

YJKP Servo Press - Host Interface

Host interface of the servo press kit YJKP:

- Communication possibilities
- Object directory
- Communication protocol
- Communication Modbus TCP
- Communication EtherNet/IP
- Communication TCP/IP
- Communication Profinet IO

YJKP

TitleServo press kit YJKP - Host interface
Version 1.80
Document no. 100132
Originalen
AuthorFesto

Last saved 05.05.2026

General information:

This document is intended for qualified, trained and instructed professionals. The data provided in this document are no guaranteed specifications, in particular with regard to functionality, condition or quality in the legal sense. The information in this document is intended only as simple indications for the implementation of a specific, hypothetical application, and in no way as a substitute for the operating instructions of the respective manufacturers or the design and testing of the respective application by the user. The respective operating instructions for Festo products can be found under www.festo.com. The user of this document must ensure that each function described herein works properly in his application. The user remains solely responsible for his or her own use of this document, even by studying this document and using the information mentioned therein. This also applies to any software (in source code and/or object code) that is made available to the user as an appendix to this document.

Rights of use of the software:

If software is made available to the user as an appendix to this document, the user is granted a simple and unlimited right of use hereto. This right also includes the processing and distribution of the software to third parties in edited or unedited form. The User is not permitted to use the name "Festo" to endorse or promote products derived from the Software without express prior written permission.

Software is provided to the user "as is". Festo does not assume any warranty or guarantee with regard to software. Festo's liability for damages of any kind arising from the use of the software is limited to intent and gross negligence.

Legal Notices:

©Festo SE & Co. KG, all rights reserved. A change in content and form is only permitted with the express written consent of Festo SE & Co. KG. Festo grants the user the right to reproduce this document in a form that remains unchanged in terms of content and form and to pass it on to third parties.

Table of contents

1	Applicable Components/Software	5
2	Application description in general	6
2.1	Function block for different platforms: plug-and-play	6
3	Communication possibilities	7
4	Object directory	8
4.1	Object description	8
4.1.1	Identifier	8
4.1.2	Index	8
4.1.3	Sub Index	8
4.1.4	Value	8
4.1.5	Data type	9
4.1.6	Data size	9
4.1.7	Access type	9
5	Communication protocol.....	10
5.1.1	Request Interface	10
5.1.2	Response Interface	14
5.1.3	Acknowledge description	18
5.1.4	Monitoring communication	18

6	Communication Modbus TCP	19
6.1	Communication task and devices	19
6.2	Example settings Modbus TCP Client (CODESYS)	20
6.3	Example settings Modbus TCP Server (CODESYS)	21
6.4	Handshake	22
6.5	Communication example	23
7	Communication EtherNet/IP.....	24
7.1	Communication task and devices	24
7.2	Example setting EtherNet/IP Client (CODESYS)	25
7.3	Handshake	28
7.4	Communication example	29
8	Communication TCP/IP.....	30
8.1	Implementation TCP/IP (CODESYS)	30
8.2	Example implementation TCP/IP Client (CODESYS)	30
8.3	Handshake	30
8.4	Communication example	31
9	Communication Profinet IO.....	32

1 Applicable Components/Software

Type/Name	Version Software/Firmware
Servo press kit YJKP	General
Application software YJKP	GSAY-A4-F0-Z4-2.1.5
Firmware controller (CECC-X)	V3.8.14
Firmware motor controller (CMMT-AS)	V32.0.9.9
Host library / function blocks	V2.1.5

Table 1.1: Applicable software and components used in this application note

GSAY	CECC-X Firmware	CMMT-AS Firmware
GSAY-A4-F0-Z4-2.1.3	3.8.14	32.0.9.9
GSAY-A4-F0-Z4-2.1.5	3.8.14	32.0.9.9
GSAY-A4-F0-Z4-2.1.7	3.8.14	33.0.9.10
GSAY-A4-F0-Z4-2.1.9	3.8.14	33.0.9.10
GSAY-A4-F0-Z4-2.1.11	3.8.4	33.0.9.10

Table 1.2: Recommended firmware for each GSAY in EVO 4.1

GSAY	Host Function Block
Lower than GSAY-1.4.1	1.0.1
From GSAY-1.4.1	1.1.1
From GSAY-2.1.5	1.1.1 2.1.5

Table 1.3: Application software and host function block compatibility of all EVO versions

2 Application description in general

This application note describes how you can communicate with a higher level plc with the servo press kit.



Note

It refers to the functionality of the application software YJKP version GSAY-1.4.1 and above.
For lower version use other appropriate application note, which is also available on the Festo website.

Following descriptions are part of the application note:

- Communication possibilities
- Object directory
- Communication protocol
- Communication Modbus TCP
- Communication EtherNet/IP
- Communication TCP/IP
- Communication Profinet IO

2.1 Function block for different platforms: plug-and-play

We also provide the prepared [Function blocks](#) to control YJKP on following platform:

- Siemens : Profinet IO
- Rockwell / AllenBradley : EtherNet/IP
- Codesys : ModbusTCP
- Omron : EtherNet/IP
- Mitsubishi : ModbusTCP
- Beckhoff : Modbus TCP

Please follow the instructions in the [application note](#) “AppNote Servo Press Kit YJKP - Integration of host function blocks in **” for each platform, which you can find on our support portal searching for YJKP in the tab expert knowledge.

3 Communication possibilities

The standard fieldbus of the servo press kit are:

- TCP/IP
- Modbus TCP
- EtherNet/IP
- Profinet IO Device

Other fieldbus can be used over a Anybus X-gateway Modbus-TCP (Gateway is not part of the servo press kit package!):

- CANopen-Slave
- CC-Link-Slave
- ControlNet-Slave
- DeviceNet-Slave
- EtherCAT-Slave
- EtherNet/IP-Slave
- Modbus-RTU-Slave
- Modbus-TCP-Slave
- Profibus-Slave

4 Object directory

The object directory is a list of objects, which describes communication and application parameters. You can see the whole objects of the object directory in the document: *AppNote Servo Press Kit YJKP - Host interfaceV1.5 - Object directory.pdf*. (The document is part of the download file) The object directory describes the whole functionality of the system. A CSV-File which is stored internally represents the object directory. This file contains all objects needed to control the system or get the status of the system.

4.1 Object description

A object is a data structure which includes all properties of a parameter. Afterwards all structure elements will be described.

4.1.1 Identifier

Is a description of a object as string.

4.1.2 Index

The index in connection with the sub index is a unique identification number for a object.

4.1.3 Sub Index

For standard objects the sub index is 0. Objects with a structure and more elements are separated through the sub index, but are part of the structure through the same index.

4.1.4 Value

Actual value of the object.

4.1.5 Data type

The data type describes how the byte order for the object must be interpreted.

Index	Type	Bytes	Description	Range of values
0x00	--	--	Unbekannter Datentyp	--
0x01	BOOL	1	8 bit boolean	0 (FALSE) ... 1 (TRUE)
0x02	SINT	1	8 bit signed short integer	-128 ... 127
0x03	USINT	1	8 bit unsigned short integer	0 ... 255
0x04	INT	2	16 bit signed integer	-32768 ... 32767
0x05	UINT	2	16 bit unsigned integer	0 ... 65535
0x06	DINT	4	32 bit signed long integer	-2147483648 ... 2147483647
0x07	UDINT	4	32 bit unsigned long integer	0 ... 4294967295
0x08	BYTE	1	8 bit unsigned short integer	0 ... 255
0x09	WORD	2	16 bit unsigned integer	0 ... 65535
0x0A	DWORD	4	32 bit unsigned long integer	0 ... 4294967295
0x0B	REAL	4	32 bit float	1.400e-45 ... 3.403e+38

Table 4.1: Data types

4.1.6 Data size

Size of the object in order to the data type.

4.1.7 Access type

Defines how the object can be accessed:

- RO: read only
- WO: write only
- RW: readable and writeable

5 Communication protocol

The basic functionality between the control device and the servo press kit is the client server principle. The servo press kit is the server and shares all his supported functions. The control device is the client and can use all shared functions after established connection. The communication with the servo press kit performs on basic of the request response principle. That means after established connection the servo press kit is waiting for a request from the client and sends after receiving the request the response to the client.

5.1.1 Request Interface

The interface used to communicate with servo press kit:

Request				
<i>Data byte</i>	<i>Name</i>	<i>Data type</i>	<i>Data size</i>	<i>Description</i>
1	usiMessageID	USINT	1	Message ID Free usable message ID of the application Can be set from the client, in the related answer the same message ID will be send to the client A unique assignment of the request and response is possible
2	dwControl word	DWORD	4	Control word Bit 0 = Manual mode Bit 1 = Automatic mode Bit 2 = Start homing Bit 3 = Start press process Bit 4 = Abort press process (low active) Bit 5 = Load program Bit 6 = Quit error Bit 7 = Enable press Bit 8 = Tare Bit 9 = Move Bit 10 = Stop move Bit 11 = Jog pos Bit 12 = Jog neg Bit 13 = Load hardware configuration Bit 14 = Logging USB Bit 15 = Logging SD-Card Bit 16 = System reset Bit 17 = Reset statistic Bit 18 = Step mode Bit 19 = Logging FTP-Server Bit 20 = Force control continue Bit 21 = Open holding brake 22-31 = Reserved
3				
4				
5				

6	diOffsetForceSensor	DINT	4	Offset of the force sensor Unit [N] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 N)
7				
8				
9				
10	byMotionMode	BYTE	1	Motion mode (Jog, Move absolute/relative) 0x00 = Jog 0x01 = Move absolute 0x02 = Move relative
11	diMotionVelocity	DINT	4	Motion velocity (Jog, Move absolute/relative) Unit [mm/s] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 mm/s)
12				
13				
14				
15	diMotionPositionDistance	DINT	4	Motion position/distance (Move absolute/relative) Unit [mm] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 mm)
16				
17				
18				
19	uiSelectedProgramNumber	UINT	2	Selected program number
20				
21	byDigitalInputs	BYTE	1	Digital inputs (Step enabling conditions in the sequencer of the press process) Bit 0 = Input 9 Bit 1 = Input 10 Bit 2 = Input 11 Bit 3 = Input 12 Bit 4 = Input 13 Bit 5 = Input 14 Bit 6 = Input 15 Bit 7 = Input 16
22	Reserved	Reserved	Reserved	Reserved
23				
24				
25				
26	Reserved	Reserved	Reserved	Reserved
27				
28				
29				
30	Reserved	Reserved	Reserved	Reserved
31				
32				
33				
34	Reserved	Reserved	Reserved	Reserved
35				
36				
37				
38	Reserved	Reserved	Reserved	Reserved
39				
40				

41				
42	Reserved	Reserved	Reserved	Reserved
43				
44	byControlParameterChannel	BYTE	1	Control paramter channel Bit 0 = Execute
45	usiAccessType	USINT	1	Access type 0x00 = Read object 0x01 = Write object
46	usiAccessOption	USINT	1	Access option 0x00 = Single read/write object 0x01 = Multi read/write object
47	uiObjectIndex	UINT	2	Object index
48				
49	usiObjectSubIndex	USINT	1	Object sub index
50	usiObjectDataType	USINT	1	Object data type
51	uiDataLengthToBeWritten	UINT	2	Data length of the value to be written
52				
53	abyDataToBeWritten	ARRAY [1 .. 76] OF BYTE	76	Data bytes of the value to be written
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
78				
80				
81				
82				
83				
84				
85				

86				
87				
88				
89				
90				
91				
93				
94				
95				
97				
98				
99				
100				
101				
102				
103				
104				
105				
106				
107				
108				
109				
110				
111				
112				
113				
114				
115				
116				
117				
118				
119				
120				
121				
122				
123				
124				
125				
126				
127				
128				

Table 5.1: Request interface

5.1.2 Response Interface

Response				
<i>Data byte</i>	<i>Name</i>	<i>Data type</i>	<i>Data size</i>	<i>Description</i>
1	usiMessageID	USINT	1	Message ID Message ID related to the request sent by the client
2	dwStatusWord	DWORD	4	Status word
3				Bit 0 = Manual mode
4				Bit 1 = Automatic mode
5				Bit 2 = Homing required Bit 3 = Program loaded Bit 4 = Step mode Bit 5 = In operation Bit 6 = Step done Bit 7 = OK Bit 8 = NOK Bit 9 = Press enabled Bit 10 = Tared Bit 11 = Status ready (Status hardware configuration = TRUE Homing required = FALSE Status error = FALSE Status warning = FALSE) Bit 12 = Status servo press ready (Status ready = TRUE In operation = FALSE) Bit 13 = Status error Bit 14 = Status warning Bit 15 = Status safety Bit 16 = Status hardware configuration Bit 17 = Status logging USB Bit 18 = Status logging SD-Card Bit 19 = Status logging FTP-Server Bit 20 = Force reached Bit 21 = Holding brake opened Bit 22-23 = Reserved Bit 24-26 = Control highness Bit 27-31 = Reserved
6	diActualOffsetForceSensor	DINT	4	Actual offset of the force sensor Unit [N] 1×10^{-2} (2 decimal places, e.g. 1 = 0,01 N)
7				
8				
9				
10	byActualMotionMode	BYTE	1	Actual motion mode (Jog, Move absolute/relative) 0x00 = Jog 0x01 = Move absolute 0x02 = Move relative
11	diActualMotionVelocity	DINT	4	

12				Actual motion velocity (Jog, Move absolute/relative) Unit [mm/s] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 mm/s)
13				
14				
15	diActualMotionPositionDistance	DINT	4	Actual motion position/distance (Move absolute/relative) Unit [mm] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 mm)
16				
17				
18				
19	uiLoadedProgramNumber	UINT	2	Loaded program number
20				
21	byDigitalOutputs	BYTE	1	Digital outputs (Configurable as output signals in the sequencer of the press process) Bit 0 = Output 5 Bit 1 = Output 6 Bit 2 = Output 7 Bit 3 = Output 8 Bit 4 = Output 9 Bit 5 = Output 10 Bit 6 = Output 11 Bit 7 = Output 12
22	diActualPosition	DINT	4	Actual position Unit [mm] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 mm)
23				
24				
25				
26	diActualForce	DINT	4	Actual force Unit [N] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 N)
27				
28				
29				
30	diActualVelocity	DINT	4	Actual velocity Unit [mm/s] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 mm/s)
31				
32				
33				
34	diMaximumPosition	DINT	4	Maximum position in the press process Unit [mm] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 mm)
35				
36				
37				
38	diMaximumForce	DINT	4	Maximum force in the press process Unit [N] $1 * 10^{-2}$ (2 decimal places, e.g. 1 = 0,01 N)
39				
40				
41				
42	wNOK_Reason	WORD	2	NOK reason of the press process More detailed information see online help of the YJKP
43				
44	byStatusParameterChannel	BYTE	1	Status parameter channel Bit 0 = Busy Bit 1 = Done
45	byAcknowledge	BYTE	1	Acknowledge
46	usiReadObjectDataType	USINT	1	Read object data type

47	uiReadDataLength	UINT	2	Data length of the value to be written
48				
49	abyReadData	ARRAY [1 .. 76] OF BYTE	76	Data bytes of the read value
50				
51				
52				
53				
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
92				
93				
94				
96				
97				
98				

99				
100				
101				
102				
103				
104				
105				
106				
107				
108				
109				
110				
111				
112				
113				
114				
115				
116				
117				
118				
119				
120				
121				
122				
123				
124				
125	Reserved	Reserved	Reserved	Reserved
126	Reserved	Reserved	Reserved	Reserved
127	Reserved	Reserved	Reserved	Reserved
128	Reserved	Reserved	Reserved	Reserved

Table 5.2: Response interface

5.1.3 Acknowledge description

Ack	Description	Recovery
0x00	OK	--
0x01	Service is not supported	Check service ID in the request data
0x02	Data length invalid	Check data length in the request data
0x03	Object index invalid	Check index and sub index of the object in the request data
0x04	Access type invalid	Check access type of the object in the request data
0x05	Data type invalid	Check data type of the object in the request data
0x06	Value invalid	Check range of values of the object in the request data
0x07	Read type invalid	Check read type in the request data
0x08	Write type invalid	Check write type in the request data
0x09	Values invalid	Check range of values of the object in the request data

Table 5.3: Acknowledge description

5.1.4 Monitoring communication

A timeout mechanism is implemented within the cyclic communication. The server will reset the communication if the timeout (2s) is elapsed without receiving any data from the client. In this case the client must reconnect (TCP/IP) again to the server. The elapsed time will be reset each time the server receives data from the client in cyclic communication.

6 Communication Modbus TCP

This example is to show, how to set up the connection between a client and a server based on Codesys.

6.1 Communication task and devices

The communication task between client and server has to be set to 4ms.

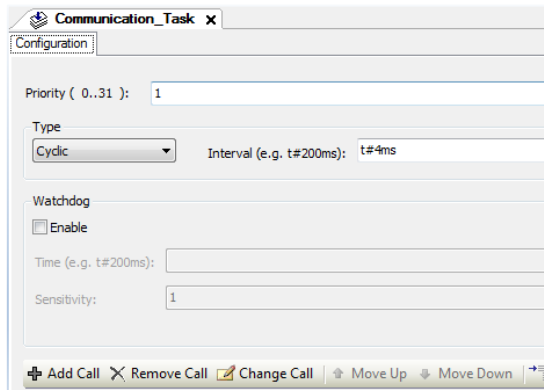


Figure 6-1: Communication task

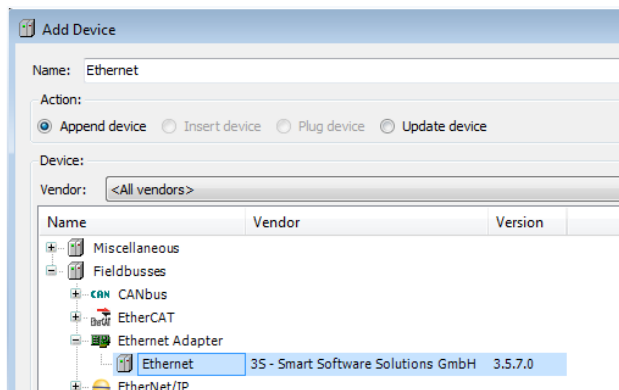


Figure 6-2: Add Ethernet

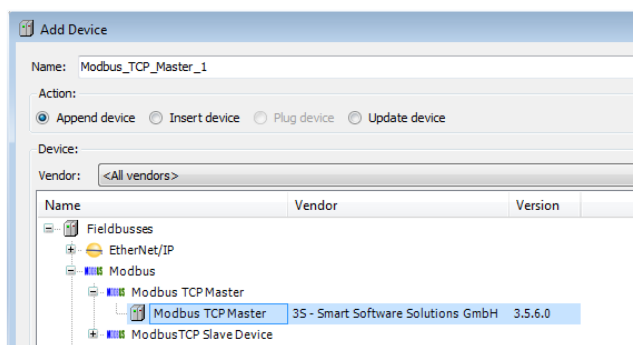


Figure 6-3: Add Modbus TCP Master

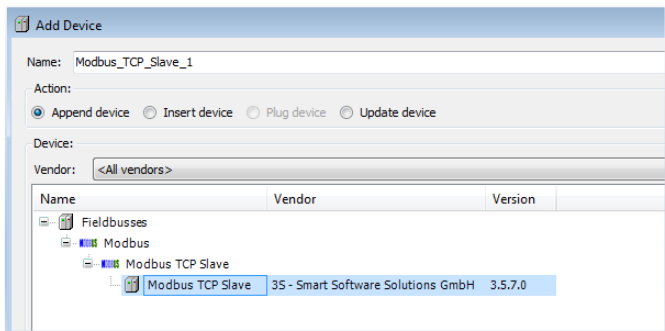


Figure 6-4: Add Modbus TCP Slave

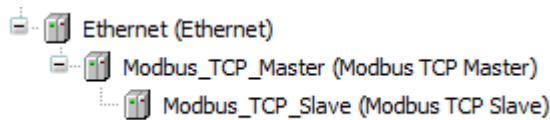


Figure 6-5: All necessary components

6.2 Example settings Modbus TCP Client (CODESYS)

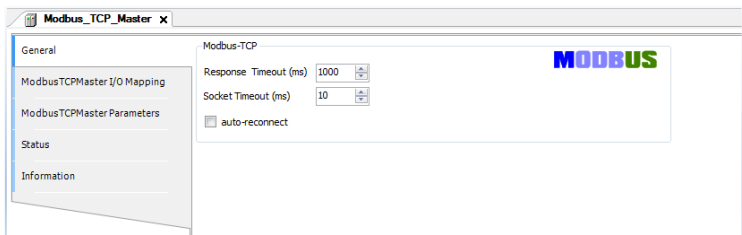


Figure 6-6: Modbus TCP Client "General"

Parameter	Type	Value	Defa...	Unit	Description
ExtendedChannelConfig	BOOL	true	true		Use the new Channel-Config format
OptimizationOn	BOOL	TRUE	TRUE		the driver optimizes the io update
Socket Timeout	UDINT	10	10		Socket Timeout in milliseconds
ResponseTimeOut	UDINT	1000	1000		Response time in milliseconds
AutoReconnect	BOOL	FALSE	FALSE		auto-confirm error and re-establish TCP connection
ModbusTCP Slave Instance					Implicit Function Block for Modbus Slaves.
FBType	STRING	'Modb...	'Modb...		
FBInstName	STRING	'Modb...	'Modb...		
InitMethodName	STRING	'Initia...	'Initia...		

Figure 6-7: Modbus TCP Client "ModbusTCPMaster Parameters"

6.3 Example settings Modbus TCP Server (CODESYS)

Slave IP address: IP address of the controller CECC-X

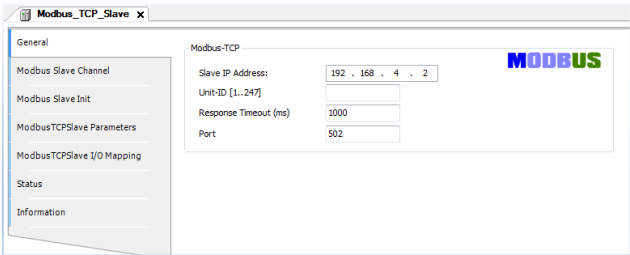


Figure 6-8: Modbus TCP Server "General"

Add a new slave channel “Read Input Registers (Function Code 4) with a length of 52 (104 Bytes).

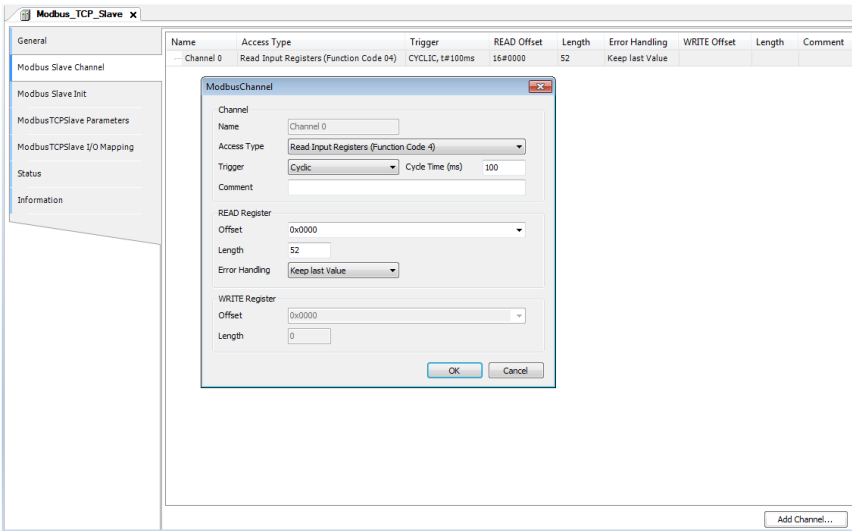
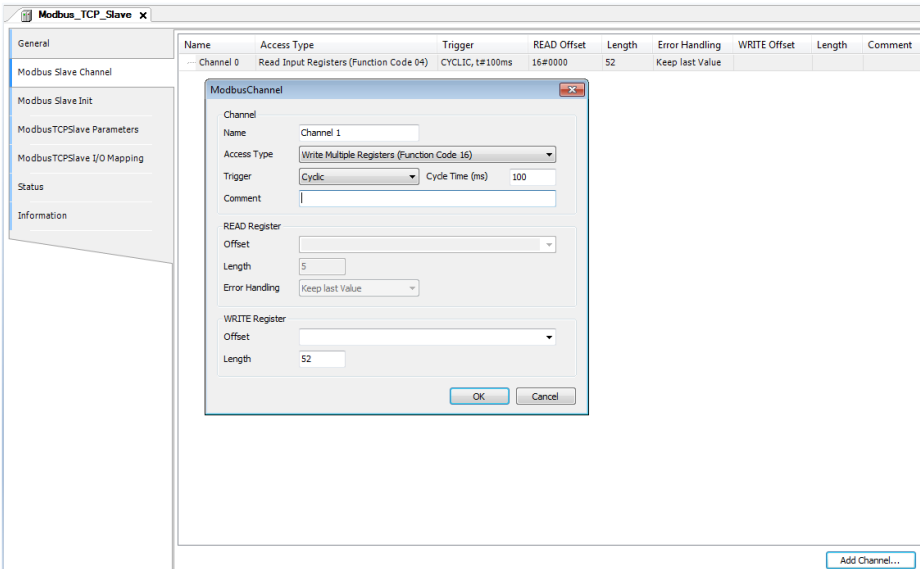
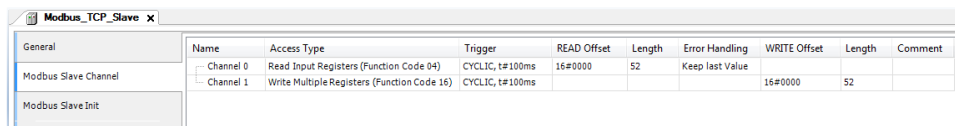


Figure 6-9: Modbus TCP Server Read Channel

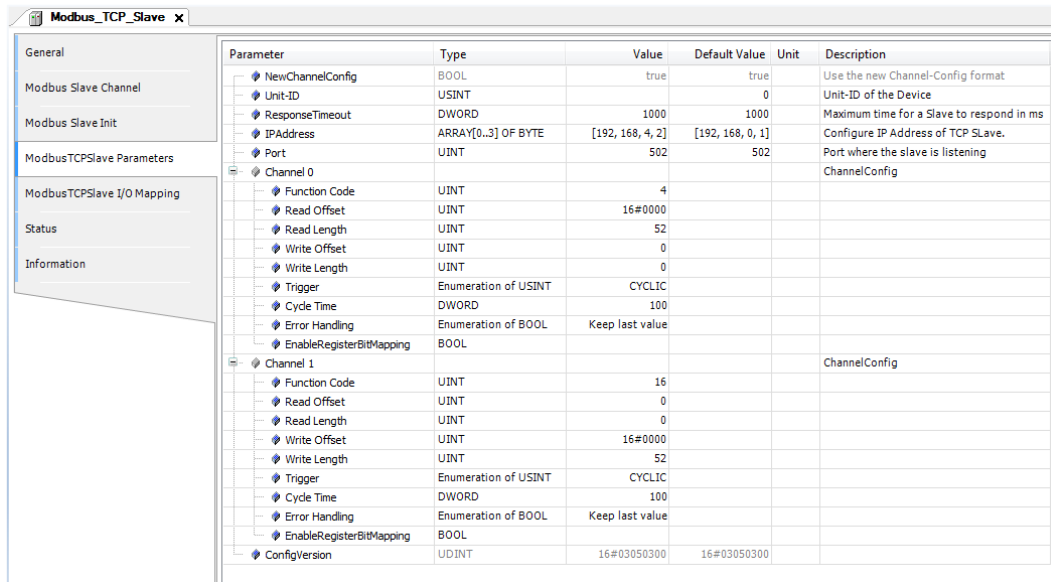
Add a new slave channel “Write Multiple Registers (Function Code 16) with a length of 52 (104 Bytes).





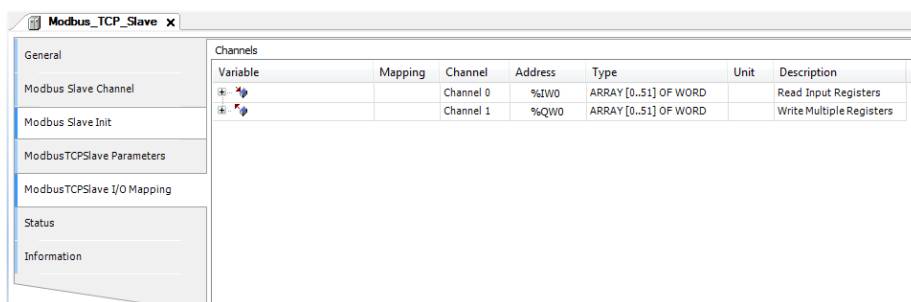
General	Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset	Length	Comment
Modbus Slave Channel	Channel 0	Read Input Registers (Function Code 04)	CYCLIC, 1#100ms	16#0000	52	Keep last value			
Modbus Slave Init	Channel 1	Write Multiple Registers (Function Code 16)	CYCLIC, 1#100ms				16#0000	52	

Figure 6-10: Final view slave channels



Parameter	Type	Value	Default Value	Unit	Description
NewChannelConfig	BOOL	true	true		Use the new Channel-Config format
Unit-ID	USINT		0		Unit-ID of the Device
ResponseTimeout	DWORD	1000	1000		Maximum time for a Slave to respond in ms
IPAddress	ARRAY[0..3] OF BYTE	[192, 168, 4, 2]	[192, 168, 0, 1]		Configure IP Address of TCP Slave.
Port	UINT	502	502		Port where the slave is listening
Channel 0					ChannelConfig
Function Code	UINT	4			
Read Offset	UINT	16#0000			
Read Length	UINT	52			
Write Offset	UINT	0			
Write Length	UINT	0			
Trigger	Enumeration of USINT	CYCLIC			
Cycle Time	DWORD	100			
Error Handling	Enumeration of BOOL	Keep last value			
EnableRegisterBitMapping	BOOL				
Channel 1					ChannelConfig
Function Code	UINT	16			
Read Offset	UINT	0			
Read Length	UINT	0			
Write Offset	UINT	16#0000			
Write Length	UINT	52			
Trigger	Enumeration of USINT	CYCLIC			
Cycle Time	DWORD	100			
Error Handling	Enumeration of BOOL	Keep last value			
EnableRegisterBitMapping	BOOL				
ConfigVersion	UDINT	16#03050300	16#03050300		

Figure 6-11: Modbus TCP Server "ModbusTCPSlave Parameters"



Variable	Mapping	Channel	Address	Type	Unit	Description
		Channel 0	%IW0	ARRAY [0..51] OF WORD		Read Input Registers
		Channel 1	%QW0	ARRAY [0..51] OF WORD		Write Multiple Registers

Figure 6-12: Modbus TCP Server "ModbusTCPSlave I/O Mapping"

6.4 Handshake

Handshake could be done manually. For handshake toggle or count Message ID. With counted message ID, the failure of cable break could be recognized in Host PLC.

6.5 Communication example

1. Change control highness of the servo press kit to “Host” and fieldbus to “Modbus TCP”

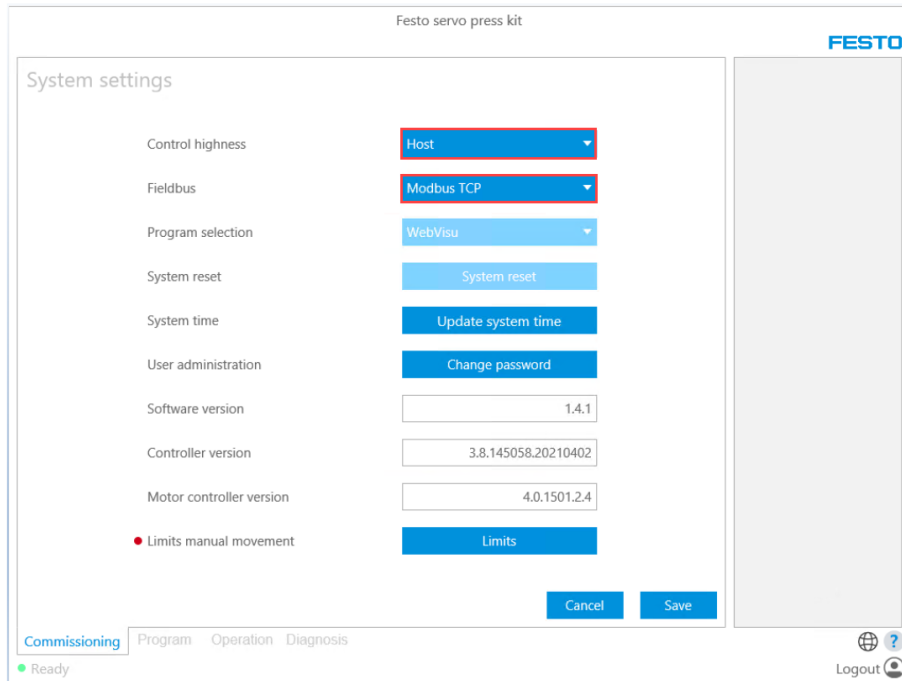


Figure 6-13: Change control highness and fieldbus

2. Send a parameter.

In this example we will send a parameter (Variable 1) by using the communication interface. According to Object directory, please see **AppNote Object directory, variable 1** has the following indexes:

Identifier	Index	Subindex	Data type	Data size	Access	Description
Variable 1	0x2300	0x01	DINT	4	RW	Value (2 decimal places)(multiplied by 100 0.01 = 100)

Table 6.1: Index table

Now you can set the request as following:

```

usiAccessType:      0x01
usiAccessOption:    0x00
uiObjectIndex:     0x2300
uiObjectSubIndex:  0x01
uiObjectDataType:  DINT
uiDataLengthToBeWritten: 4
abyDataToBeWritten: 100
  
```

By executing fb_ParameterChannel **variable 1** value will be “1”.

7 Communication EtherNet/IP

This example is to show, how to set up the connection between a client and a server based on Codesys. Until now it is tested on Codesys 3.5 SP15.

7.1 Communication task and devices

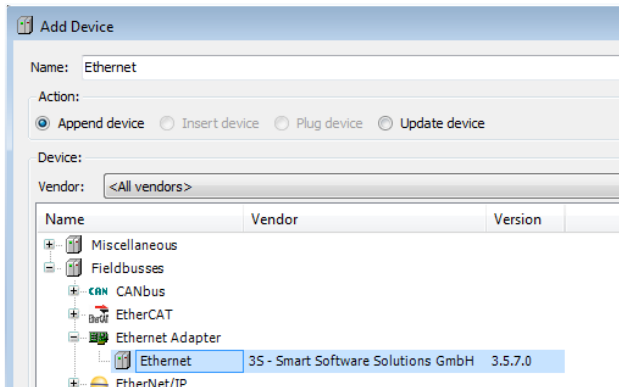


Figure 7-1: Add Ethernet

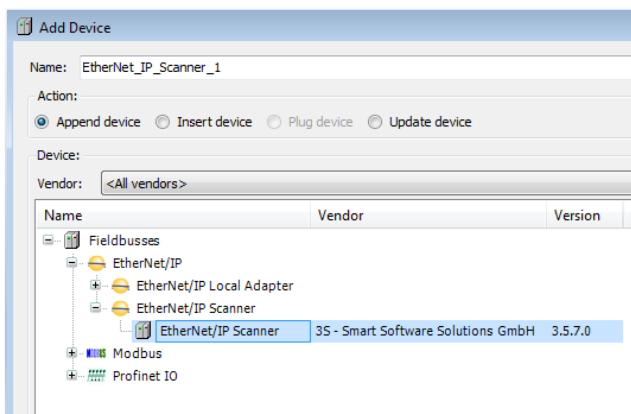


Figure 7-2: Add EtherNet/IP Scanner

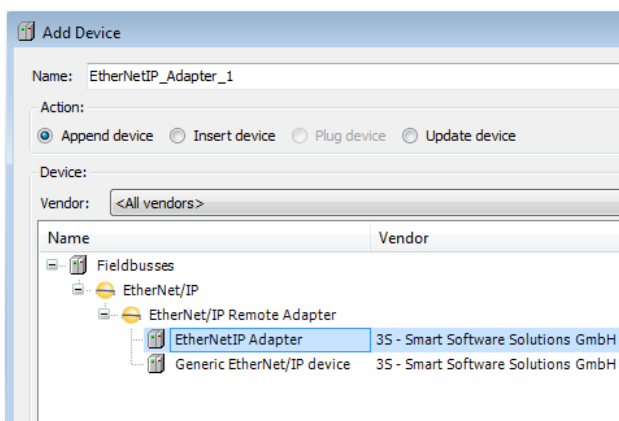


Figure 7-3: Add EtherNetIP Adapter

Two Tasks are added automatically.
The communication task between client and server has to be set to 4ms.

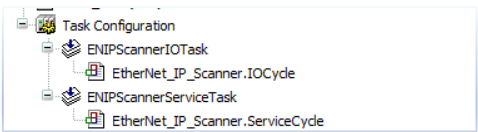


Figure 7-4: Ethernet Tasks

7.2 Example setting EtherNet/IP Client (CODESYS)

After adding the devices, client for communication with YJKP has to be set.
It is only necessary to change the settings of the EthernetIP_Adapter.

IP Address has to be set to the address of the YJKP.

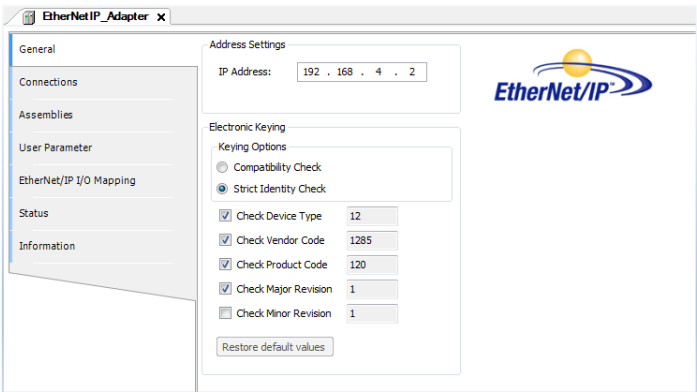


Figure 7-5: EtherNet/IP Client "General"

Edit connection parameter to 104 Byte input and output

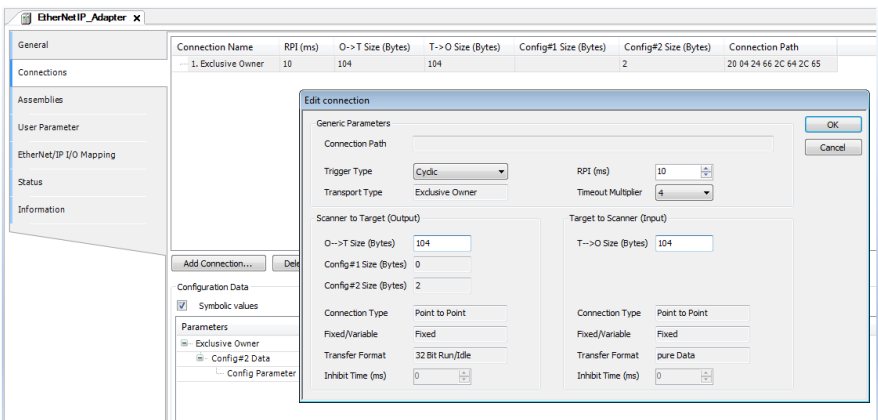


Figure 7-6: EtherNet/IP Client "Connections"

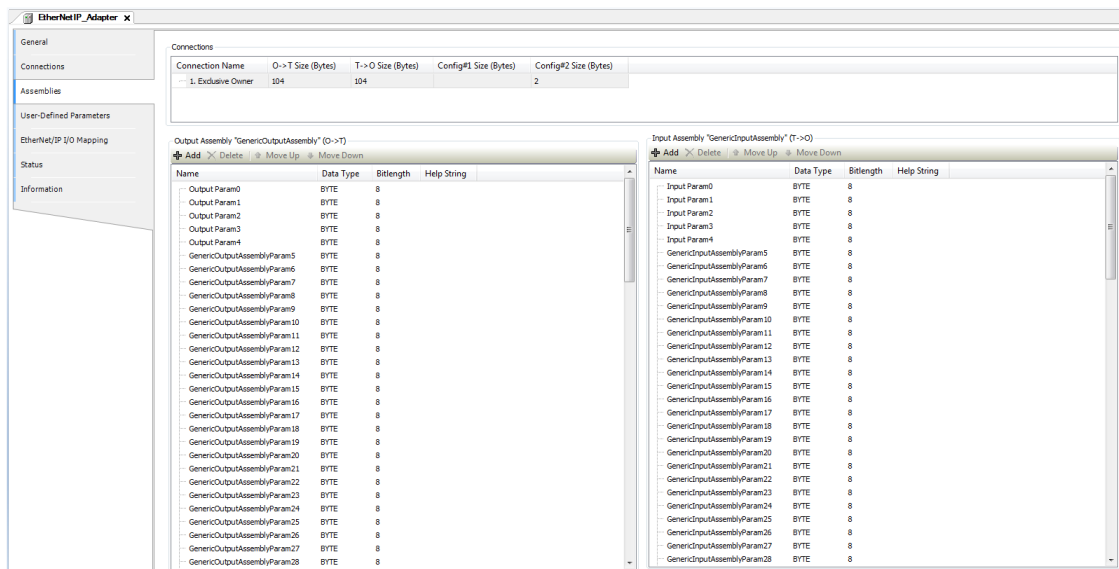


Figure 7-7: EtherNet/IP Client "Assemblies"

Variable	Mapping	Channel	Address	Type	Unit	Description
* [icon]		Input Param0	%IB0	BYTE		
* [icon]		Input Param1	%IB1	BYTE		
* [icon]		Input Param2	%IB2	BYTE		
* [icon]		Input Param3	%IB3	BYTE		
* [icon]		Input Param4	%IB4	BYTE		
* [icon]		GenericInputAssemblyParam5	%IB5	BYTE		
* [icon]		GenericInputAssemblyParam6	%IB6	BYTE		
* [icon]		GenericInputAssemblyParam7	%IB7	BYTE		
* [icon]		GenericInputAssemblyParam8	%IB8	BYTE		
* [icon]		GenericInputAssemblyParam9	%IB9	BYTE		
* [icon]		GenericInputAssemblyParam10	%IB10	BYTE		
* [icon]		GenericInputAssemblyParam11	%IB11	BYTE		
* [icon]		GenericInputAssemblyParam12	%IB12	BYTE		
* [icon]		GenericInputAssemblyParam13	%IB13	BYTE		
* [icon]		GenericInputAssemblyParam14	%IB14	BYTE		
* [icon]		GenericInputAssemblyParam15	%IB15	BYTE		
* [icon]		GenericInputAssemblyParam16	%IB16	BYTE		
* [icon]		GenericInputAssemblyParam17	%IB17	BYTE		
* [icon]		GenericInputAssemblyParam18	%IB18	BYTE		
* [icon]		GenericInputAssemblyParam19	%IB19	BYTE		
* [icon]		GenericInputAssemblyParam20	%IB20	BYTE		
* [icon]		GenericInputAssemblyParam21	%IB21	BYTE		
* [icon]		GenericInputAssemblyParam22	%IB22	BYTE		
* [icon]		GenericInputAssemblyParam23	%IB23	BYTE		
* [icon]		GenericInputAssemblyParam24	%IB24	BYTE		
* [icon]		GenericInputAssemblyParam25	%IB25	BYTE		
* [icon]		GenericInputAssemblyParam26	%IB26	BYTE		
* [icon]		GenericInputAssemblyParam27	%IB27	BYTE		
* [icon]		GenericInputAssemblyParam28	%IB28	BYTE		
* [icon]		GenericInputAssemblyParam29	%IB29	BYTE		
* [icon]		GenericInputAssemblyParam30	%IB30	BYTE		

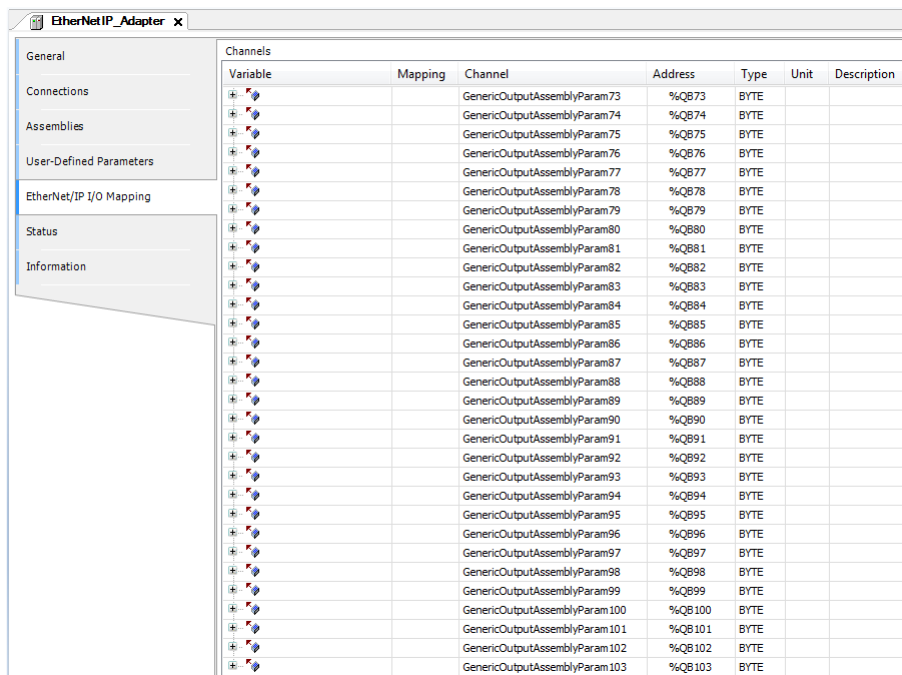
Figure 7-8: EtherNet/IP Client "EtherNet I/O Mapping (%IB0 - %IB30)"

EtherNet/IP_Adapter x							
General	Channels						
	Variable	Mapping	Channel	Address	Type	Unit	Description
Connections	*		GenericInputAssemblyParam73	%IB73	BYTE		
Assemblies	*		GenericInputAssemblyParam74	%IB74	BYTE		
User-Defined Parameters	*		GenericInputAssemblyParam75	%IB75	BYTE		
EtherNet/IP I/O Mapping	*		GenericInputAssemblyParam76	%IB76	BYTE		
Status	*		GenericInputAssemblyParam77	%IB77	BYTE		
Information	*		GenericInputAssemblyParam78	%IB78	BYTE		
	*		GenericInputAssemblyParam79	%IB79	BYTE		
	*		GenericInputAssemblyParam80	%IB80	BYTE		
	*		GenericInputAssemblyParam81	%IB81	BYTE		
	*		GenericInputAssemblyParam82	%IB82	BYTE		
	*		GenericInputAssemblyParam83	%IB83	BYTE		
	*		GenericInputAssemblyParam84	%IB84	BYTE		
	*		GenericInputAssemblyParam85	%IB85	BYTE		
	*		GenericInputAssemblyParam86	%IB86	BYTE		
	*		GenericInputAssemblyParam87	%IB87	BYTE		
	*		GenericInputAssemblyParam88	%IB88	BYTE		
	*		GenericInputAssemblyParam89	%IB89	BYTE		
	*		GenericInputAssemblyParam90	%IB90	BYTE		
	*		GenericInputAssemblyParam91	%IB91	BYTE		
	*		GenericInputAssemblyParam92	%IB92	BYTE		
	*		GenericInputAssemblyParam93	%IB93	BYTE		
	*		GenericInputAssemblyParam94	%IB94	BYTE		
	*		GenericInputAssemblyParam95	%IB95	BYTE		
	*		GenericInputAssemblyParam96	%IB96	BYTE		
	*		GenericInputAssemblyParam97	%IB97	BYTE		
	*		GenericInputAssemblyParam98	%IB98	BYTE		
	*		GenericInputAssemblyParam99	%IB99	BYTE		
	*		GenericInputAssemblyParam100	%IB100	BYTE		
	*		GenericInputAssemblyParam101	%IB101	BYTE		
	*		GenericInputAssemblyParam102	%IB102	BYTE		
	*		GenericInputAssemblyParam103	%IB103	BYTE		

Figure 7-9: EtherNet/IP Client "EtherNet I/O Mapping (%IB73 - %IB103)"

EtherNet/IP_Adapter x							
General	Channels						
	Variable	Mapping	Channel	Address	Type	Unit	Description
Connections	*		Output Param0	%QB0	BYTE		
Assemblies	*		Output Param1	%QB1	BYTE		
User-Defined Parameters	*		Output Param2	%QB2	BYTE		
EtherNet/IP I/O Mapping	*		Output Param3	%QB3	BYTE		
Status	*		Output Param4	%QB4	BYTE		
Information	*		GenericOutputAssemblyParam5	%QB5	BYTE		
	*		GenericOutputAssemblyParam6	%QB6	BYTE		
	*		GenericOutputAssemblyParam7	%QB7	BYTE		
	*		GenericOutputAssemblyParam8	%QB8	BYTE		
	*		GenericOutputAssemblyParam9	%QB9	BYTE		
	*		GenericOutputAssemblyParam10	%QB10	BYTE		
	*		GenericOutputAssemblyParam11	%QB11	BYTE		
	*		GenericOutputAssemblyParam12	%QB12	BYTE		
	*		GenericOutputAssemblyParam13	%QB13	BYTE		
	*		GenericOutputAssemblyParam14	%QB14	BYTE		
	*		GenericOutputAssemblyParam15	%QB15	BYTE		
	*		GenericOutputAssemblyParam16	%QB16	BYTE		
	*		GenericOutputAssemblyParam17	%QB17	BYTE		
	*		GenericOutputAssemblyParam18	%QB18	BYTE		
	*		GenericOutputAssemblyParam19	%QB19	BYTE		
	*		GenericOutputAssemblyParam20	%QB20	BYTE		
	*		GenericOutputAssemblyParam21	%QB21	BYTE		
	*		GenericOutputAssemblyParam22	%QB22	BYTE		
	*		GenericOutputAssemblyParam23	%QB23	BYTE		
	*		GenericOutputAssemblyParam24	%QB24	BYTE		
	*		GenericOutputAssemblyParam25	%QB25	BYTE		
	*		GenericOutputAssemblyParam26	%QB26	BYTE		
	*		GenericOutputAssemblyParam27	%QB27	BYTE		
	*		GenericOutputAssemblyParam28	%QB28	BYTE		
	*		GenericOutputAssemblyParam29	%QB29	BYTE		
	*		GenericOutputAssemblyParam30	%QB30	BYTE		

Figure 7-10: EtherNet/IP Client "EtherNet I/O Mapping (%QB0 - %QB30)"



EtherNet/IP Adapter							
Channels							
Variable	Mapping	Channel	Address	Type	Unit	Description	
*		GenericOutputAssemblyParam73	%QB73	BYTE			
*		GenericOutputAssemblyParam74	%QB74	BYTE			
*		GenericOutputAssemblyParam75	%QB75	BYTE			
*		GenericOutputAssemblyParam76	%QB76	BYTE			
*		GenericOutputAssemblyParam77	%QB77	BYTE			
*		GenericOutputAssemblyParam78	%QB78	BYTE			
*		GenericOutputAssemblyParam79	%QB79	BYTE			
*		GenericOutputAssemblyParam80	%QB80	BYTE			
*		GenericOutputAssemblyParam81	%QB81	BYTE			
*		GenericOutputAssemblyParam82	%QB82	BYTE			
*		GenericOutputAssemblyParam83	%QB83	BYTE			
*		GenericOutputAssemblyParam84	%QB84	BYTE			
*		GenericOutputAssemblyParam85	%QB85	BYTE			
*		GenericOutputAssemblyParam86	%QB86	BYTE			
*		GenericOutputAssemblyParam87	%QB87	BYTE			
*		GenericOutputAssemblyParam88	%QB88	BYTE			
*		GenericOutputAssemblyParam89	%QB89	BYTE			
*		GenericOutputAssemblyParam90	%QB90	BYTE			
*		GenericOutputAssemblyParam91	%QB91	BYTE			
*		GenericOutputAssemblyParam92	%QB92	BYTE			
*		GenericOutputAssemblyParam93	%QB93	BYTE			
*		GenericOutputAssemblyParam94	%QB94	BYTE			
*		GenericOutputAssemblyParam95	%QB95	BYTE			
*		GenericOutputAssemblyParam96	%QB96	BYTE			
*		GenericOutputAssemblyParam97	%QB97	BYTE			
*		GenericOutputAssemblyParam98	%QB98	BYTE			
*		GenericOutputAssemblyParam99	%QB99	BYTE			
*		GenericOutputAssemblyParam100	%QB100	BYTE			
*		GenericOutputAssemblyParam101	%QB101	BYTE			
*		GenericOutputAssemblyParam102	%QB102	BYTE			
*		GenericOutputAssemblyParam103	%QB103	BYTE			

Figure 7-11: EtherNet/IP Client "EtherNet I/O Mapping (%QB73 - %QB103)"

7.3 Handshake

Handshake must be done manually. For handshake toggle or count Message ID. A change in the Message ID (toggle/count) means that the request will be processed. If you don't change the Message ID (toggle/count) means that the request won't be processed. At start up of the system the Message ID is always zero.

7.4 Communication example

1. Change control highness of the servo press kit to “Host” and fieldbus to “EtherNet/IP”

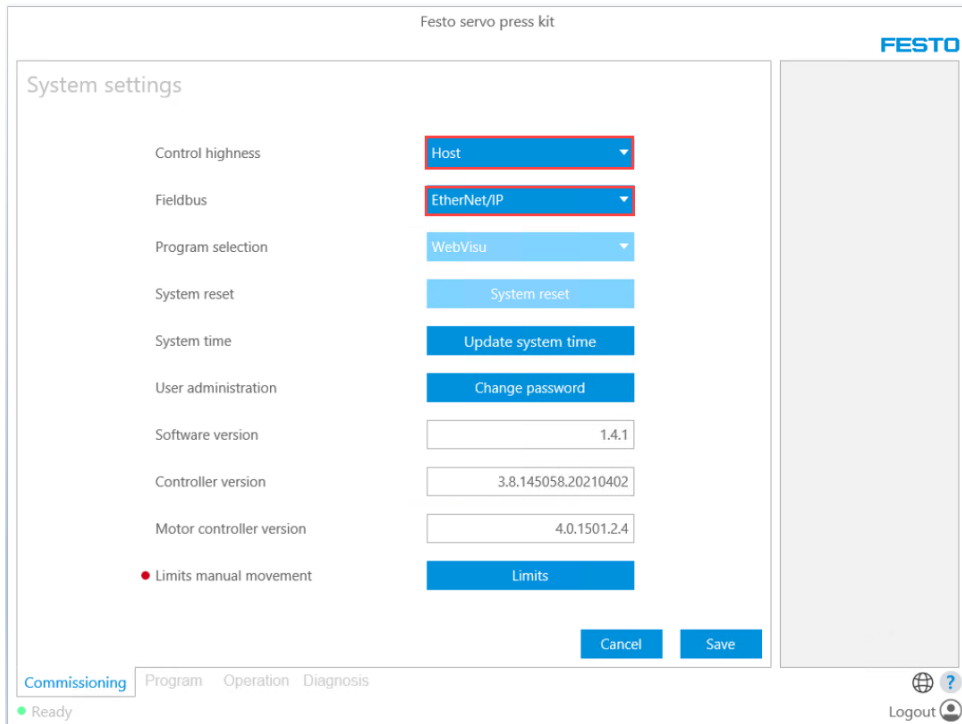


Figure 7-12: Change control highness and fieldbus

2. Send a parameter.

In this example we will send a parameter (Variable 1) by using the communication interface. According to Object directory, please see **AppNote Object directory**, **variable 1** has the following indexes:

Identifier	Index	Subindex	Data type	Data size	Access	Description
Variable 1	0x2300	0x01	DINT	4	RW	Value (2 decimal places)(multiplied by 100 0.01 = 100)

Table 7.1: Index table

Now you can set the request as following:

```

usiAccessType:      0x01
usiAccessOption:   0x00
uiObjectIndex:     0x2300
uiObjectSubIndex:  0x01
uiObjectDataType:  DINT
uiDataLengthToBeWritten: 4
abyDataToBeWritten: 100
  
```

By executing fb_ParameterChannel **variable 1** value will be “1”.

8 Communication TCP/IP

8.1 Implementation TCP/IP (CODESYS)

Communication task: 4ms

IP address: IP address of the controller CECC-X

Port: 4444

Input data: 128 Bytes

Output data: 128 Bytes

8.2 Example implementation TCP/IP Client (CODESYS)

Communication task: 4ms

Input data: 128 Bytes

Output data: 128 Bytes

8.3 Handshake

Handshake is done automatically. Client and server gets a signal if there is data received.



Note

Client must wait for reply from server before sending next message.

- Due to the nature of client / server communication. Sending message before receiving a reply from server (YJKP) will cause communication to break.

8.4 Communication example

1. Change control highness of the servo press kit to “Host” and fieldbus to “TCP/IP”

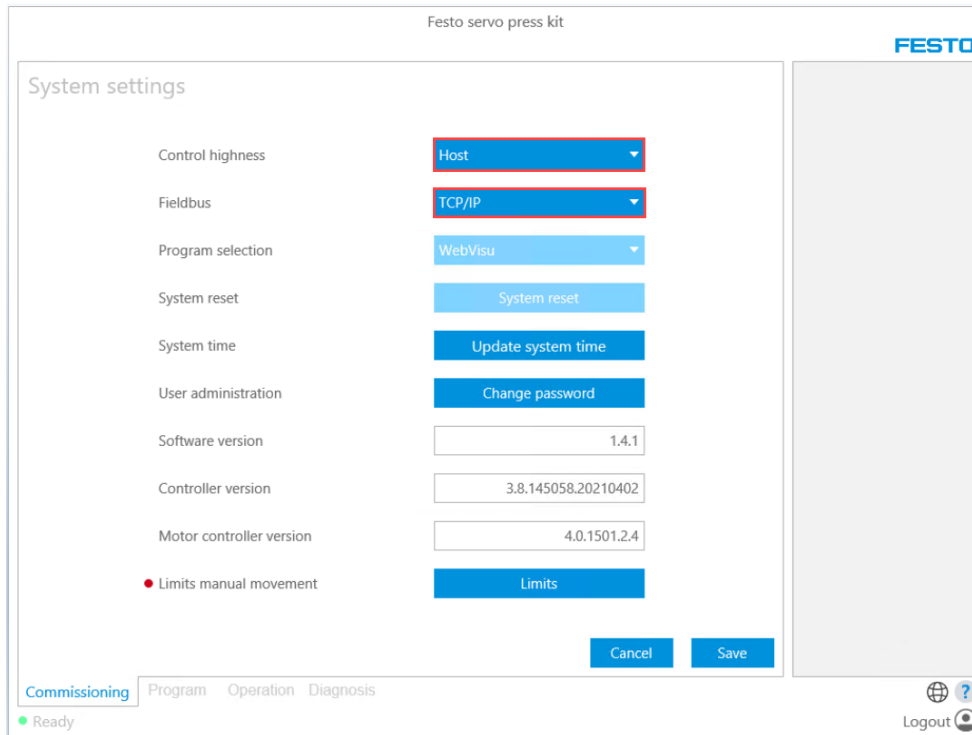


Figure 8-1: Change control highness and fieldbus

2. Send a parameter.

In this example we will send a parameter (Variable 1) by using the communication interface.

According to Object directory, please see **AppNote Object directory**, **variable 1** has the following indexes:

Identifier	Index	Subindex	Data type	Data size	Access	Description
Variable 1	0x2300	0x01	DINT	4	RW	Value (2 decimal places)(multiplied by 100 0.01 = 100)

Table 8.1: Index table

Now you can set the request as following:

```

usiAccessType:      0x01
usiAccessOption:   0x00
uiObjectIndex:     0x2300
uiObjectSubIndex:  0x01
uiObjectDataType:  DINT
uiDataLengthToBeWritten: 4
abyDataToBeWritten: 100
  
```

By executing fb_ParameterChannel **variable 1** value will be “1”.

9 Communication Profinet IO

Please follow the instructions in the application note “AppNote Servo Press Kit YJKP - Integration of host function blocks in Siemens TIA14 *”, which you can find on our support portal searching for [YJKP](#) in the tab expert knowledge.