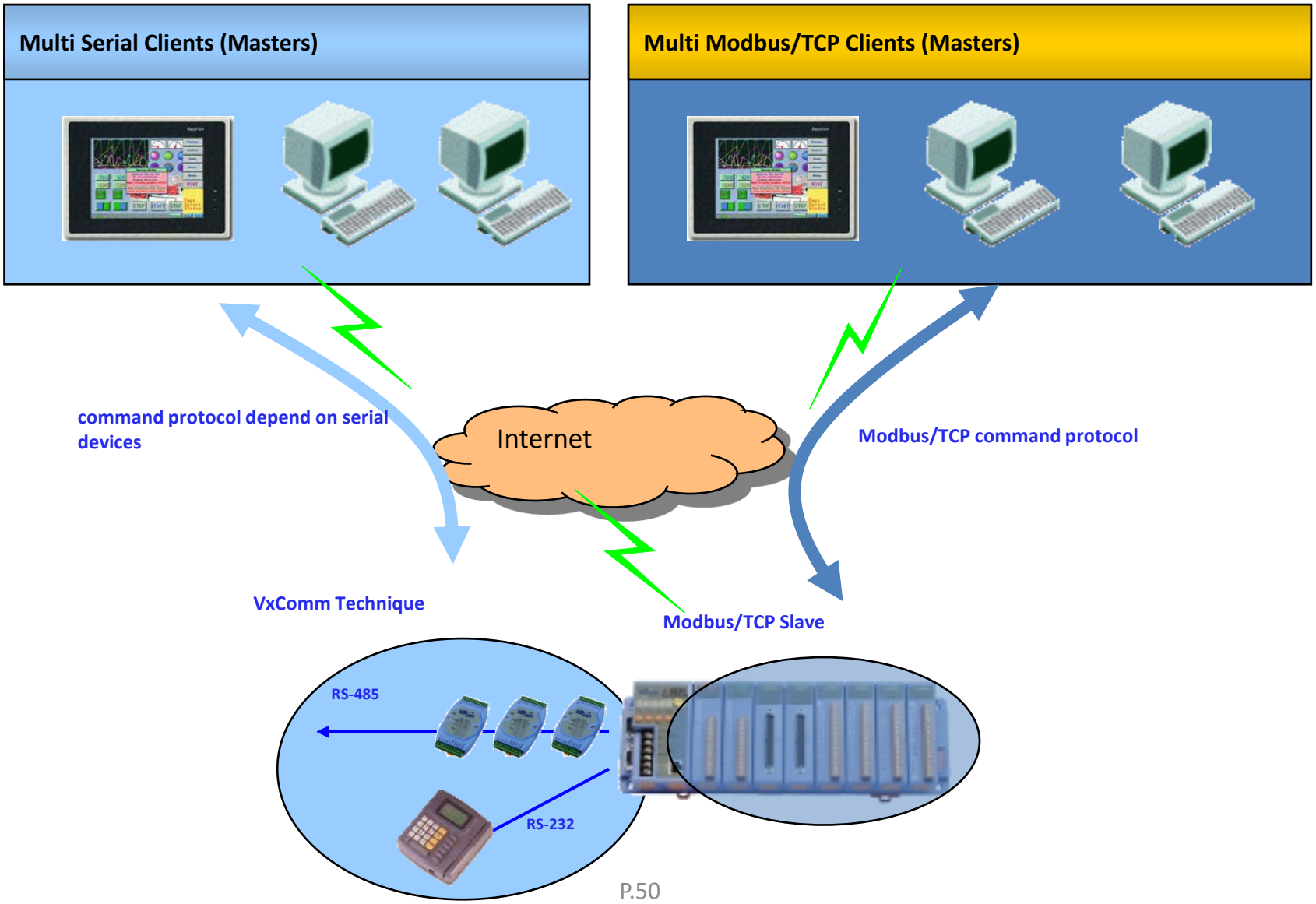
Prefixed Data	Station Number	Function Code	Data
6 x 8 Bits			

Byte 0, 1: transaction ID – usually 0
Byte 2, 3: protocol ID = 0
Byte 4, 5: number of bytes following

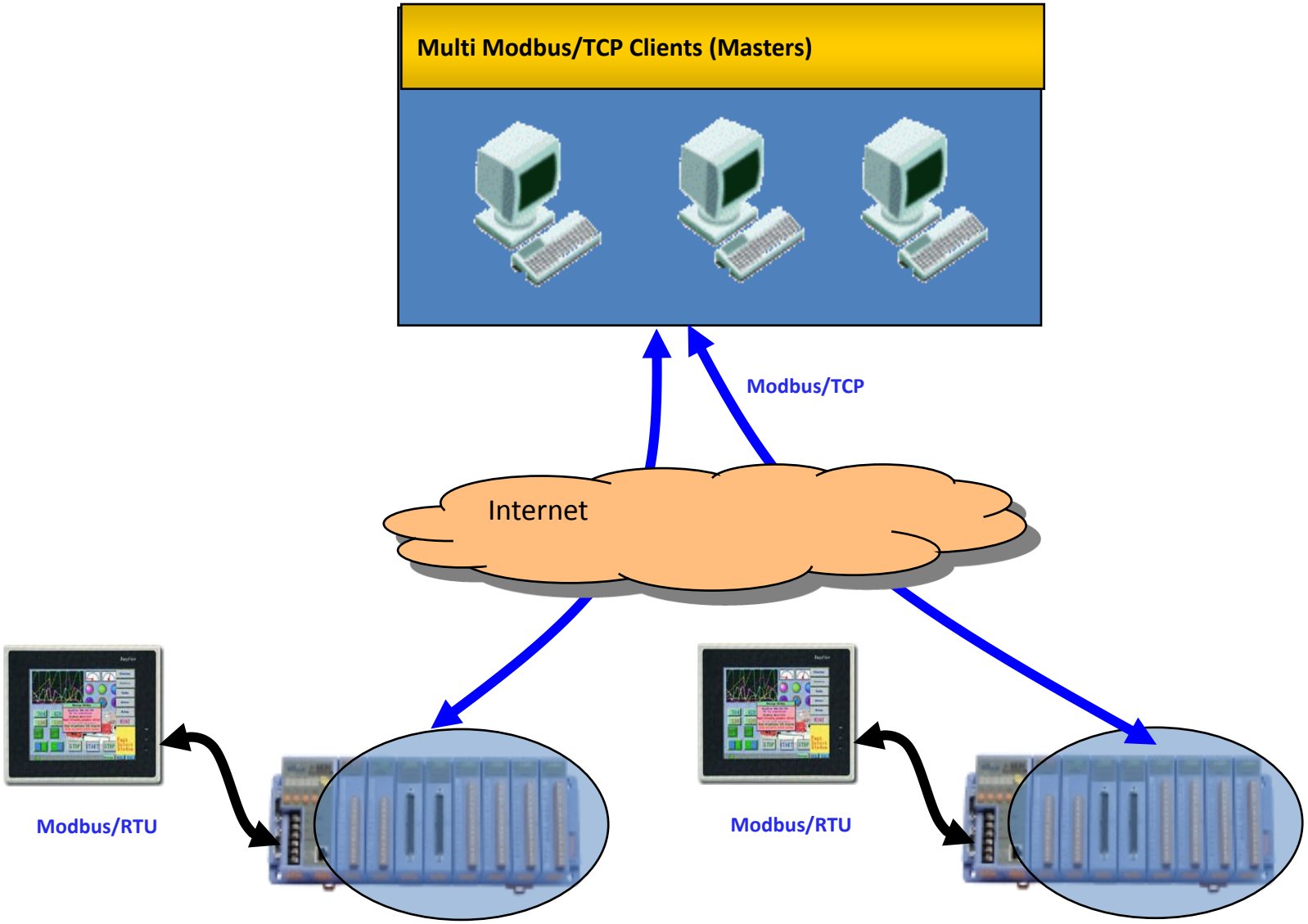
Modbus Function Code

- 01: read DOs (0xxxx)
- 02: read DIs (1xxxx)
- 03: read AOs (4xxxx)
- 04: read AIs (3xxxx)
- 05: write single DO (0xxxx)
- 06: write single AO (4xxxx)
- 15: wirte DOs (0xxxx)
- 16: write AOs (4xxxx)

System Application



System Application



8000E –MTCP Features

- Supports Modbus/TCP communication protocol to access I/Os that plug on slots
- Supports VxComm technique for every COM port of controllers
- Auto scan I/O modules

Digital Module Mapping | Analog Module Mapping | **Summary**

Slot	Module	DI (1xxxx) address	Points	DO (0xxxx) address	Points	AI (3xxxx) address	Points	AO (4xxxx) address	Points
0	I-8053	00 [00]	16	-	-	-	-	-	-
1	I-8017H	-	-	-	-	00 [00]	8	-	-
2	I-8024	-	-	-	-	-	-	00 [00]	4
3	I-8041	-	-	00 [00]	32	-	-	-	-

- Automatically range register address of I/O modules

Digital Module Mapping | Analog Module Mapping | Summary

Digital Input (1xxxx)

Address	Module	Slot	Channel
13 [0D]	I-8053	0	13
14 [0E]	I-8053	0	14
15 [0F]	I-8053	0	15
16 [10]	I-8040	2	0
17 [11]	I-8040	2	1
18 [12]	I-8040	2	2
19 [13]	I-8040	2	3
20 [14]	I-8040	2	4
21 [15]	I-8040	2	5

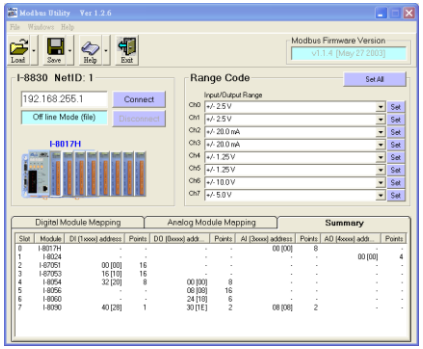
Digital Output (0xxxx)

Address	Module	Slot	Channel
00 [00]	I-8041	3	0
01 [01]	I-8041	3	1
02 [02]	I-8041	3	2
03 [03]	I-8041	3	3
04 [04]	I-8041	3	4
05 [05]	I-8041	3	5
06 [06]	I-8041	3	6
07 [07]	I-8041	3	7
08 [08]	I-8041	3	8

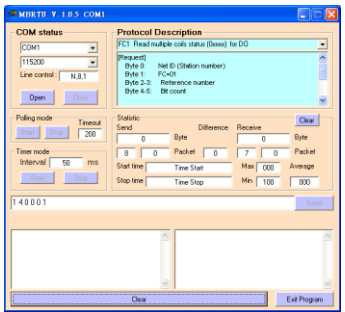
- Allows multi-client (or master) access simultaneously
- Online configuration (using Modbus Utility via Ethernet)
- Supports I-8000 and I-87000 series I/O modules
- Firmware updateable and programmable

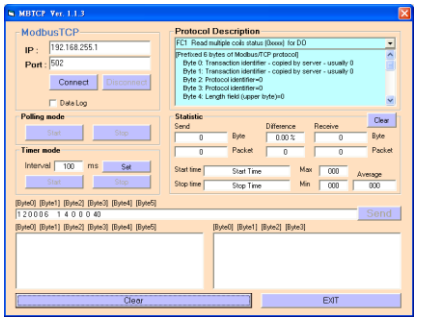
Tools

- MiniOS7 Utility (Download files and update OS image)

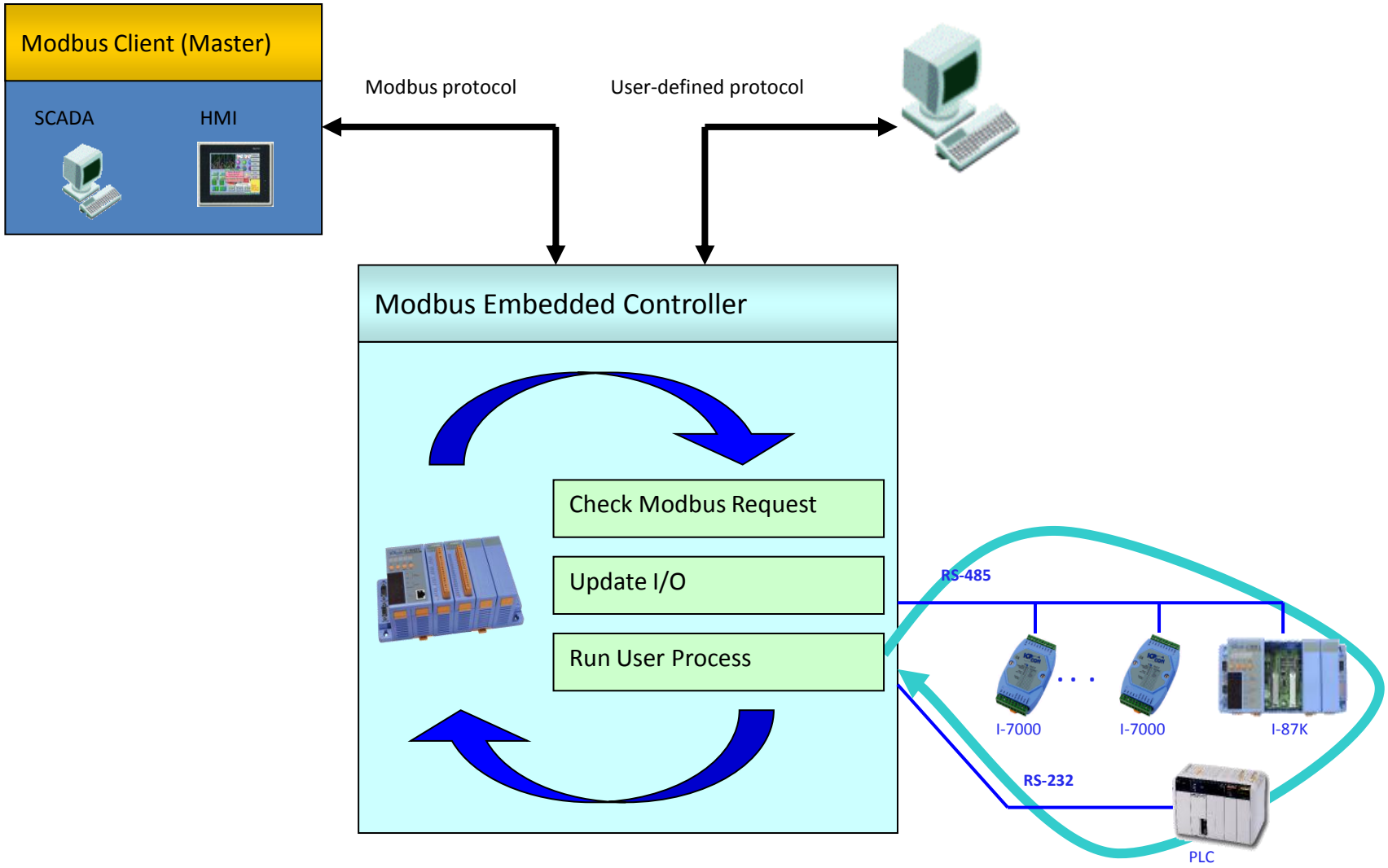


- PCDiag (Diagnostic tools)
- NAP OPC Server (Check I/O action quickly)
- MBTCP.exe (Check Modbus/TCP package details)
- MBRTU.exe (Check Modbus/RTU package details)



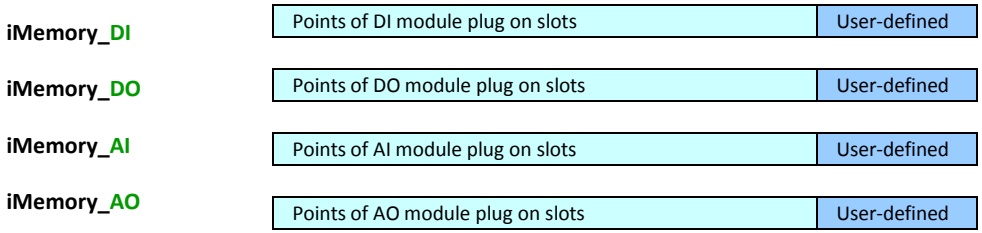


8000E-MTCP Program Block

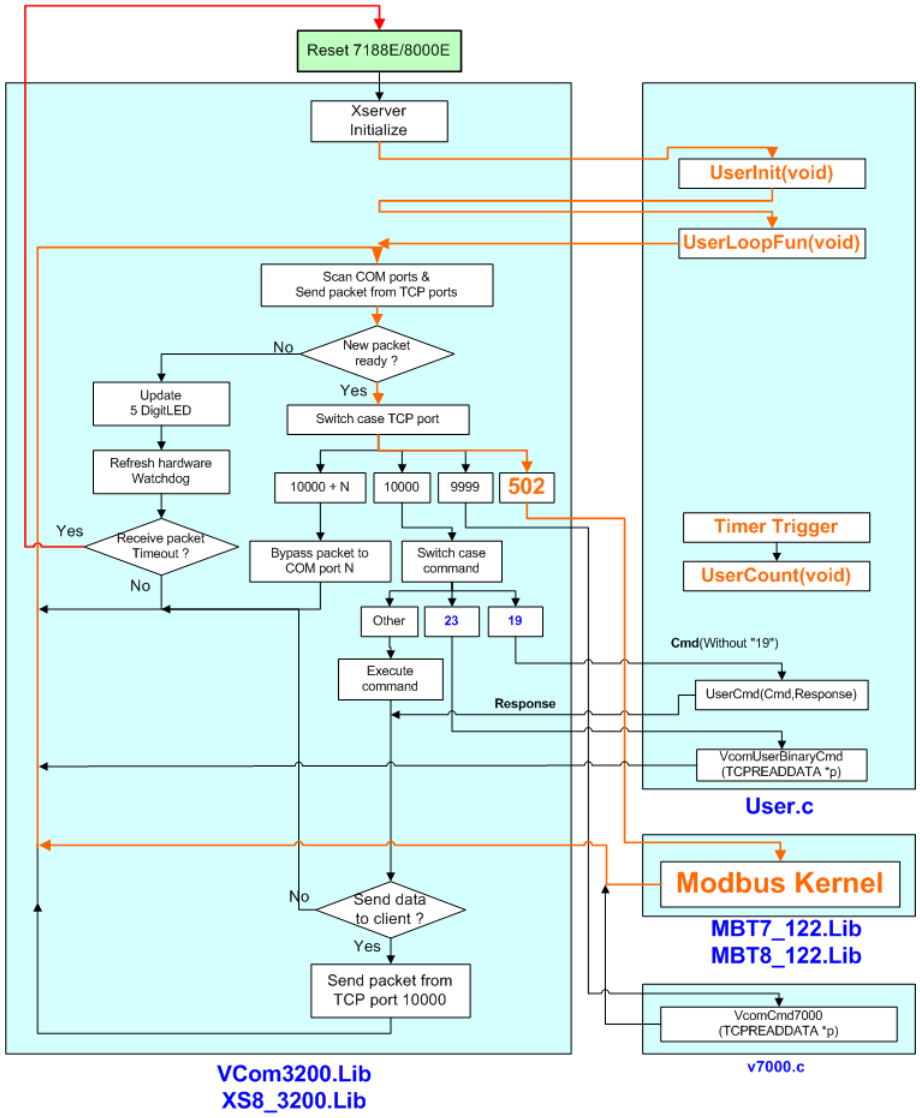


8000E -MTCP SDK Features

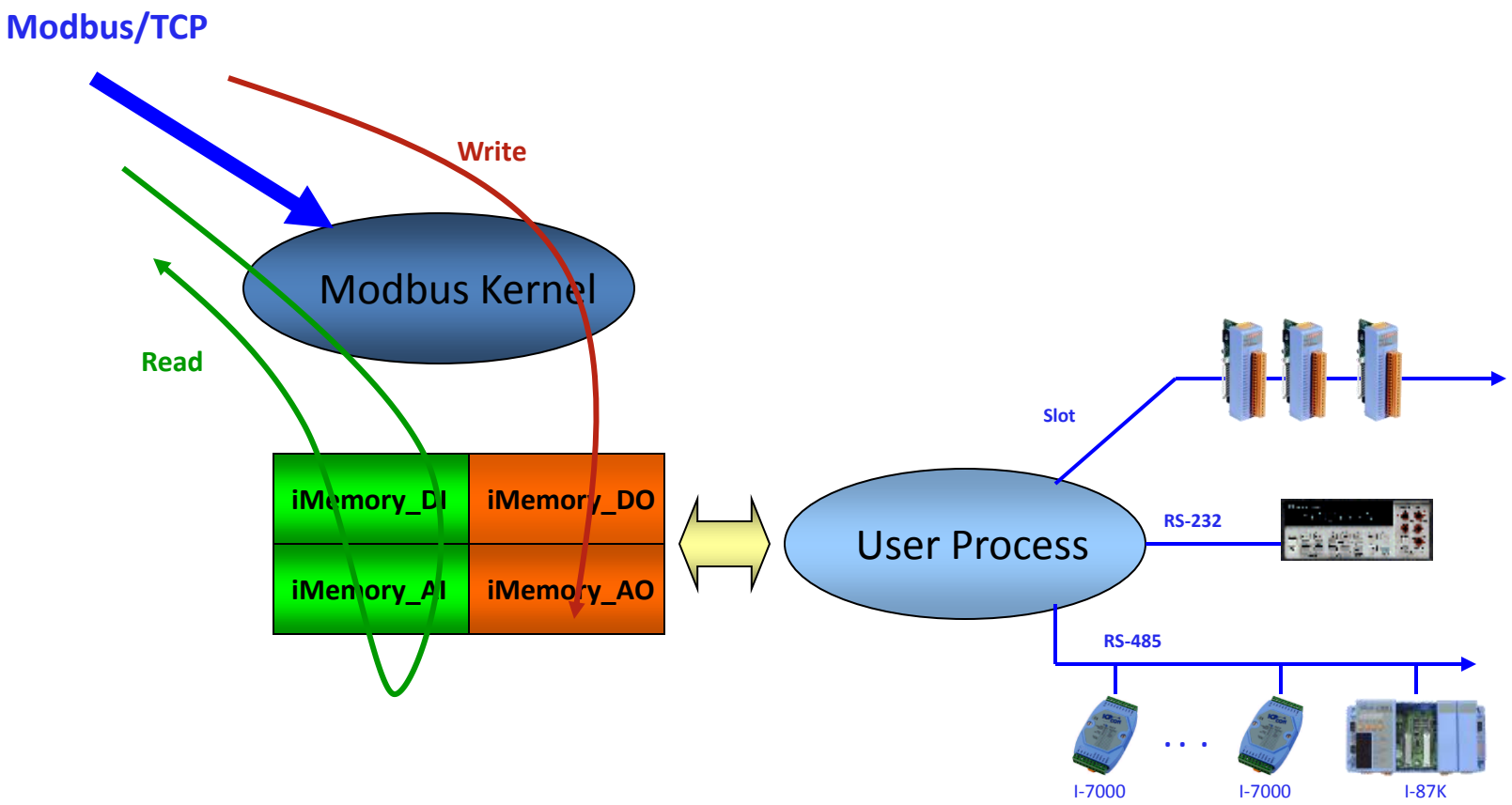
- 2 communication protocols
  - User-defined: port 10000
  - Modbus/TCP: port 502
- 4 Internal register tables (MTDemo50)



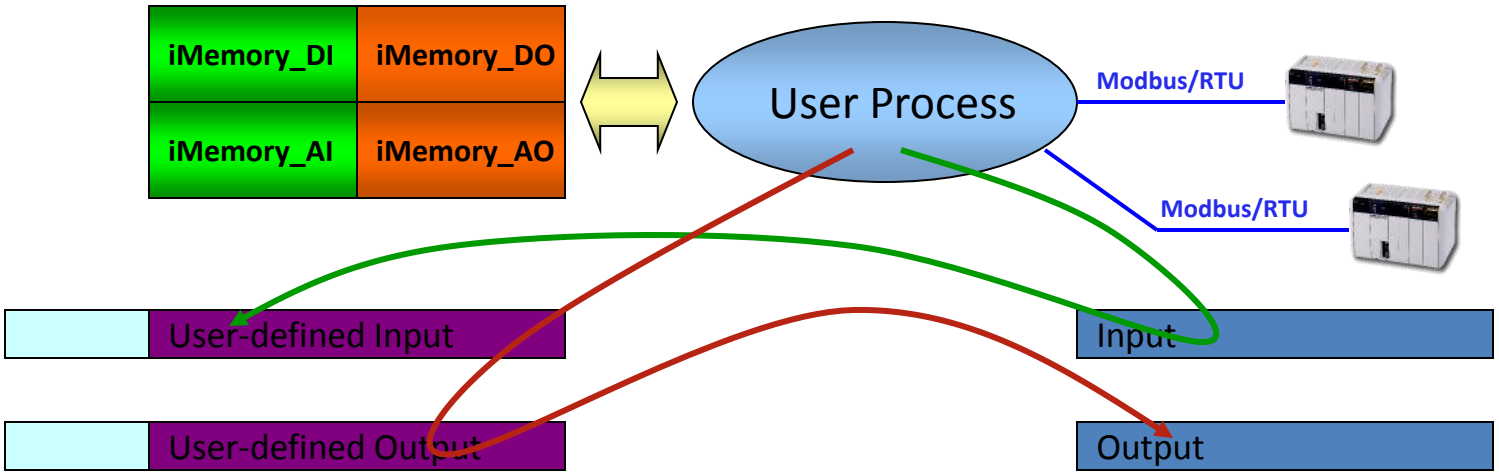
- Access I/Os that plug on slots (MTDemo51)
- Link I-7000 or I-87000 series modules via COM ports (MTDemo52)
- Modbus/RTU master (MTDemo53)


Reset 7188E/8000E
Xserver Initialize
UserInit(void)
UserLoopFun(void)
Scan COM ports & Send packet from TCP ports
No
New packet ready ?
Yes
Update 5 DigitLED
Switch case TCP port
Refresh hardware Watchdog
10000 + N
10000
9999
502
Yes
Receive packet Timeout ?
No
Bypass packet to COM port N
Switch case command
Timer Trigger
UserCount(void)
Other
23
19
Cmd(Without "19")
Execute command
Response
UserCmd(Cmd,Response)
VcomUserBinaryCmd (TCPREADDATA *p)
User.c
Modbus Kernel
MBT7_122.Lib
MBT8_122.Lib
No
Send data to client ?
Yes
Send packet from TCP port 10000
VcomCmd7000 (TCPREADDATA *p)
v7000.c
VCom3200.Lib
XS8_3200.Lib

User-defined Internal RegisterS



Be a Modbus/RTU Master



int **ModbusMaster2Slave**(int **iPort**,unsigned char **cNetID**, unsigned char **cFunction**,

int **iControllerMemoryBaseAddress**, int **iDeviceMemoryBaseAddress**,int **iIOCount**);

Modify 8000E-MTCP Firmware

- **User.c**

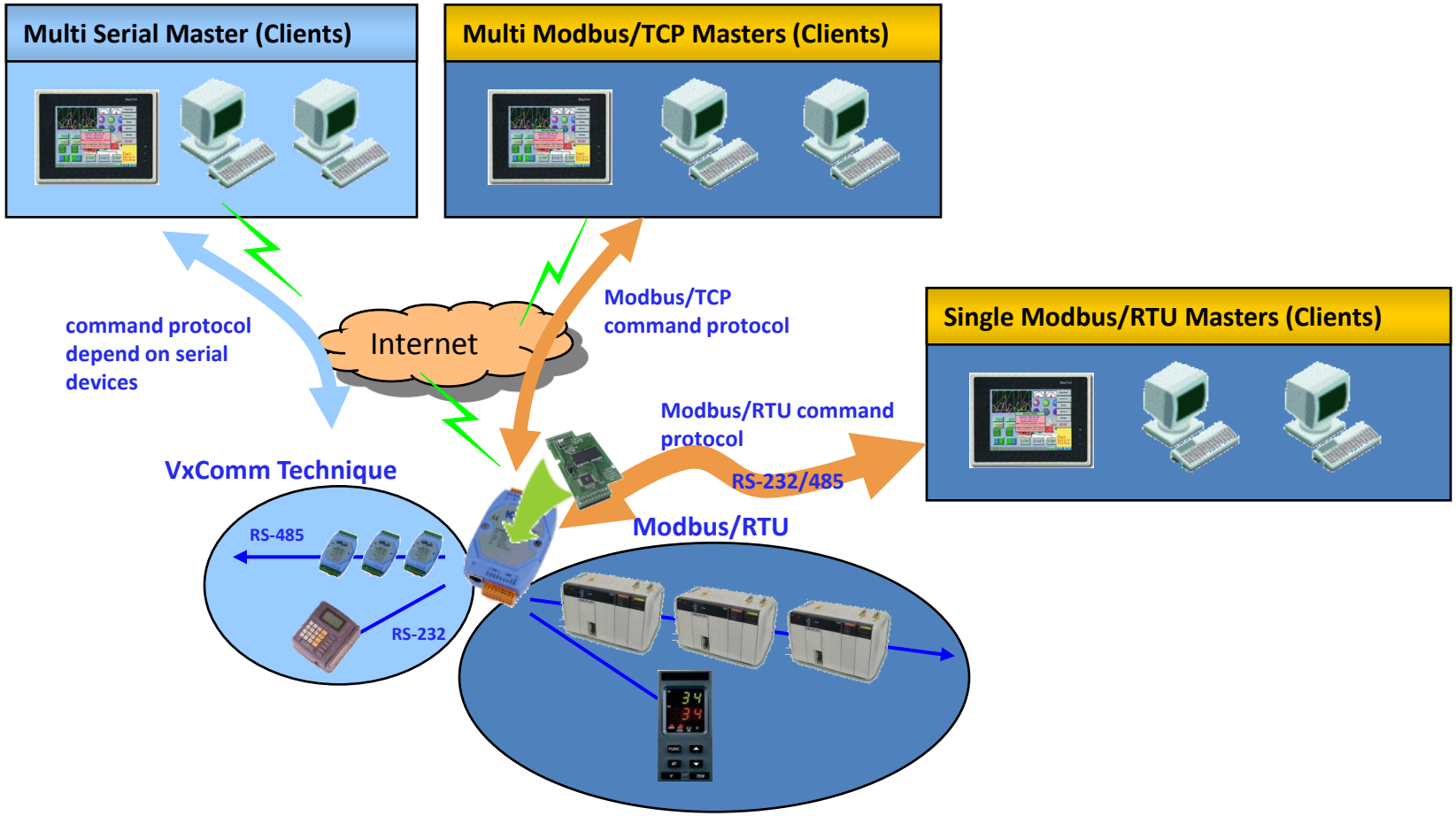
```
void UserInit(void)
{
        int iRet;
        iRet=InitModbus();
}

void UserLoopFun(void)
{
        UpdateIOModule();
      CheckModbusRequest(iModbusUpLinkPort);  //Is any Modbus/RTU request from COM port ?
      CheckLEDMenu();
}

int UserCmd(unsigned char *Cmd,unsigned char *Response)
{
        int iRet;

        if(Cmd[0]=='!')
                iRet=Configuration(Cmd,Response);

        return 1;
}
```

Modify 8000E-MTCP Firmware

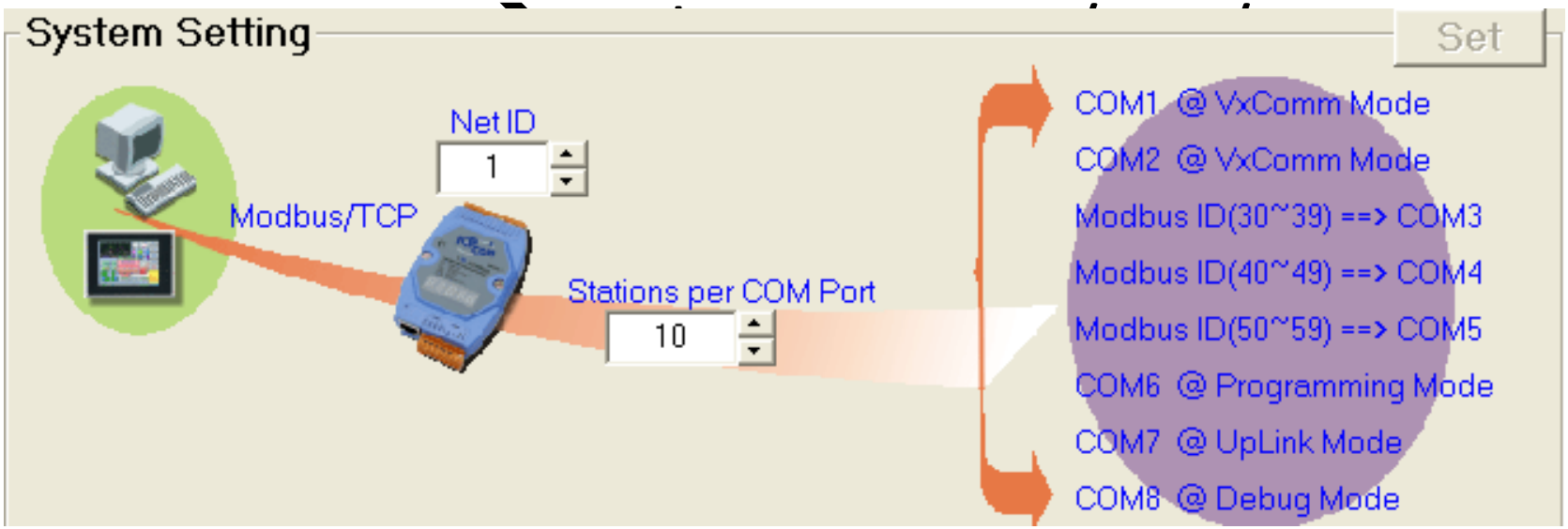
- **MBTCP_8E.h**

```
//Memory base address of every slot
extern unsigned int iMemoryAddr_DI[8];
extern unsigned int iMemoryAddr_DO[8];
extern unsigned int iMemoryAddr_AI[8];
extern unsigned int iMemoryAddr_AO[8];

//I/O points of every slot
extern unsigned int iMemoryNum_DI[8];
extern unsigned int iMemoryNum_DO[8];
extern unsigned int iMemoryNum_AI[8];
extern unsigned int iMemoryNum_AO[8];

//The I/O values
extern unsigned char* iMemory_DI;
extern unsigned char* iMemory_DO;
extern int*           iMemory_AI;
extern int*           iMemory_AO;

//Total DI,DO,AI,AO points
extern int iDINum,iDONum,iAINum,iAONum;
```

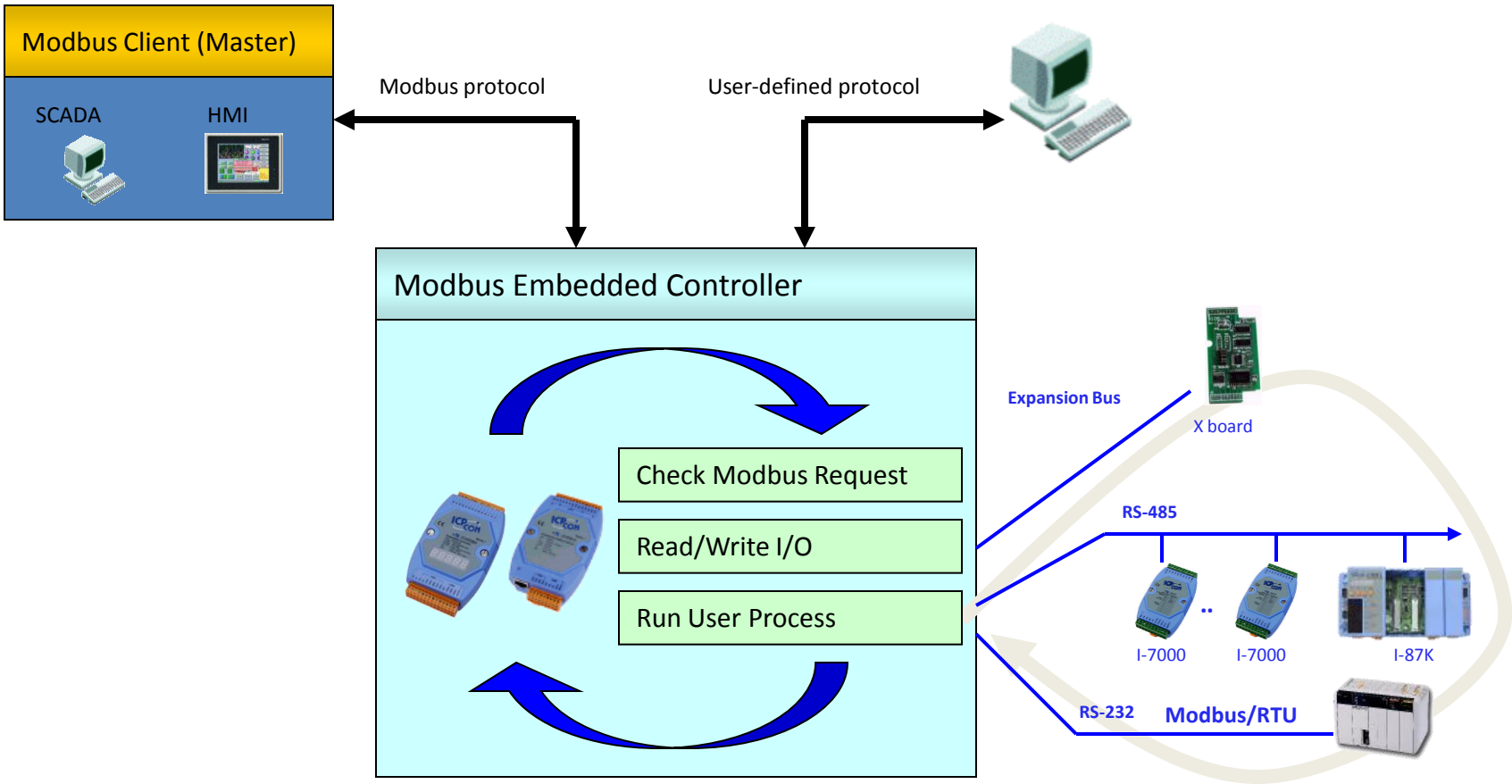
7188E-MTCP System Application



7188E-MTCP COM Port Enable Mode

- VxComm (Virtaul COM)
- Modbus/RTU ➔ Links to Modbus/RTU slave devices
- 

7188E-MTCP Internal Block

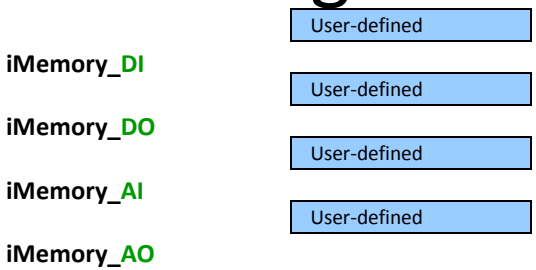


7188E-MTCP Features

- Converts single Modbus/TCP to multi Modbus/RTU (Modbus/TCP slave port)
- Converts single Modbus/RTU to multi Modbus/RTU (Modbus/RTU slave port)
- Supports VxComm technique for every COM port of controllers
- Allowed multi-client (or master) access simultaneously
- Firmware updateable and programmable

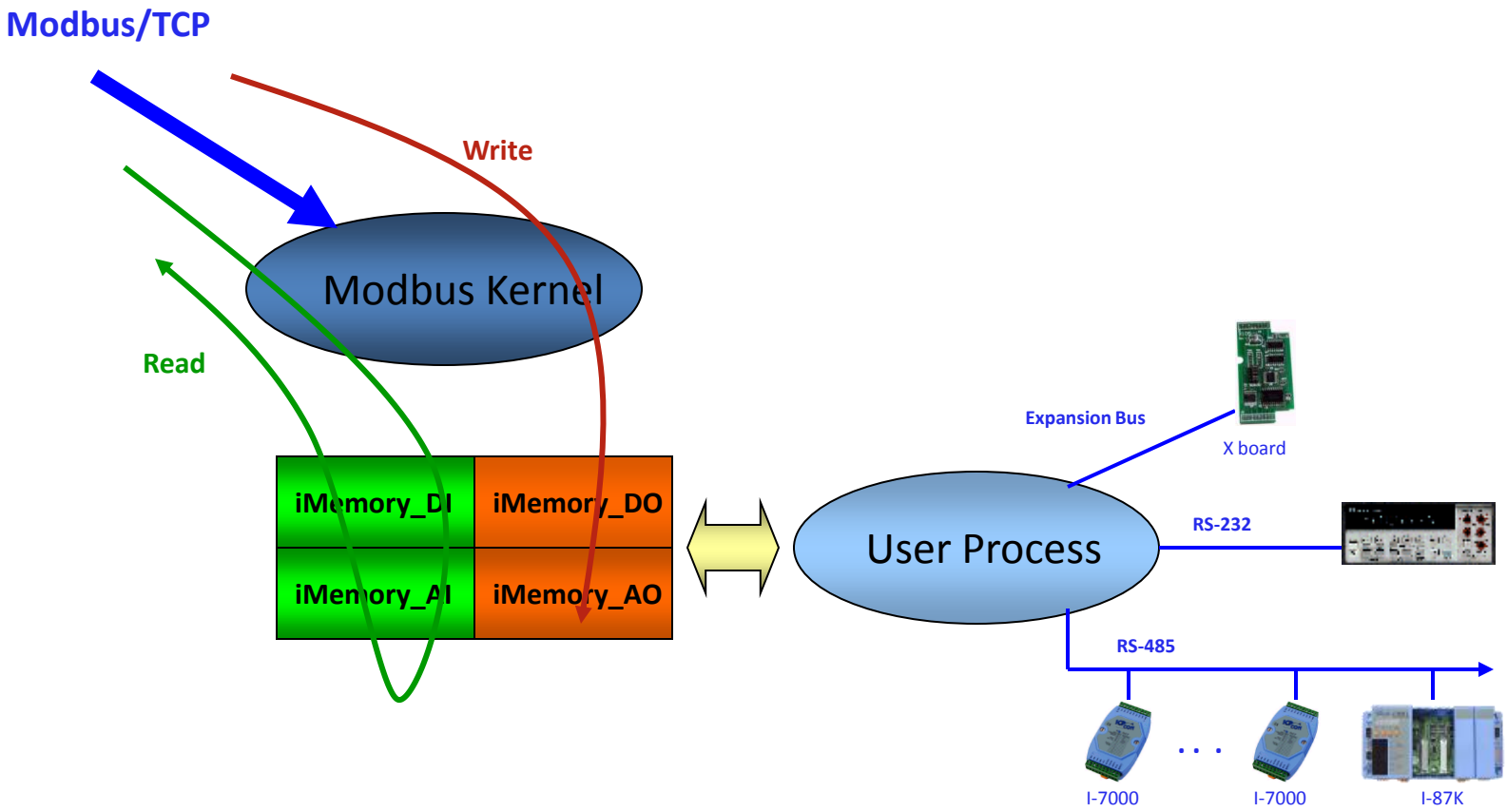
7188E-MTCP SDK Features

- Modbus/TCP to Modbus/RTU converter (Default function)
- 4 Internal register tables (MTDemo00)

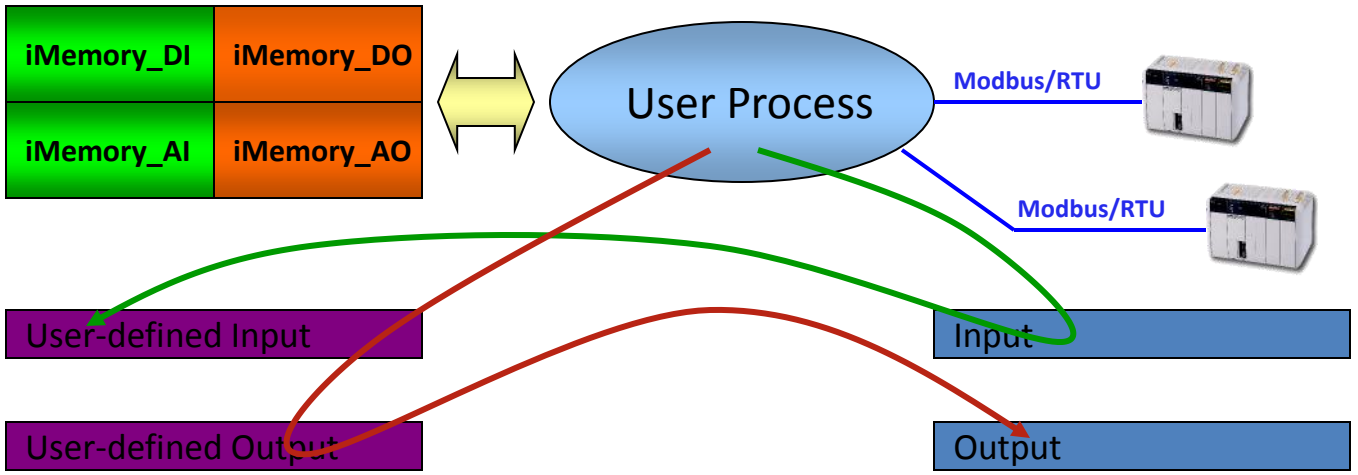


- Link I-7000 or I-87000 series modules via COM ports (MTDemo01)
- Access X-board (MTDemo02)
- Modbus/RTU master (MTDemo03)

User-defined Internal Register (7188XB,7188E)



Modbus/RTU master (7188XB,7188E)

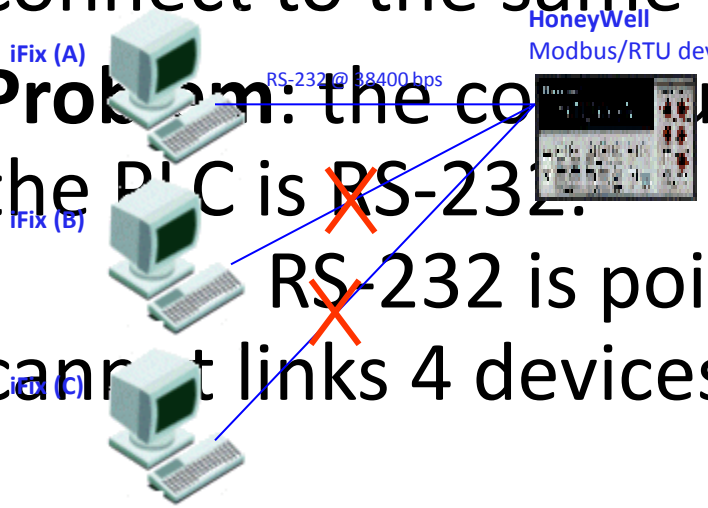


int **ModbusMaster2Slave**(int **iPort**,unsigned char **cNetID**, unsigned char **cFunction**,

int **iControllerMemoryBaseAddress**, int **iDeviceMemoryBaseAddress**,int **iIOCount**);

Modbus Gateway Application 1

- **Original system**: one PC connect to a HoneyWell PLC
- **Requirement**: allow two extra PCs to connect to the same PLC
- **Problem**: the communication interface of the PLC is RS-232.



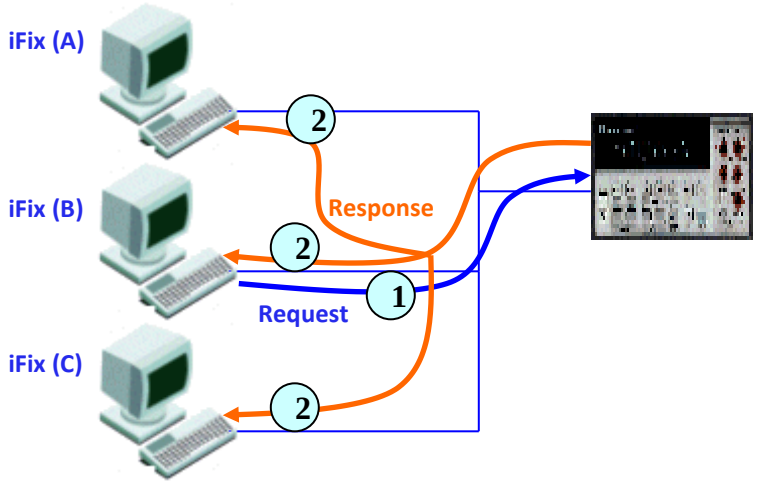
RS-232 is point to point interface, it cannot links 4 devices (3 PCs + 1 PLC)

Thinking 1 (RS-485 method)

- **Thinking**: RS-485 is a broadcast interface. Change to RS-485 interface can allow all PCs communicate with the PLC.

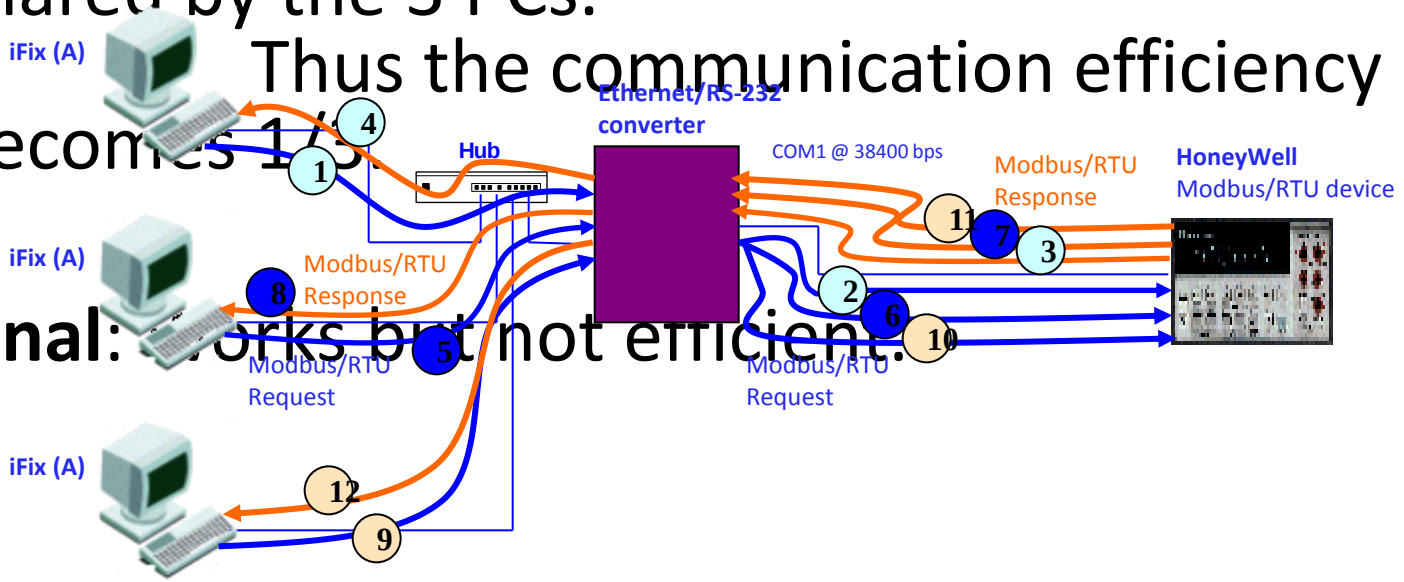
- **Problem**: The PLC will broadcast its response to every PC.
  The two PCs will feel confuse.

- **Final**: Doesn't work



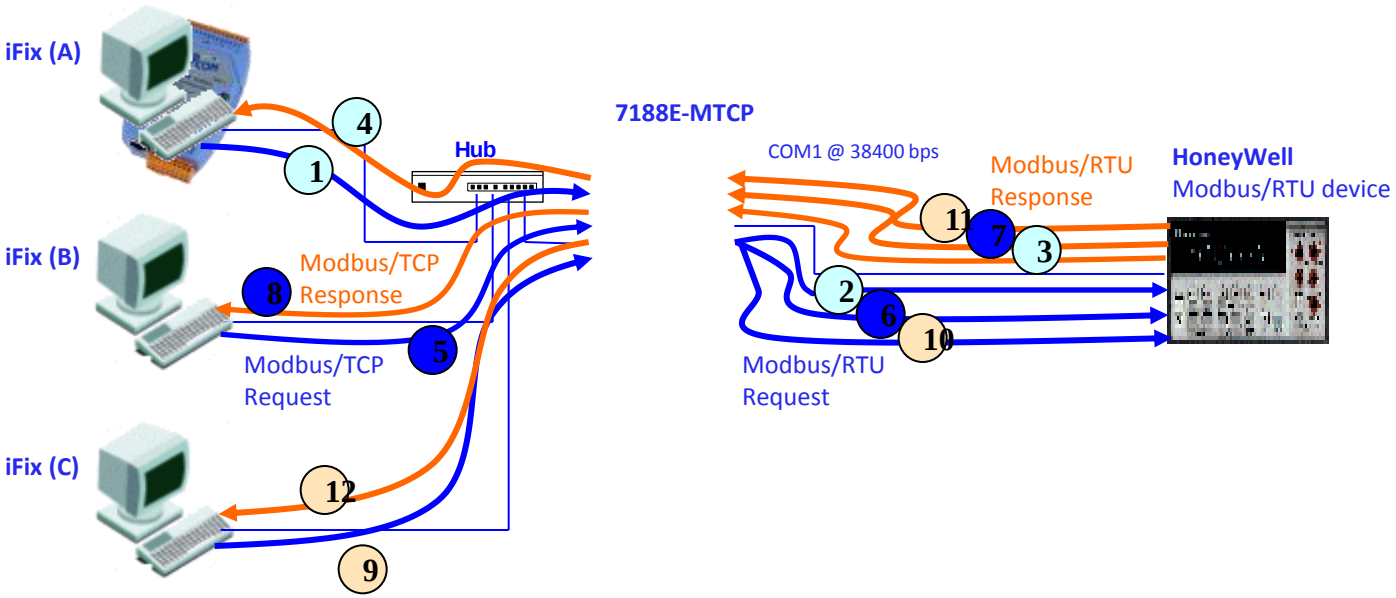
Thinking 2 (Ethernet to RS-232 converter)

- **Thinking**: the converter allow the 3 PCs share one COM port
- **Problem**: The communication band width is shared by the 3 PCs. Thus the communication efficiency becomes 1/3.
- **Final**: Works but not efficient.



Thinking 3 (Modbus/TCP Gateway)

- **Thinking**: Doesn't need to install extra VxComm driver on the PC
- **Problem**: The communication band width is shared by the 3 PCs.
  Thus the communication efficiency becomes 1/3.
- **Final**: Works but inefficient.

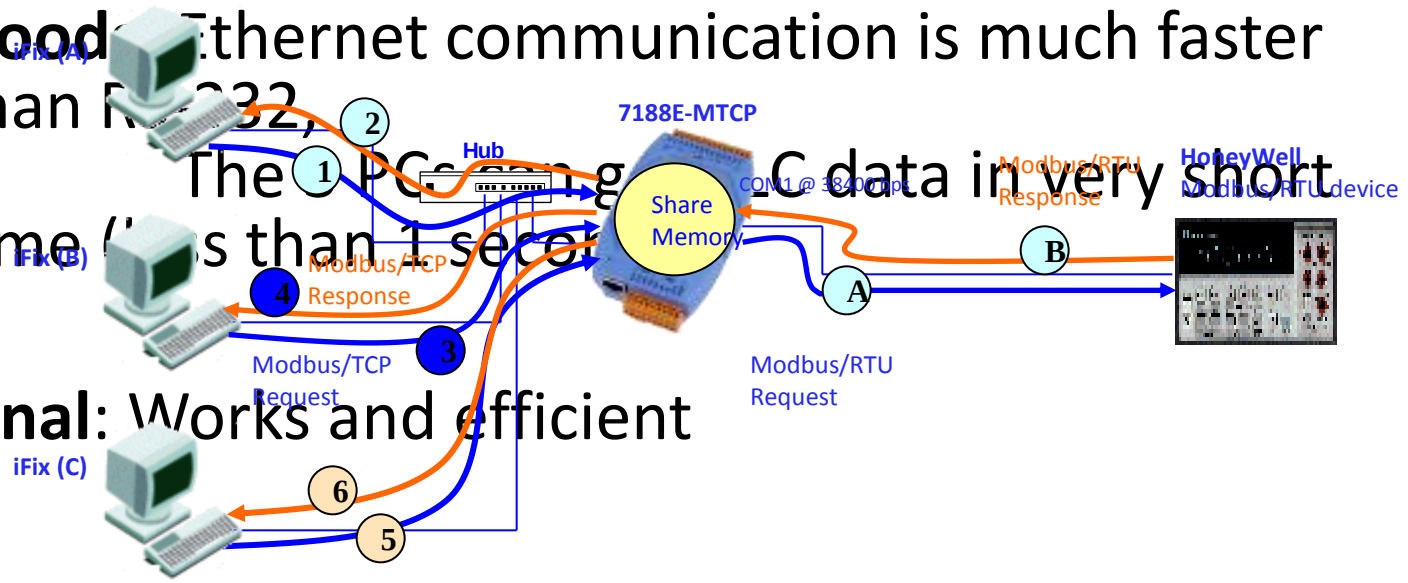


Thinking 3 (Modbus/TCP Gateway)

- **Thinking**: 7188E polls PLC's memory to its share memory
  The 3 PCs get PLC's data from the share memory

- **Good**: Ethernet communication is much faster than RS232,
  The 3 PCs get PLC data in very short time (less than 1 second)

- **Final**: Works and efficient

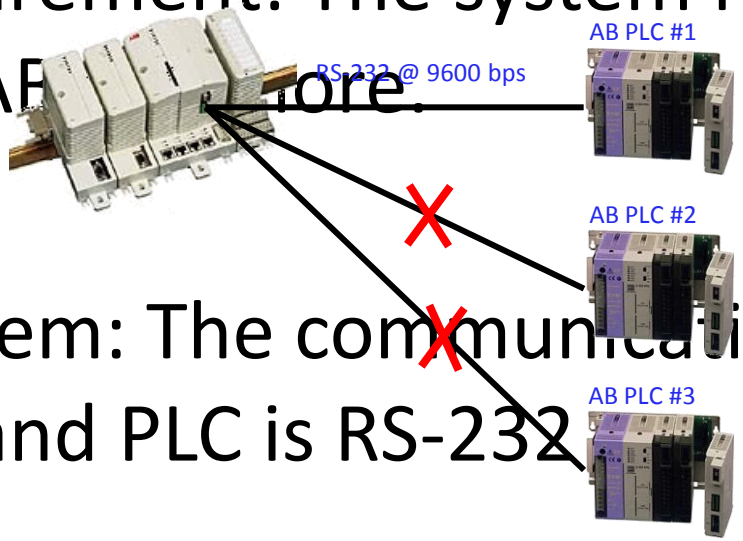


Modbus Gateway Application 2

- **Original system**: one ABB DCS connect to one AB PLC

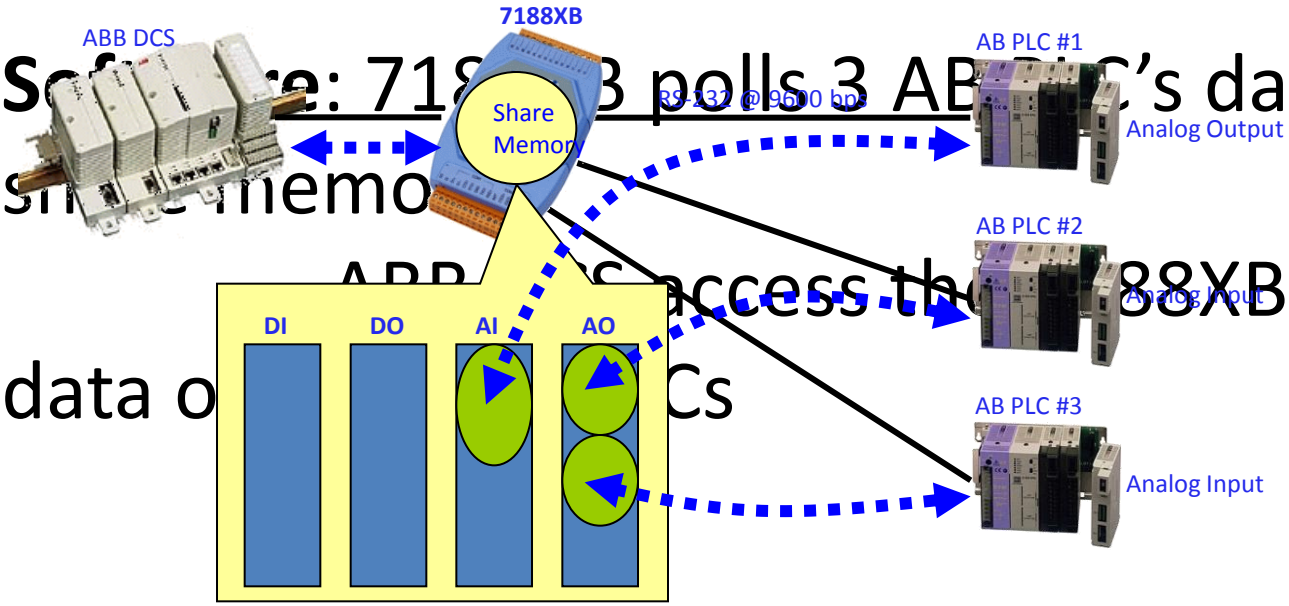
- Requirement: The system needs to include two AB PLC more.

- Problem: The communication interface of the DCS and PLC is RS-232



Solution

- **Hardware**: 7188XB + X505 = 4* RS-232 port + 1* RS-485
- **Software**: 7188XB polls 3 AB PLC's data to its share memory, ABB DCS access the 7188XB to get all data of PLCs



Multi PC access PLCs on the same RS-485