BECKHOFF New Automation Technology

Manual | EN

TS650x

TwinCAT PLC IEC 60870-5-101, -102, -103, -104

Supplement | Communication

2023-07-05 | Version: 1.3

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Technical reference

In telecontrol applications, devices from different manufacturers have to communicate with each other. The IEC 60870-5 family defines the application-related standards IEC 60870-5-101, -102, -103 and 104, based on the five basic standards IEC 60870-5-1 (telegram formats), -2 (link layer transfer procedures), -3 (application data structures), -4 (information elements) and -5 (basic application functions). Standardised data exchange can easily be realised via the internationally standardised telecontrol protocols IEC 60870-5-101/-102/-103 for serial transfer and IEC 60870-5-104 for the TCP/IP-based transfer.

Manufacturers are not obliged to implement the complete standard in their devices. For this reason there may be incompatibilities between devices during commissioning. In order to prevent this, manufacturers offer a suitable compatibility list for each device. In this list the implemented functions are listed or marked. The compatibility list can be used to compare the required functionality of two devices in advance. Please regard the compatibility lists to the TwinCAT IEC 60870-5-10x libraries

The PLC libraries have partial two software interfaces ('Low level'- and 'High level' interface). The end application is imposed on one of these interfaces. The choice of interface depends on the requirements for the end application. Please also refer the notes in the relative product informations. In the following you will find a list of the most important product features.

TwinCAT PLC IEC 60870-5-101 Master

- Data transfer: serial;
- Product information [▶ 18];
  - System requirements [▶ 351];
  - Product components [▶ 351];
  - Installation [▶ 351];
- Available interfaces:
  - 'High level' interface: IEC 60870-5-101 Controlling Station (implemented in TcIEC870_5_101Master.Lib, include this library in your PLC project);
  - 'Low level' interface: IEC 60870-5-101 Serial Link Interface (implemented in TcIEC870_5_101Link.Lib, include this library in your PLC project);
- Possible application:
  - 'High level': Controlling station only (master);
  - 'Low level': Controlling and/or controlled station (master/slave);
- Interface documentation:
  - 'High level': TwinCAT PLC Library: IEC 60870-5-101 Controlling Station [▶ 350];
  - 'Low level': TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 397];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
  - TwinCAT PLC Library: Serial Communication;
- Examples:
  - 'High level': IEC 60870-5-101 Controlling Station [▶ 369];

- 'Low level': IEC 60870-5-101 Serial Link Interface [▶ 415];

TwinCAT PLC IEC 60870-5-101 Slave

- Data transfer: serial;
- Product information [▶ 19];
  - System requirements [▶ 372];
  - Product components [▶ 372];
  - Installation [▶ 372];
- System requirements;
- Available interfaces:
  - 'High level' interface: IEC 60870-5-101 Controlled Station (implemented in TcIEC870_5_101Slave.Lib, include this library in your PLC project);
  - 'Low level' interface: IEC 60870-5-101 Serial Link Interface (implemented in TcIEC870_5_101Link.Lib, include this library in your PLC project);
- Possible application:
  - 'High level': Controlled station only (slave);
  - 'Low level': Controlling and/or controlled station (master/slave);
- Interface documentation:
  - 'High level': TwinCAT PLC Library: IEC 60870-5-101 Controlled Station [▶ 371];
  - 'Low level': TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 397];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
  - TwinCAT PLC Library: Serial Communication;
- Examples:
  - 'High level': IEC 60870-5-101 Controlled Station [▶ 396];
  - 'Low level': IEC 60870-5-101 Serial Link Interface [▶ 415];

TwinCAT PLC IEC 60870-5-102 Master

- Data transfer: serial;
- Product information [▶ 19];
  - System requirements [▶ 418];
  - Product components [▶ 418];
  - Installation [▶ 419];
- Available interfaces:
  - 'Low level' interface: IEC 60870-5-102 Serial Link Interface (implemented in TcIEC870_5_102Link.Lib, include this library in your PLC project);
- Possible application:
  - 'Low level': Controlling station only (master);
- Interface documentation:
  - 'Low level': TwinCAT PLC Library: IEC 60870-5-102 Serial Link Interface [▶ 418];
  - TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 397];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
  - TwinCAT PLC Library: Serial Communication;
- Examples:

- 'Low level': IEC 60870-5-102 Serial Link Interface [▶ 431];

TwinCAT PLC IEC 60870-5-103 Master

- Data transfer: serial;
- Product information [▶ 20];
  - System requirements [▶ 432];
  - Product components [▶ 432];
  - Installation [▶ 433];
- Available interfaces:
  - 'Low level' interface: IEC 60870-5-103 Serial Link Interface (implemented in TcIEC870_5_103Link.Lib, include this library in your PLC project);
- Possible application:
  - 'Low level': Controlling station only (master);
- Interface documentation:
  - 'Low level': TwinCAT PLC Library: IEC 60870-5-103 Serial Link Interface [▶ 432];
  - TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 397];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
  - TwinCAT PLC Library: Serial Communication;
- Examples:
  - "Low level": IEC 60870-5-103 Serial Link Interface [▶ 442];

TwinCAT PLC IEC 60870-5-104 Master

- Data transfer: TCP/IP;
- Product information [▶ 20];
  - System requirements [▶ 482];
  - Product components [▶ 482];
  - Installation [▶ 482];
- Available interfaces:
  - 'High level' interface: IEC 60870-5-104 Controlling Station (implemented in TcIEC870_5_104Master.Lib, include this library in your PLC project);
  - 'Low level' interface: IEC 60870-5-104 Transport Interface (implemented in TcIEC870_5_104.Lib, include this library in your PLC project);
- Possible application:
  - 'High level': Controlling station only (master);
  - 'Low level': Controlling and/or controlled station (master/slave);
- Interface documentation:
  - 'High level': TwinCAT PLC Library: IEC 60870-5-104 Controlling Station [▶ 481];
  - 'Low level': TwinCAT PLC Library: IEC 60870-5-104 Transport Interface [▶ 443];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
  - TwinCAT TCP/IP Connection Server;
- Examples:
  - 'High level': IEC 60870-5-104 Controlling Station [▶ 500];
  - 'Low level': IEC 60870-5-104 Transport Interface [▶ 450];

TwinCAT PLC IEC 60870-5-104 Slave

- Data transfer: TCP/IP;
- Product information [▶ 21];
  - System requirements [▶ 452];
  - Product components [▶ 452];
  - Installation [▶ 452];
- Available interfaces:
  - 'High level' interface: IEC 60870-5-104 Controlled Station (implemented in TcIEC870_5_104Slave.Lib, include this library in your PLC project);
  - 'Low level' interface: IEC 60870-5-104 Transport Interface (implemented in TcIEC870_5_104.Lib, include this library in your PLC project);
- Possible application:
  - 'High level': Controlled station only (slave);
  - 'Low level': Controlling and/or controlled station (master/slave);
- Interface documentation:
  - 'High level': TwinCAT PLC Library: IEC 60870-5-104 Controlled Station [▶ 451];
  - 'Low level': TwinCAT PLC Library: IEC 60870-5-104 Transport Interface [▶ 443];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
  - TwinCAT TCP/IP Connection Server;
- Examples:
  - 'High level': IEC 60870-5-104 Controlled Station [▶ 479];
  - 'Low level': IEC 60870-5-104 Transport Interface [▶ 450];

3 Telegram structures

3.1 IEC 60870-5-101 telegram structure

- FT 1.2 frame format is used;
- Frames with fixed and variable block length and single control characters are used;
- Hamming distance = 4;

Frame with variable length

This frame type is used to transmit user data between controlling and controlled station

byte\bit	7	6	5	4	3	2	1	0					
0	Start byte 1 (0x68)								Header		LPCI		LPDU
1	Block length												
2	Block length (copy)												
3	Start byte 2 (0x68)												
4	RES	PRM	FCB / ACD	FCV / DFC	Function code				Control field				
5	Link address fields												
	(0, 1 or 2 octets)												
	Type identification [▶ 37]								DATA UNIT IDENTIFIER		ASDU		
	SQ	Number of objects											
	T	P/N	Cause of transmission (COT) [▶ 37]										
	Originator address (ORG, 0 or 1 octets)												
	ASDU address fields												
	(1 or 2 octets)												
	Information object address fields (IOA)								Info-object				
	(1,2 or 3 octets)												
	Object information												
n-1	Checksum								Tail		LPCI		
n	Stop byte (0x16)												

Frame with fixed length

This frame type is used for link layer services. Sometimes used as an confirmation.

byte\bit	7	6	5	4	3	2	1	0			

0	Start byte 1 (0x10)							LPCI
1	RES	PRM	FCB / ACD	FCV / DFC	Function code	Control field		
2	Link address							
	(0, 1 or 2 octets)							
n-1	Checksum							
n	Stop byte (0x16)							

Single character data

Single characters are used to confirm link layer and user data transmissions.

- 0xE5 (positive acknowledge);
- 0xA2 (negative acknowledge);

Simple samples and explanations

101substation configuration: Link address = 2 octets, COT = 1 octet (ORG address not used), ASDU address = 2 octets, IOA address = 2 octets

Sample 1

10 49 0C 00 55 16

LPDU bytes	Explanation
10	Start byte: frame with fixed length
49	Control filed: PRM-bit set (frame from primary station), function code = 9 (link status)
0C 00	Link address (2 octets) = 12 dec.
55	Checksum
16	Stop byte

Sample 2

10 0B 0C 00 17 16

LPDU bytes	Explanation
10	Start byte: frame with fixed length
0B	Control field: PRM-bit not set (frame from secondary station), function code = 11 (status of link or access demand)
0C 00	Link address (2 octets) = 12 dec.
17	Checksum
16	Stop byte

Sample 3

68 0B 0B 68 08 0C 00 65 01 0A 0C 00 00 00 05 95 16

LPDU bytes	Explanation
68	Start byte 1: frame with variable length
0B 0B	Block length and block length copy
68	Start byte 2
08	Control field: PRM-bit not set (frame from secondary station), function code = 8 (user data)
0C 00	Link address (2 octets) = 12 dec.

LPDU bytes	Explanation
65	Type identification: C_CI_NA_1 (counter interrogation)
01	Number of objects = 1
0A	Cause of transmission = 10 (activation confirmation)
0C 00	Common ASDU address (2 octets) = 12 dec.
00 00	Object address (2 octets) = 0
05	Counter interrogation request qualifier = 5 (general counter interrogation)
95	Checksum
16	Stop byte

Sample 4

68 0F 0F 68 08 0C 00 0F 01 03 0C 00 81 30 DA 16 00 00 07 DB 16

LPDU bytes	Explanation
68	Start byte 1: frame with variable length
0F 0F	Block length and block length copy
68	Start byte 2
08	Control field: PRM-bit not set (frame from secondary station), function code = 8 (user data)
0C 00	Link address (2 octets) = 12 dec.
0F	Type identification: M_IT_NA_1 (integrated total)
01	Number of objects = 1
03	Cause of transmission = 3 (spontanous)
0C 00	Common ASDU address (2 octets) = 12 dec.
81 30	Object address (2 octets)
DA 16 00 00	BCR (binary counter value)
07	Quality descriptor = 7 (sequence)
DB	Checksum
16	Stop byte

Sample 4

68 2B 2B 68 08 0C 00 0B 07 03 0C 00 10 30 BE 09 00 11 30 90 09 00 0E 30 75 00 00 28 30 25 09 00 29 30 75 00 00 0F 30 0F 0A 00 2E 30 AE 05 00 85 16

LPDU bytes	Explanation
68	Start byte 1: frame with variable length
2B 2B	Block length and block length copy
68	Start byte 2
08	Control field: PRM-bit not set (frame from secondary station), function code = 8 (user data)
0C 00	Link address (2 octets) = 12 dec.
0B	Type identification: M_ME_NB_1(measured value, scaled value)
07	Number of objects = 7
03	Cause of transmission = 3 (spontanous)
0C 00	Common ASDU address (2 octets) = 12 dec.
10 30	Object address (2 octets) of **first** information object
BE 09 00	Scaled value + QDS (quality descriptor) of **first** information object
11 30	Object address (2 octets) of **second** information object
90 09 00	Scaled value + QDS (quality descriptor) of **second** information object
0E 30	Object address (2 octets) of **third** information object
75 00 00	Scaled value + QDS (quality descriptor) of **third** information object

LPDU bytes	Explanation
28 30 25 09 00 29 30 75 00 00 0F 30 0F 0A 00 2E 30 AE 05 00	Object address + Scaled value + QDS (quality descriptor) of information object four to seven
85	Checksum
16	Stop byte

3.2 IEC 60870-5-102 telegram structure

- FT 1.2 frame format is used;
- Frames with fixed and variable block length and single control characters are used;
- Hamming distance = 4;

Frame with variable length

This frame type is used to transmit user data between controlling and controlled station

byte\ bit	7	6	5	4	3	2	1	0					
0	Start byte 1 (0x68)								Header		LPCI		LPDU
1	Block length												
2	Block length (copy)												
3	Start byte 2 (0x68)												
4	RES	PRM	FCB / ACD	FCV / DFC	Function code				Control field				
5	Link address fields												
	(0, 1 or 2 octets)												
	Type identification [▶ 427]								DATA UNIT IDENTIFIER		ASDU		
	SQ	Number of object											
	T	P/N	Cause of transmission [▶ 428]										
	ASDU address fields												
	(1 or 2 octets)												
	Record address												
	Information object address								Info-object				
	Information elements												
n-1	Checksum								Tail		LPCI		
n	Stop byte (0x16)												

Frame with fixed length

This frame type is used for link layer services. Sometimes used as an confirmation.

byte\bit	7	6	5	4	3	2	1	0	
0	Start byte 1 (0x10)								
1	RES	PRM	FCB / ACD	FCV / DFC	Function code				Control field

byte\bit	7	6	5	4	3	2	1	0	
2	Link address								
	(0, 1 or 2 octets)								
n-1	Checksum								
n	Stop byte (0x16)								

Single character data

Single characters are used to confirm link layer and user data transmissions.

- 0xE5 (positive acknowledge);

3.3 IEC 60870-5-103 telegram structure

- FT 1.2 frame format is used;
- Frames with fixed and variable block length and single control characters are used;
- Hamming distance = 4;

Frame with variable length

This frame type is used to transmit user data between controlling and controlled station

byte\ bit	7	6	5	4	3	2	1	0					
0	Start byte 1 (0x68)								Header		LPCI		LPDU
1	Block length												
2	Block length (copy)												
3	Start byte 2 (0x68)												
4	RES	PRM	FCB / ACD	FCV / DFC	Function code				Control field				
5	Link address												
6	Type identification (monitoring-direction [▶ 439], control-direction [▶ 440])								DATA UNIT IDENTIFIER		ASDU		
7	SQ	Number of object											
8	Cause of transmission (monitoring-direction [▶ 440], control-direction [▶ 441])												
9	ASDU address												
10	Function type [▶ 270]												
11	Information number [▶ 270]								Info-object				
	Information elements												
n-1	Checksum								Tail		LPCI		
n	Stop byte (0x16)												

Frame with fixed length

This frame type is used for link layer services. Sometimes used as an confirmation.

byte\bit	7	6	5	4	3	2	1	0	
0	Start byte 1 (0x10)								

byte\bit	7	6	5	4	3	2	1	0	
1	RES	PRM	FCB / ACD	FCV / DFC	Function code				Control field
2	Link address								
3	Checksum								
4	Stop byte (0x16)								

Single character data

Single characters are used to confirm link layer and user data transmissions.

- 0xE5 (positive acknowledge);

3.4 IEC 60870-5-104 telegram structure

APCI = Application Protocol Control Information

ASDU = Application Service Data Unit

APDU = Application Protocol Data Unit

Telegram format with variable length

This frame type is used to transmit user data between controlling and controlled station

byte\bit	7	6	5	4	3	2	1	0				
0	Start byte (0x68)									APCI		APDU
1	Length of the APDU (max. 253)											
2	Control field 1											
3	Control field 2											
4	Control field 3											
5	Control field 4											
6	Type identification									ASDU		
7	SQ	Number of objects										
8	T	P/N	Cause of transmission (COT)									
9	Originator address (ORG)											
10	ASDU address fields											
11	(2 octets)											
12	Information object address fields (IOA)											
13	(3 octets)											
14												
15	Object information											
...												
...												
...												
n-1												
n												

Telegram format with fixed length

byte\bit	7	6	5	4	3	2	1	0		
0	Start byte (0x68)									APCI
1	4 (Length of the APDU)									
2	Control field 1									
3	Control field 2									
4	Control field 3									
5	Control field 4									

Control field formats

Two types of control field formats: I-Format, S-Format are used to perform numbered information transfer.

The third: U-Format control field is used to perform unnumbered link layer control functions.

I-Format

byte\bit	7	6	5	4	3	2	1	0
0	Send sequence number N(S) LSB							0
1	Send sequence number N(S) MSB							
2	Receive sequence number N(R) LSB							0
3	Receive sequence number N(R) MSB							

S-Format

byte\bit	7	6	5	4	3	2	1	0
0	0						0	1
1	0							
2	Receive sequence number N(R) LSB							0
3	Receive sequence number N(R) MSB							

U-Format

byte\bit	7	6	5	4	3	2	1	0
0	TESTFR		STOPDT		STARTDT		1	1
1	0							
2	0							
3	0							

Simple samples and explanations

104 substation configuration: COT = 2 octets (includes originator address), ASDU address = 2 octets, IOA address = 3 octets

Sample 1

68 0E 4E 14 7C 00 65 01 0A 00 0C 00 00 00 00 05

LPDU bytes	Explanation
68	Start byte
0E	Length of the APDU = 14
4E	Send sequence number N(S) LSB, bit 0 = 0 => I-Format
14	Send sequence number N(S) MSB
7C	Receive sequence number N(R) LSB

LPDU bytes	Explanation
00	Receive sequence number N(R) MSB
65	Type identification: C_CI_NA_1 (counter interrogation command)
01	Number of objects = 1
0A	Cause of transmission = 10 (activation termination)
00	Originator address = 0
0C 00	Common ASDU address (2 octets) = 12 dec.
00 00 00	Object address (3 octets)
05	Counter interrogation request qualifier = 5 (general counter interrogation)

Sample 2

68 34 5A 14 7C 00 0B 07 03 00 0C 00 10 30 00 BE 09 00 11 30 00 90 09 00 0E 30 00 75 00 00 28 30 00 25 09 00 29 30 00 75 00 00 0F 30 00 0F 0A 00 2E 30 00 AE 05 00

LPDU bytes	Explanation
68	Start byte
34	Length of the APDU = 52
5A	Send sequence number N(S) LSB, bit 0 = 0 => I-Format
14	Send sequence number N(S) MSB
7C	Receive sequence number N(R) LSB
00	Receive sequence number N(R) MSB
0B	Type identification: M_ME_NB_1(measured value, scaled value)
07	Number of objects = 7
03	Cause of transmission = 3 (spontanous)
00	Originator address = 0
0C 00	Common ASDU address (2 octets) = 12 dec.
10 30 00	Object address (3 octets) of **first** information object
BE 09 00	Scaled value + QDS (quality descriptor) of **first** information object
11 30 00	Object address (3 octets) of **second** information object
90 09 00	Scaled value + QDS (quality descriptor) of **second** information object
0E 30 00	Object address (3 octets) of **third** information object
75 00 00	Scaled value + QDS (quality descriptor) of **third** information object
28 30 00 25 09 00 29 30 00 75 00 00 0F 30 00 0F 0A 00 2E 30 00 AE 05 00	Object address + Scaled value + QDS (quality descriptor) of information object four to seven

Sample 3

68 04 01 00 7E 14

LPDU bytes	Explanation
68	Start byte
04	Length of the APDU = 4
01	bits 2..7 reserved, bit 0 = 1 and bit 1 = 0 => S-Format
00	reserved
7E	Receive sequence number N(R) LSB
14	Receive sequence number N(R) MSB

4 ASDU object description

4.1 Standard IEC 60870-5-101 data types

Type	Dec	Hex	Description
ASDU_TYPEUNDEF	0	0x00	Not used
M_SP_NA_1 [▶ 37]	1	0x01	Single-point information
M_SP_TA_1 [▶ 39]	2	0x02	Single-point information with time tag
M_DP_NA_1 [▶ 43]	3	0x03	Double-point information
M_DP_TA_1 [▶ 45]	4	0x04	Double-point information with time tag
M_ST_NA_1 [▶ 49]	5	0x05	Step position information
M_ST_TA_1 [▶ 50]	6	0x06	Step position information with time tag
M_BO_NA_1 [▶ 54]	7	0x07	Bitstring of 32 bit
M_BO_TA_1 [▶ 56]	8	0x08	Bitstring of 32 bit with time tag
M_ME_NA_1 [▶ 60]	9	0x09	Measured value, normalised value
M_ME_TA_1 [▶ 61]	10	0x0A	Measured value, normalized value with time tag
M_ME_NB_1 [▶ 67]	11	0x0B	Measured value, scaled value
M_ME_TB_1 [▶ 69]	12	0x0C	Measured value, scaled value wit time tag
M_ME_NC_1 [▶ 73]	13	0x0D	Measured value, short floating point number
M_ME_TC_1 [▶ 74]	14	0x0E	Measured value, short floating point number with time tag
M_IT_NA_1 [▶ 78]	15	0x0F	Integrated totals
M_IT_TA_1 [▶ 80]	16	0x10	Integrated totals with time tag
M_EP_TA_1 [▶ 191]	17	0x11	Event of protection equipment with time tag
M_EP_TB_1 [▶ 193]	18	0x12	Packed start events of protection equipment with time tag
M_EP_TC_1 [▶ 195]	19	0x13	Packed output circuit information of protection equipment with time tag
M_PS_NA_1 [▶ 204]	20	0x14	Packed single point information with status change detection
M_ME_ND_1 [▶ 65]	21	0x15	Measured value, normalized value without quality descriptor
ASDU_TYPE_22..29	22..29	0x16..0x1D	Reserved (standard area)
M_SP_TB_1 [▶ 41]	30	0x1E	Single-point information with time tag CP56Time2a
M_DP_TB_1 [▶ 47]	31	0x1F	Double-point information with time tag CP56Time2a
M_ST_TB_1 [▶ 52]	32	0x20	Step position information with time tag CP56Time2a
M_BO_TB_1 [▶ 58]	33	0x21	Bitstring of 32 bit with time tag CP56Time2a
M_ME_TD_1 [▶ 63]	34	0x22	Measured value, normalised value with time tag CP56Time2a
M_ME_TE_1 [▶ 71]	35	0x23	Measured value, scaled value with time tag CP56Time2a
M_ME_TF_1 [▶ 76]	36	0x24	Measured value, short floating point number with time tag CP56Time2a
M_IT_TB_1 [▶ 82]	37	0x25	Integrated totals with time tag CP56Time2a
M_EP_TD_1 [▶ 198]	38	0x26	Event of protection equipment with time tag CP56Time2a
M_EP_TE_1 [▶ 200]	39	0x27	Packed start events of protection equipment with time tag CP56Time2a
M_EP_TF_1 [▶ 202]	40	0x28	Packed output circuit information of protection equipment with time tag CP56Time2a
ASDU_TYPE_41..44	41..44	0x29..0x2C	Reserved (standard area)
C_SC_NA_1 [▶ 84]	45	0x2D	Single command

Type	Dec	Hex	Description
C_DC_NA_1 [▶ 88]	46	0x2E	Double command
C_RC_NA_1 [▶ 91]	47	0x2F	Regulating step command
C_SE_NA_1 [▶ 95]	48	0x30	Set-point Command, normalised value
C_SE_NB_1 [▶ 99]	49	0x31	Set-point Command, scaled value
C_SE_NC_1 [▶ 102]	50	0x32	Set-point Command, short floating point number
C_BO_NA_1 [▶ 106]	51	0x33	Bitstring 32 bit command
ASDU_TYPE_52..57	52..57	0x34..0x39	Reserved (standard area)
C_SC_TA_1 [▶ 86]	58	0x3A	Single command with time tag CP56Time2a
C_DC_TA_1 [▶ 89]	59	0x3B	Double command with time tag CP56Time2a
C_RC_TA_1 [▶ 93]	60	0x3C	Regulating step command with time tag CP56Time2a
C_SE_TA_1 [▶ 97]	61	0x3D	Measured value, normalised value command with time tag CP56Time2a
C_SE_TB_1 [▶ 100]	62	0x3E	Measured value, scaled value command with time tag CP56Time2a
C_SE_TC_1 [▶ 104]	63	0x3F	Measured value, short floating point number command with time tag CP56Time2a
C_BO_TA_1 [▶ 108]	64	0x40	Bitstring of 32 bit command with time tag CP56Time2a
ASDU_TYPE_65..69	65..69	0x41..0x45	Reserved (standard area)
M_EI_NA_1 [▶ 113]	70	0x46	End of Initialisation
ASDU_TYPE_71..99	71..99	0x47..0x63	Reserved (standard area)
C_IC_NA_1 [▶ 125]	100	0x64	Interrogation command
C_CI_NA_1 [▶ 127]	101	0x65	Counter interrogation command
C_RD_NA_1 [▶ 131]	102	0x66	Read command
C_CS_NA_1 [▶ 124]	103	0x67	Clock synchronisation command
C_TS_NA_1 [▶ 110]	104	0x68	Test command
C_RP_NA_1 [▶ 129]	105	0x69	Reset process command
C_CD_NA_1	106	0x6A	Delay acquisition command
C_TS_TA_1	107	0x6B	Test command with time tag CP56Time2a
ASDU_TYPE_108..109	108..109	0x6C..0x6D	Reserved (standard area)
P_ME_NA_1 [▶ 206]	110	0x6E	Parameter of measured values, normalized value
P_ME_NB_1 [▶ 208]	111	0x6F	Parameter of measured values, scaled value
P_ME_NC_1 [▶ 210]	112	0x70	Parameter of measured values, short floating point number
P_AC_NA_1 [▶ 212]	113	0x71	Parameter activation
ASDU_TYPE_114..119	114..119	0x72..0x77	Reserved (standard area)
F_FR_NA_1	120	0x78	File ready
F_SR_NA_1	121	0x79	Section ready
F_SC_NA_1	122	0x7A	Call directory, select file, call file, call section
F_LS_NA_1	123	0x7B	Last section, last segment
F_FA_NA_1	124	0x7C	ACK file, ACK section
F_SG_NA_1	125	0x7D	Segment
F_DR_TA_1	126	0x7E	Directory
ASDU_TYPE_127..255	127..255	0x7F..0xFF	Reserved (user area)

4.2 Standard IEC 60870-5-102 data types

Type	Dec	Hex	Description
ASDU_TYPEUNDEF_2	0	0x00	Not used
M_SP_TA_2 [▶ 214]	1	0x01	Single-point information with time tag
M_IT_TA_2 [▶ 215]	2	0x02	Accounting integrated totals, 4 octets each
M_IT_TB_2 [▶ 217]	3	0x03	Accounting integrated totals, 3 octets each
M_IT_TC_2 [▶ 218]	4	0x04	Accounting integrated totals, 2 octets each
M_IT_TD_2 [▶ 220]	5	0x05	Periodical reset accounting integrated totals, 4 octets each
M_IT_TE_2 [▶ 221]	6	0x06	Periodical reset accounting integrated totals, 3 octets each
M_IT_TF_2 [▶ 223]	7	0x07	Periodical reset accounting integrated totals, 2 octets each
M_IT_TG_2 [▶ 225]	8	0x08	Operational integrated totals, 4 octets each
M_IT_TH_2 [▶ 226]	9	0x09	Operational integrated totals, 3 octets each
M_IT_TI_2 [▶ 228]	10	0x0A	Operational integrated totals, 2 octets each
M_IT_TK_2 [▶ 229]	11	0x0B	Periodical reset operational integrated totals, 4 octets each
M_IT_TL_2 [▶ 231]	12	0x0C	Periodical reset operational integrated totals, 3 octets each
M_IT_TM_2 [▶ 233]	13	0x0D	Periodical reset operational integrated totals, 2 octets each
	14..69	0x0E..0x45	Reserved (standard area)
M_EI_NA_2	70	0x46	End of initialization
P_MP_NA_2	71	0x47	Manufacturer and product specification of integrated total DTE
M_TI_TA_2	72	0x48	Current system time of integrated total DTE
	73..99	0x49..0x63	Reserved (standard area)
C_RD_NA_2 [▶ 132]	100	0x64	Read manufacturer and product specification
C_SP_NA_2 [▶ 134]	101	0x65	Read record of single-point information with time tag
C_SP_NB_2 [▶ 136]	102	0x66	Read record of single-point information with time tag of a selected time range
C_TI_NA_2 [▶ 138]	103	0x67	Read current system time of integrated total DTE
C_CI_NA_2 [▶ 139]	104	0x68	Read accounting integrated totals of the oldest integration period
C_CI_NB_2 [▶ 141]	105	0x69	Read accounting integrated totals of the oldest integration period and of a selected range of addresses
C_CI_NC_2 [▶ 143]	106	0x6A	Read accounting integrated totals of a specific past integration period
C_CI_ND_2 [▶ 145]	107	0x6B	Read accounting integrated totals of a specific past integration period and of a selected range of addresses
C_CI_NE_2 [▶ 147]	108	0x6C	Read periodical reset accounting integrated totals of the oldest integration period
C_CI_NF_2 [▶ 149]	109	0x6D	Read periodical reset accounting integrated totals of the oldest integration period and of a selected range of addresses
C_CI_NG_2 [▶ 151]	110	0x6E	Read periodical reset accounting integrated totals of a specific past integration period
C_CI_NH_2 [▶ 153]	111	0x6F	Read periodical reset accounting integrated totals of a specific past integration period and of a selected range of addresses
C_CI_NI_2 [▶ 155]	112	0x70	Read operational integrated totals of the oldest integration period
C_CI_NK_2 [▶ 156]	113	0x71	Read operational integrated totals of the oldest integration period and of a selected range of addresses

Type	Dec	Hex	Description
C_CI_NL_2 [▶ 158]	114	0x72	Read operational integrated totals of a specific past integration period
C_CI_NM_2 [▶ 160]	115	0x73	Read operational integrated totals of a specific past integration period and of a selected range of addresses
C_CI_NN_2 [▶ 162]	116	0x74	Read periodical reset operational integrated totals of the oldest integration period
C_CI_NO_2 [▶ 164]	117	0x75	Read periodical reset operational integrated totals of the oldest integration period and of a selected range of addresses
C_CI_NP_2 [▶ 166]	118	0x76	Read periodical reset operational integrated totals of a specific past integration period
C_CI_NQ_2 [▶ 168]	119	0x77	Read periodical reset operational integrated totals of a specific past integration period and of a selected range of addresses
C_CI_NR_2 [▶ 170]	120	0x78	Read accounting integrated totals of a specific past integration period of a selected time range and of a selected range of addresses
C_CI_NS_2 [▶ 172]	121	0x79	Read periodical reset accounting integrated totals of a specific past integration period of a selected time range and of a selected range of addresses
C_CI_NT_2 [▶ 175]	122	0x7A	Read operational integrated totals of a specific past integration period of a selected time range and of a selected range of addresses
C_CI_NU_2 [▶ 177]	123	0x7B	Read periodical reset operational integrated totals of a specific past integration period of a selected time range and of a selected range of addresses
	124..127	0x7C..0x7F	Reserved (standard area)
M_DS_TA_2	128	0x80	-
P_ME_NA_2	129	0x81	Parameters of the measuring point
M_DS_TB_2	130	0x82	-
M_CH_TA_2	131	0x83	-
C_PK_2	132	0x84	Load private key
C_TA_VC_2	133	0x85	Read tariff information (current values)
C_TA_VM_2	134	0x86	Read tariff information (stored values)
M_TA_VC_2	135	0x87	Tariff information (current values)
M_TA_VM_2	136	0x88	Tariff information (stored values)
C_TA_CP_2	137	0x89	Close accounting period
M_IB_TG_2	139	0x8B	Block of operational integrated totals (absolute values)
M_IB_TK_2	140	0x8C	Block of periodical reset operational integrated totals (increment values)
C_RM_NA_2	141	0x8D	Read configuration data of the meter device
M_RM_NA_2	142	0x8E	Configuration of the meter device
C_MR_NA_2	143	0x8F	Change configuration data of the meter device
C_PC_NA_2	144	0x90	-
M_PC_NA_2	145	0x91	-
C_MC_NA_2	146	0x92	-
C_DF_NA_2	147	0x93	-
M_DF_NA_2	148	0x94	-
C_MF_NA_2	149	0x95	-
	150..179	0x96..0xB3	Reserved
C_DS_TA_2	180	0xB4	-

Type	Dec	Hex	Description
C_CS_TA_2	181	0xB5	Change date and time (Time syncronization)
C_PI_NA_2	182	0xB6	Read parameters of the measuring point
C_AC_NA_2	183	0xB7	Start session and send access key
C_DS_TB_2	184	0xB8	-
C_CH_TA_2	185	0xB9	-
C_MH_TA_2	186	0xBA	-
C_FS_NA_2	187	0xBB	Finish session
C_MP_NA_2	188	0xBC	-
C_CB_NT_2	189	0xBD	Read a block of operational integrated totals of a period and a selected address
C_CB_UN_2	190	0xBE	Read a block of periodical reset operational integrated totals of a period and a selected address
	191..255	0xBF..0xFF	Reserved

4.3 Standard IEC 60870-5-103 data types

Monitoring direction	Control direction	Dec	Hex	Description
M_TYPEUNDEF_3	C_TYPEUNDEF_3	0	0x00	Not used
M_TTM_TA_3 [▶ 234]	-	1	0x01	Time-tagged message
M_TMR_TA_3 [▶ 236]	-	2	0x02	Time-tagged message with relative time
M_MEI_NA_3 [▶ 239]	-	3	0x03	Measurands I
M_TME_TA_3 [▶ 241]	-	4	0x04	Time-tagged measurands with relative time
M_IRC_NA_3 [▶ 115]	-	5	0x05	Identification
M_SYN_TA_3 [▶ 119]	C_SYN_TA_3 [▶ 179]	6	0x06	Time synchronization
-	C_IGI_NA_3 [▶ 181]	7	0x07	Initialization of general interrogation
M_TGI_NA_3 [▶ 117]	-	8	0x08	General interrogation
M_MEII_NA_3 [▶ 243]	-	9	0x09	Measurands II
M_GD_XA_3 [▶ 122]	C_GD_NA_3 [▶ 182]	10	0x0A	Generic data
M_GI_XA_3 [▶ 120]	-	11	0x0B	Generic identification
-	-	12..19	0x0C..0x13	Reserved (standard area)
-	C_GRC_NA_3 [▶ 184]	20	0x14	General command
-	C_GC_NA_3 [▶ 186]	21	0x15	Generic command
-	-	22	0x16	Reserved (standard area)

Monitoring direction	Control direction	Dec	Hex	Description
M_LRD_TA_3 [▶ 247]	-	23	0x17	List of recorded disturbances
-	C_ODT_NA_3 [▶ 187]	24	0x18	Order for disturbance data transmission
-	C_ADT_NA_3 [▶ 189]	25	0x19	Acknowledgement for disturbance data transmission
M_RTD_TA_3 [▶ 248]	-	26	0x1A	Ready for transmission of disturbance data
M_RTC_NA_3 [▶ 250]	-	27	0x1B	Ready for transmission of channel
M_RTT_NA_3 [▶ 252]	-	28	0x1C	Ready for transmission of tags
M_TOT_NA_3 [▶ 254]	-	29	0x1D	Transmission of tags
M_TOV_NA_3 [▶ 255]	-	30	0x1E	Transmission of disturbance values
M_EOT_NA_3 [▶ 258]	-	31	0x1F	End of transmission
-	-	32..255	0x1F..0xFF	Reserved (user defined area)

4.4 Standard IEC 60870-5-104 data types

Type	Dec	Hex	Description
ASDU_TYPEUNDEF	0	0x00	Not used
M_SP_NA_1	1	0x01	Single-point information
M_SP_TA_1	2	0x02	Single-point information with time tag
M_DP_NA_1	3	0x03	Double-point information
M_DP_TA_1	4	0x04	Double-point information with time tag
M_ST_NA_1 [▶ 266]	5	0x05	Step position information
M_ST_TA_1	6	0x06	Step position information with time tag
M_BO_NA_1	7	0x07	Bitstring of 32 bit
M_BO_TA_1	8	0x08	Bitstring of 32 bit with time tag
M_ME_NA_1	9	0x09	Measured value, normalised value
M_ME_TA_1	10	0x0A	Measured value, normalized value with time tag
M_ME_NB_1	11	0x0B	Measured value, scaled value
M_ME_TB_1	12	0x0C	Measured value, scaled value wit time tag
M_ME_NC_1	13	0x0D	Measured value, short floating point number
M_ME_TC_1	14	0x0E	Measured value, short floating point number with time tag
M_IT_NA_1	15	0x0F	Integrated totals
M_IT_TA_1	16	0x10	Integrated totals with time tag
M_EP_TA_1	17	0x11	Event of protection equipment with time tag
M_EP_TB_1	18	0x12	Packed start events of protection equipment with time tag
M_EP_TC_1	19	0x13	Packed output circuit information of protection equipment with time tag

Type	Dec	Hex	Description
M_PS_NA_1	20	0x14	Packed single point information with status change detection
M_ME_ND_1	21	0x15	Measured value, normalized value without quality descriptor
ASDU_TYPE_22..29	22..29	0x16..0x1D	Reserved (standard area)
M_SP_TB_1	30	0x1E	Single-point information with time tag CP56Time2a
M_DP_TB_1	31	0x1F	Double-point information with time tag CP56Time2a
M_ST_TB_1 [▶ 268]	32	0x20	Step position information with time tag CP56Time2a
M_BO_TB_1	33	0x21	Bitstring of 32 bit with time tag CP56Time2a
M_ME_TD_1	34	0x22	Measured value, normalised value with time tag CP56Time2a
M_ME_TE_1	35	0x23	Measured value, scaled value with time tag CP56Time2a
M_ME_TF_1	36	0x24	Measured value, short floating point number with time tag CP56Time2a
M_IT_TB_1	37	0x25	Integrated totals with time tag CP56Time2a
M_EP_TD_1	38	0x26	Event of protection equipment with time tag CP56Time2a
M_EP_TE_1	39	0x27	Packed start events of protection equipment with time tag CP56Time2a
M_EP_TF_1	40	0x28	Packed output circuit information of protection equipment with time tag CP56Time2a
ASDU_TYPE_41..44	41..44	0x29..0x2C	Reserved (standard area)
C_SC_NA_1	45	0x2D	Single command
C_DC_NA_1	46	0x2E	Double command
C_RC_NA_1	47	0x2F	Regulating step command
C_SE_NA_1	48	0x30	Set-point Command, normalised value
C_SE_NB_1	49	0x31	Set-point Command, scaled value
C_SE_NC_1	50	0x32	Set-point Command, short floating point number
C_BO_NA_1	51	0x33	Bitstring 32 bit command
ASDU_TYPE_52..57	52..57	0x34..0x39	Reserved (standard area)
C_SC_TA_1	58	0x3A	Single command with time tag CP56Time2a
C_DC_TA_1	59	0x3B	Double command with time tag CP56Time2a
C_RC_TA_1	60	0x3C	Regulating step command with time tag CP56Time2a
C_SE_TA_1	61	0x3D	Measured value, normalised value command with time tag CP56Time2a
C_SE_TB_1	62	0x3E	Measured value, scaled value command with time tag CP56Time2a
C_SE_TC_1	63	0x3F	Measured value, short floating point number command with time tag CP56Time2a
C_BO_TA_1	64	0x40	Bitstring of 32 bit command with time tag CP56Time2a
ASDU_TYPE_65..69	65..69	0x41..0x45	Reserved (standard area)
M_EI_NA_1	70	0x46	End of Initialisation
ASDU_TYPE_71..99	71..99	0x47..0x63	Reserved (standard area)
C_IC_NA_1	100	0x64	Interrogation command
C_CI_NA_1	101	0x65	Counter interrogation command
C_RD_NA_1	102	0x66	Read command
C_CS_NA_1	103	0x67	Clock synchronisation command

Type	Dec	Hex	Description
C_TS_NA_1	104	0x68	Test command
C_RP_NA_1	105	0x69	Reset process command
C_CD_NA_1	106	0x6A	Delay acquisition command
C_TS_TA_1 [▶ 112]	107	0x6B	Test command with time tag CP56Time2a
ASDU_TYPE_108..109	108..109	0x6C..0x6D	Reserved (standard area)
P_ME_NA_1	110	0x6E	Parameter of measured values, normalized value
P_ME_NB_1	111	0x6F	Parameter of measured values, scaled value
P_ME_NC_1	112	0x70	Parameter of measured values, short floating point number
P_AC_NA_1	113	0x71	Parameter activation
ASDU_TYPE_114..119	114..119	0x72..0x77	Reserved (standard area)
F_FR_NA_1	120	0x78	File ready
F_SR_NA_1	121	0x79	Section ready
F_SC_NA_1	122	0x7A	Call directory, select file, call file, call section
F_LS_NA_1	123	0x7B	Last section, last segment
F_FA_NA_1	124	0x7C	ACK file, ACK section
F_SG_NA_1	125	0x7D	Segment
F_DR_TA_1	126	0x7E	Directory
ASDU_TYPE_127..255	127..255	0x7F..0xFF	Reserved (user area)

4.5 Single-point information

4.5.1 M_SP_NA_1

Single-point information without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x01 (1)	= M_SP_NA_1	Type identification [▶ 37]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 1								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	SPI [▶ 37]	SIQ = Single-point information with quality descriptor
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.5.2 M_SP_TA_1

Single-point information with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x02 (2)	= M_SP_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= 1									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 4									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	SPI [▶ 37]	SIQ = Single-point information with quality descriptor
			|---	data[1..3] =	CP24Time2a [▶ 37]								Three octets binary time tag
			|---	data[4..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.5.3 M_SP_TB_1

Single-point information with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x1E (30)	= M_SP_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 8									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	SPI [▶ 37]	SIQ = Single-point information with quality descriptor
			|---	data[1..7] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.6 Double-point information

4.6.1 M_DP_NA_1

Double-point information without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x03 (3)	= M_DP_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	DPI [▶ 37]	DIQ = Double-point information with quality descriptor
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =								Reserved

4.6.2 M_DP_TA_1

Double-point information with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x04 (4)	= M_DP_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 4									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	DPI [▶ 37]		DIQ = Double-point information with quality descriptor
			|---	data[1..3] =	CP24Time2a [▶ 37]								Three octet binary time tag

			|---	data[4..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.6.3 M_DP_TB_1

Double-point information with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x1F (31)	= M_DP_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 8									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	DPI [▶ 37]		DIQ = Double-point information with quality descriptor
			|---	data[1..7] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.7 Step position information

4.7.1 M_ST_NA_1

Step position information without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x05 (5)	= M_ST_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu

|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 2								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	VTI [▶ 37]								Value with transient state indication
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.7.2 M_ST_TA_1

Step position information with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x06 (6)	= M_ST_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]

	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 5								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	VTI [▶ 37]								Value with transient state indication
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[2..4] =	CP24Time2a [▶ 37]								Three octets binary time tag
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.7.3 M_ST_TB_1

Step position information with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x20 (32)	= M_ST_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 9								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	VTI [▶ 37]								Value with transient state indication
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[2..8] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[9..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.8 Bitstring of 32 bits

4.8.1 M_BO_NA_1

Bitstring of 32 bits without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x07 (7)	= M_BO_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 5								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	BSI [▶ 37]								Binary state information
			|---	data[4] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS [▶ 265]
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.8.2 M_BO_TA_1

Bitstring of 32 bits with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x08 (8)	= M_BO_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 8									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	BSI [▶ 37]								Binary state information

			|---	data[4] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS [▶ 265]
			|---	data[5..7] =	CP24Time2a [▶ 37]								Three octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.8.3 M_BO_TB_1

Bitstring of 32 bits with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x21 (33)	= M_BO_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	BSI [▶ 37]								Binary state information
			|---	data[4] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS [▶ 265]
			|---	data[5..11] =	CP56Time2a [▶ 37]								Seven octets binary time tag

			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.9 Measured value, normalized value

4.9.1 M_ME_NA_1

Measured value, normalized value without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x09 (9)	= M_ME_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 3								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	NVA [▶ 37]								Normalized value
			|---	data[2] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[3..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.9.2 M_ME_TA_1

Measured value, normalized value with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x0A (10)	= M_ME_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]

|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 6								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	NVA [▶ 37]								Normalized value
			|---	data[2] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[3..5] =	CP24Time2a [▶ 37]								Three octets binary time tag
			|---	data[6..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.9.3 M_ME_TD_1

Measured value, normalized value with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x22 (34)	= M_ME_TD_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]

	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 10								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	NVA [▶ 37]								Normalized value;
			|---	data[2] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[3..9] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.9.4 M_ME_ND_1

Measured value, normalized value without quality descriptor.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x15 (21)	= M_ME_ND_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 2								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	NVA [▶ 37]								Normalized value
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.10 Measured value, scaled value

4.10.1 M_ME_NB_1

Measured value, scaled value without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x0B (11)	= M_ME_NB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 3									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	SVA [▶ 37]								Scaled value

			|---	data[2] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[3..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.10.2 M_ME_TB_1

Measured value, scaled value with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x0C (12)	= M_ME_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT										Cause of transmission [▶ 37]
|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 6								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	SVA [▶ 37]								Scaled value
			|---	data[2] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[3..5] =	CP24Time2a [▶ 37]								Three octets binary time tag

			|---	data[6..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.10.3 M_ME_TE_1

Measured value, scaled value with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x23 (35)	= M_ME_TE_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 10									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	SVA [▶ 37]								Scaled value
			|---	data[2] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[3..9] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.11 Measured value, short floating point value

4.11.1 M_ME_NC_1

Measured value, short floating point value without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x0D (13)	= M_ME_NC_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu

|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 5								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 37]								Short floating point value
			|---	data[4] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.11.2 M_ME_TC_1

Measured value, short floating point value with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x0E (14)	= M_ME_TC_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]

	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 8								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 37]								Short floating point value
			|---	data[4] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[5..7] =	CP24Time2a [▶ 37]								Three octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.11.3 M_ME_TF_1

Measured value, short floating point value with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x24 (36)	= M_ME_TF_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 12								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 37]								Short floating point value
			|---	data[4] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[5..11] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.12 Integrated totals

4.12.1 M_IT_NA_1

Integrated total without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x0F (15)	= M_IT_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 5								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	BCR [▶ 37]								Binary counter reading
			|---	data[4] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.12.2 M_IT_TA_1

Integrated total with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x10 (16)	= M_IT_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= 1									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 8									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0..3] =	BCR [▶ 37]				Binary counter reading
			|---	data[4] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence	Quality descriptor
			|---	data[5..7] =	CP24Time2a [▶ 37]				Three octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =					Reserved

4.12.3 M_IT_TB_1

Integrated total with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x25 (37)	= M_IT_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	BCR [▶ 37]								Binary counter reading

			|---	data[4] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence	Quality descriptor
			|---	data[5..11] =	CP56Time2a [▶ 37]				Seven octets binary time tag
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =					Reserved

4.13 Single command

4.13.1 C_SC_NA_1

Single command without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x2D (45)	= C_SC_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN										Positive/ negative confirmation/ activation
|	|---	eCOT										Cause of transmission [▶ 37]
|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 1								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	S/E [▶ 37]	QU [▶ 37]					0	SCS [▶ 37]	SCO = Single command

			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.13.2 C_SC_TA_1

Single command with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x3A (58)	= C_SC_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 8									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	S/E [▶ 37]	QU [▶ 37]					0	SCS [▶ 37]	SCO = Single command
			|---	data[1..7] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.14 Double command

4.14.1 C_DC_NA_1

Double command without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x2E (46)	= C_DC_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu

|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 1								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	S/E [▶ 37]	QU [▶ 37]					DCS [▶ 37]		DCO = Double command
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.14.2 C_DC_TA_1

Double command with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x3B (59)	= C_DC_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 8								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	S/E [▶ 37]	QU [▶ 37]					DCS [▶ 37]		DCO = Double command
			|---	data[1..7] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.15 Regulating step command

4.15.1 C_RC_NA_1

Regulating step command without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x2F (47)	= C_RC_NA_1	Type identification [▶ 37]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 1								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	S/E [▶ 37]	QU [▶ 37]					RCS [▶ 37]		RCO = Regulating step command
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.15.2 C_RC_TA_1

Regulating step command with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x3C (60)	= C_RC_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN										Positive/ negative confirmation/ activation
|	|---	eCOT										Cause of transmission [▶ 37]
|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 8								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	S/E [▶ 37]	QU [▶ 37]					RCS [▶ 37]		RCO = Regulating step command

			|---	data[1..7] =	CP56Time2a [▶ 37]	Seven octets binary time tag
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.16 Set-point command, normalized value

4.16.1 C_SE_NA_1

Set-point command, normalized value without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x30 (48)	= C_SE_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 3									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	NVA [▶ 37]								Normalized value
			|---	data[2] =	S/E [▶ 37]	QL [▶ 37]							QOS = Qualifier of set-point command

			|---	data[3..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.16.2 C_SE_TA_1

Set-point command, normalized value with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x3D (61)	= C_SE_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 10									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	NVA [▶ 37]								Normalized value
			|---	data[2] =	S/E [▶ 37]	QL [▶ 37]							QOS = Qualifier of command
			|---	data[3..9] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.17 Set-point command, scaled value

4.17.1 C_SE_NB_1

Set-point command, scaled value without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x31 (49)	= C_SE_NB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu

|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 3									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	SVA [▶ 37]								Scaled value
			|---	data[2] =	S/E [▶ 37]	QL [▶ 37]							QOS = Qualifier of set-point command
			|---	data[3..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.17.2 C_SE_TB_1

Set-point command, scaled value with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x3E (62)	= C_SE_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]

	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 10								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	SVA [▶ 37]								Scaled value
			|---	data[2] =	S/E [▶ 37]	QL [▶ 37]							QOS = Qualifier of command
			|---	data[3..9] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.18 Set-point command, short floating value

4.18.1 C_SE_NC_1

Set-point command, short floating point value without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x32 (50)	= C_SE_NC_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 5								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 37]								Short floating point value
			|---	data[4] =	S/E [▶ 37]	QL [▶ 37]							QOS = Qualifier of set-point command
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.18.2 C_SE_TC_1

Set-point command, short floating point value with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x3F (63)	= C_SE_TC_1	Type identification [▶ 37]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 12								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 37]								Short floating point value
			|---	data[4] =	S/E [▶ 37]	QL [▶ 37]							QOS = Qualifier of command
			|---	data[5..11] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.19 Bitstring command

4.19.1 C_BO_NA_1

Bitstring of 32 bits without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x33 (51)	= C_BO_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 4									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	BSI [▶ 37]								Binary state information
			|---	data[4..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.19.2 C_BO_TA_1

Bitstring of 32 bits with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x40 (64)	= C_BO_TA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 11									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	BSI [▶ 37]								Binary state information

			|---	data[4..10] =	CP56Time2a [▶ 37]	Seven octets binary time tag
			|---	data[11..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.20 Test command

4.20.1 C_TS_NA_1

Test command without time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x68 (104)	= C_TS_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr		= 0									Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 2									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] = 0xAA	1	0	1	0	1	0	1	0	FBP = Fixed test pattern
			|---	data[1] = 0x55	0	1	0	1	0	1	0	1	
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.20.2 C_TS_TA_1

Test command with CP56Time2a time tag.

- obj					ASDU object
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER
|	|---	eType		= C_TS_TA_1	Type identification
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class
|---	- info				INFORMATION OBJECT

	|---	objAddr											Information object address
	|---	stream											Information element/ object data
		|---	length	= 9									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	TSC								Test counter
			|---	data[2..8] =	CP56Time2a								Seven octets binary time tag
			|---	data[9..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.21 System information in monitor direction

4.21.1 M_EI_NA_1

End of initialization.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x46 (70)	= M_EI_NA_1	Type identification [▶ 37]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr		= 0	Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 1								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	LPC [▶ 37]	COI [▶ 37]							Cause of initialization
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.21.2 M_IRC_NA_3

Identification.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x5 (5)	= M_IRC_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects

|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 440]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc											Function code/ type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 13									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	COL [▶ 260]	Compatibility level
			|---	data[1..8] =	ASC [▶ 260]	ASCII characters
			|---	data[9..12] = 0x20 if not used	free	Internal vendor identification
			|---	data[13..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.21.3 M_TGI_NA_3

Termination of general interrogation.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x8 (8)	= M_TGI_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	eCOT											Cause of transmission [▶ 440]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc		= GLB									Function code/ type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	SCN [▶ 263]								Scan number

			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.21.4 M_SYN_TA_3

Time synchronisation.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x6 (6)	= M_SYN_TA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu

|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc		= GLB									Function code/ type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 7									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..6] =	CP56Time2a [▶ 344]								Seven octets binary time format
			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.21.5 M_GI_XA_3

Generic identification.

- obj													ASDU object [▶ 438]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0xB (11)	= M_GI_XA_3									Type identification [▶ 439]
|	|---	bSQ		= TRUE									Sequence of information objects
|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 440]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc		GEN									Function code/type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= variable									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	RII [▶ 263]								Return information identifier
			|---	data[1] =	GROUP [▶ 261]								GIN = Generic identification number
			|---	data[2] =	ENTRY [▶ 261]								
			|---	data[3] =	CONT [▶ 260]	COUNT [▶ 260]	NO [▶ 262]						NDE = Number of descriptive elements

			|---	data[4] =	KOD [▶ 261]	Kind of description	Element 1
			|---	data[5] =	DATATYPE [▶ 260]	GDD = Generic data description	
			|---	data[6] =	DATASIZE [▶ 260]		
			|---	data[7] =	CONT [▶ 260] NUMBER		
			|---	data[8..8+(DATASIZE*NUMBER)] =	GID	Generic identification data	
			|---	data[...IEC870_MAX_ASDU_DATA_BYTE] =			Element 2..i

4.21.6 M_GD_XA_3

Generic data.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0xA (10)	= M_GD_XA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]

|---	- info												INFORMATION OBJECT [▶ 438]	
	|---	fc		GEN									Function code/type [▶ 270]	
	|---	n											Information number [▶ 270]	
	|---	stream											Information element/ object data [▶ 316]	
		|---	length	= variable										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	RII [▶ 263]								Return information identifier	
			|---	data[1] =	CONT [▶ 260]	COUNT [▶ 260]	NO [▶ 262]						NGD = Number of generic data sets	
			|---	data[2] =	GROUP [▶ 261]								GIN = Generic identification number	Dataset 1
			|---	data[3] =	ENTRY [▶ 261]									
			|---	data[4] =	KOD [▶ 261]								Kind of description	
			|---	data[5] =	DATATYPE [▶ 260]								GDD = Generic data description	
			|---	data[6] =	DATASIZE [▶ 260]									
			|---	data[7] =	CONT [▶ 260]	NUMBER								
			|---	data[8..8+(DATASIZE*NUMBER)] =	GID								Generic identification data	

			|---	data[..IEC870_MAX_ASDU_DATA_BYTE] =			Dataset 2..i

4.22 System information in control direction

4.22.1 C_CS_NA_1

Clock synchronisation command

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x67 (103)	= C_CS_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr		= 0									Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 7									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..6] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.2 C_IC_NA_1

Interrogation command.

- obj													ASDU object [▶ 37]

|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x64 (100)	= C_IC_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]

|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr		= 0									Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	QOI [▶ 37]								Qualifier of interrogation
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.3 C_CI_NA_1

Counter interrogation command.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x65 (101)	= C_CI_NA_1	Type identification [▶ 37]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr		= 0	Information object address

	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	FRZ [▶ 37]		RQT [▶ 37]						QCC = Qualifier of counter interrogation
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.4 C_RP_NA_1

Reset process command.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x69 (105)	= C_RP_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr		= 0									Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	QRP [▶ 37]								Qualifier of reset process

			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.5 C_RD_NA_1

Read command.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x66 (102)	= C_RD_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.6 C_RD_NA_2

Read manufacturer and product specification.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x64 (100)	= C_RD_NA_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr		= 0									Record address
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.7 C_SP_NA_2

Read record of single-point information with time tag.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x65 (101)	= C_SP_NA_2	Type identification [▶ 427]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]
	|---	rcdAddr			Record address

	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.8 C_SP_NB_2

Read record of single-point information with time tag of selected time range.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x66 (102)	= C_SP_NB_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 10									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 430]								Five octets binary time tag (from)

			|---	data[5..9] =	CP40Time2a [▶ 430]	Five octets binary time tag (until)
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.9 C_TI_NA_2

Read current system time of integrated total data terminal equipment.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x67 (103)	= C_TI_NA_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr		= 0									Record address
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.10 C_CI_NA_2

Read accounting integrated totals of the oldest integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x68 (104)	= C_CI_NA_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.11 C_CI_NB_2

Read accounting integrated totals of the oldest integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0x69 (105)	= C_CI_NB_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]

	|---	rcdAddr										Record address of accounting period
	|---	stream										Information element/ object data [▶ 215]
		|---	length	= 2								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.12 C_CI_NC_2

Read accounting integrated totals of a specific past integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0x6A (106)	= C_CI_NC_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 430]								Five octets binary time tag
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.13 C_CI_ND_2

Read accounting integrated totals of a specific past integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x6B (107)	= C_CI_ND_2	Type identification [▶ 427]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]
	|---	rcdAddr			Record address of accounting period

	|---	stream										Information element/ object data [▶ 215]
		|---	length	= 7								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..6] =	CP40Time2a [▶ 430]								Five octets binary time tag
			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.14 C_CI_NE_2

Read periodically reset accounting integrated totals of the oldest integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0x6C (108)	= C_CI_NE_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.15 C_CI_NF_2

Read periodically reset accounting integrated totals of the oldest integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x6D (109)	= C_CI_NF_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= 1									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 2									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =		Integrated total address (from)
			|---	data[1] =		Integrated total address (to)
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.16 C_CI_NG_2

Read periodically reset accounting integrated totals of a specific past integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x6E (110)	= C_CI_NG_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 430]								Five octets binary time tag

			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.17 C_CI_NH_2

Read periodically reset accounting integrated totals of a specific past integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x6F (111)	= C_CI_NH_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 7									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)

			|---	data[2..6] =	CP40Time2a [▶ 430]	Five octets binary time tag
			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.18 C_CI_NI_2

Read operational integrated totals of the oldest integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x70 (112)	= C_CI_NI_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.19 C_CI_NK_2

Read operational integrated totals of the oldest integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x71 (113)	= C_CI_NK_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 2									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.20 C_CI_NL_2

Read operational integrated totals of a specific past integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x72 (114)	= C_CI_NL_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 430]								Five octets binary time tag
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.21 C_CI_NM_2

Read operational integrated totals of a specific past integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0x73 (115)	= C_CI_NM_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 7									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..6] =	CP40Time2a [▶ 430]								Five octets binary time tag
			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.22 C_CI_NN_2

Read periodically reset operational integrated totals of the oldest integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x74 (116)	= C_CI_NN_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 0									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.23 C_CI_NO_2

Read periodically reset operational integrated totals of the oldest integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0x75 (117)	= C_CI_NO_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 2									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.24 C_CI_NP_2

Read periodically reset operational integrated totals of a specific past integration period.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0x76 (118)	= C_CI_NP_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 430]								Five octets binary time tag
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.25 C_CI_NQ_2

Read periodically reset operational integrated totals of a specific past integration period and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x77 (119)	= C_CI_NQ_2	Type identification [▶ 427]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]
	|---	rcdAddr			Record address of accounting period

	|---	stream										Information element/ object data [▶ 215]
		|---	length	= 7								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..6] =	CP40Time2a [▶ 430]								Five octets binary time tag
			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.26 C_CI_NR_2

Read accounting integrated totals of selected time and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0x78 (120)	= C_CI_NR_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..6] =	CP40Time2a [▶ 430]								Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 430]								Five octets binary time tag (to)
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.27 C_CI_NS_2

Read periodically reset accounting integrated totals of selected time and of selected range of addresses.

- obj					ASDU object [▶ 424]

|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x79 (121)	= C_CI_NS_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..6] =	CP40Time2a [▶ 430]								Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 430]								Five octets binary time tag (to)
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.28 C_CI_NT_2

Read operational integrated totals of selected time and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x7A (122)	= C_CI_NT_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu

|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)
			|---	data[2..6] =	CP40Time2a [▶ 430]								Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 430]								Five octets binary time tag (to)

			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.29 C_CI_NU_2

Read periodically reset operational integrated totals of selected time and of selected range of addresses.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x7B (123)	= C_CI_NU_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =									Integrated total address (from)
			|---	data[1] =									Integrated total address (to)

			|---	data[2..6] =	CP40Time2a [▶ 430]	Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 430]	Five octets binary time tag (to)
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.30 C_SYN_TA_3

Time synchronisation.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x6 (6)	= C_SYN_TA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	eCOT											Cause of transmission [▶ 441]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc		:= GLB									Function code/ type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 7									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..6] =	CP56Time2a [▶ 344]								Seven t octets binary time tag

			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.22.31 C_IGI_NA_3

General interrogation.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x7 (7)	= C_IGI_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 441]
|	|---	asduAddr			Common address of asdu

|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc		:= GLB									Function code/ type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	SCN [▶ 263]								Scan number
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.32 C_GD_NA_3

Generic data.

- obj					ASDU object [▶ 438]

|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x0A (10)	= C_GD_NA_3									Type identification [▶ 439]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 441]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc		:= GEN									Function code/type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= variable									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	RII [▶ 263]								Return information identifier
			|---	data[1] =	CONT [▶ 260]	COUNT [▶ 260]	NO [▶ 262]						NGD = Number of generic data sets

			|---	data[2] =	GROUP [▶ 261]	GIN = Generic identification number	Dataset 1
			|---	data[3] =	ENTRY [▶ 261]		
			|---	data[4] =	KOD [▶ 261]	Kind of description	
			|---	data[5] =	DATATYPE [▶ 260]	GDD = Generic data description	
			|---	data[6] =	DATASIZE [▶ 260]		
			|---	data[7] =	CONT [▶ 260] NUMBER		
			|---	data[8] =	GID	Generic identification data	
			|---	data[9..IEC870_MAX_ASDU_DATA_BYTE] =			Dataset 2..i

4.22.33 C_GRC_NA_3

General command.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x14 (20)	= C_GRC_NA_3	Type identification [▶ 439]

|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 441]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]
	|---	fc		:= GLB	Function code/ type [▶ 270]
	|---	n			Information number [▶ 270]

	|---	stream										Information element/ object data [▶ 316]
		|---	length	= 2								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =							DCS [▶ 338]		DCO = Double command
			|---	data[1] =	RII [▶ 263]								Return information identifier
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.34 C_GC_NA_3

Generic command.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x15 (21)	= C_GC_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 441]

|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc	:= GEN										Function code/type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 2 + (i * 3)									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =				RII [▶ 263]					Return information identification
			|---	data[1] =				NOG [▶ 263]					Number of generic data sets
			|---	data[2] =				GROUP [▶ 261]					GIN = Generic identification number
			|---	data[3] =				ENTRY [▶ 261]					Dataset 1
			|---	data[4] =				KOD [▶ 261]					Kind of description
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Dataset 2..i

4.22.35 C_ODT_NA_3

Order of disturbance data transmission.

- obj				ASDU object [▶ 438]
|---	+ head			Reserved
|---	- ident			DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x18 (24)	= C_ODT_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 441]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]

|---	fc											Function code/ type [▶ 270]
|---	n	= 0										not used
|---	stream											Information element/ object data [▶ 316]
	|---	length	= 5									
	|---	data		7	6	5	4	3	2	1	0	
		|---	data[0] =				TOO [▶ 263]					Type of order
		|---	data[1] =				TOV [▶ 263]					Type of disturbance value
		|---	data[2..3] =				FAN [▶ 261]					Fault number
		|---	data[4] =				ACC [▶ 260]					Actual channel
		|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									

4.22.36 C_ADT_NA_3

Acknowledgement for disturbance data transmission.

- obj												ASDU object [▶ 438]
|---	+ head											Reserved

|---	- ident			DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x19 (25) = C_ADT_NA_3	Type identification [▶ 439]
|	|---	bSQ	= TRUE	Sequence of information objects
|	|---	nObj	= 1	Number of objects
|	|---	eCOT		Cause of transmission [▶ 441]
|	|---	asduAddr		Common address of asdu
|	|---	eClass		Fifo priority class [▶ 333]
|---	- info			INFORMATION OBJECT [▶ 438]
	|---	fc		Function code/ type [▶ 270]
	|---	n	= 0	not used

|---	stream											Information element/ object data [▶ 316]
	|---	length	= 5									
	|---	data		7	6	5	4	3	2	1	0	
		|---	data[0] =				TOO [▶ 263]					Type of order
		|---	data[1] =				TOV [▶ 263]					Type of disturbance value
		|---	data[2..3] =				FAN [▶ 261]					Fault number
		|---	data[4] =				ACC [▶ 260]					Actual channel
		|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									

4.23 Protection equipment information

4.23.1 M_EP_TA_1

Event of protection equipment with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x11 (17)	= M_EP_TA_1	Type identification [▶ 37]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 6								
		|---	data		7	6	5	4	3	2	1 0	
			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	EI [▶ 264]	0	ES [▶ 37]	SEP = Single event of protection equipment
			|---	data[1..2] =	CP16Time2a [▶ 37]							Elapsed time, two octets binary time
			|---	data[3..5] =	CP24Time2a [▶ 37]							Three octets binary time tag
			|---	data[6..IEC870_MAX_ASDU_DATA_BYTE] =								Reserved

4.23.2 M_EP_TB_1

Packed start events of protection equipment with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x12 (18)	= M_EP_TB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 7								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0	0	SRD	SIE	SL3	SL2	SL1	GS	SEP = Start events of protection equipment
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	EI [▶ 264]	0	0	0	QDP = Quality descriptor for events of protection equipment
			|---	data[2..3] =	CP16Time2a [▶ 37]								Relay duration time, two octets binary time
			|---	data[4..6] =	CP24Time2a [▶ 37]								Three octets binary time tag
			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.23.3 M_EP_TC_1

Packed output circuit information of protection equipment with CP24Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x13 (19)	= M_EP_TC_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]

|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 7								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0	0	0	0	CL3	CL2	CL1	GC	OCI = Output cirquit information of protection equipment
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	EI [▶ 264]	0	0	0	QDP = Quality descriptor for events of protection equipment
			|---	data[2..3] =	CP16Time2a [▶ 37]								Relay operating time, two octets binary time
			|---	data[4..6] =	CP24Time2a [▶ 37]								Three octets binary time tag

			|---	data[7..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.23.4 M_EP_TD_1

Event of protection equipment with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x26 (38)	= M_EP_TD_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 10								
		|---	data		7	6	5	4	3	2	1	0
			|---	data[0] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	EI [▶ 264]	0	ES [▶ 37]		SEP = Single event of protection equipment
			|---	data[1..2] =	CP16Time2a [▶ 37]								Elapsed time, two octets binary time
			|---	data[3..9] =	CP56Time2a [▶ 37]								Seven t octets binary time tag

			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.23.5 M_EP_TE_1

Packed start events of protection equipment with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x27 (39)	= M_EP_TE_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]

|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class [▶ 37]
|---	- info											INFORMATION OBJECT [▶ 37]
	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 11								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0	0	SRD	SIE	SL3	SL2	SL1	GS	SEP = Start events of protection equipment
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	EI [▶ 264]	0	0	0	QDP = Quality descriptor for events of protection equipment

			|---	data[2..3] =	CP16Time2a [▶ 37]	Relay duration time, two octets binary time
			|---	data[4..10] =	CP56Time2a [▶ 37]	Seven octets binary time tag
			|---	data[11..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.23.6 M_EP_TF_1

Packed output circuit information of protection equipment with CP56Time2a time tag.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x28 (40)	= M_EP_TF_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 11									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	0	0	0	0	CL3	CL2	CL1	GC	OCI = Output cirquit information of protection equipment
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	EI [▶ 264]	0	0	0	QDP = Quality descriptor for events of protection equipment
			|---	data[2..3] =	CP16Time2a [▶ 37]								Relay operating time, two octets binary time
			|---	data[4..10] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[11..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.23.7 M_PS_NA_1

Packed single point information with status change detection.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x14 (20)	= M_PS_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]

	|---	objAddr										Information object address
	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 5								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	SCD								Status and status change detection (32 bit)
			|---	data[4] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]				OV [▶ 265]	QDS: Quality descriptor
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.24 Parameter loading/activation

4.24.1 P_ME_NA_1

Parameter of measured value, normalized value.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]

|	|---	eType	= 0x6E (110)	= P_ME_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 37]
|	|---	nORG			Originator address
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 37]
|---	- info				INFORMATION OBJECT [▶ 37]
	|---	objAddr			Information object address

	|---	stream										Information element/ object data [▶ 37]
		|---	length	= 3								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	NVA [▶ 37]								Normalized value
			|---	data[2] =	LPC	POP	KPA [▶ 37]						QPM: Qualifier of parameter of measured value
			|---	data[3..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.24.2 P_ME_NB_1

Parameter of measured value, scaled value.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x6F (111)	= P_ME_NB_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= 1									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 3									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0..1] =	SVA [▶ 37]			Scaled value
			|---	data[2] =	LPC	POP	KPA [▶ 37]	QPM: Qualifier of parameter of measured value
			|---	data[3..IEC870_MAX_ASDU_DATA_BYTE] =				Reserved

4.24.3 P_ME_NC_1

Parameter of measured value, short floating point value.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x70 (112)	= P_ME_NC_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr											Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 37]								Short floating point value

			|---	data[4] =	LPC	POP	KPA [▶ 37]	QPM: Qualifier of parameter of measured value
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =				Reserved

4.24.4 P_AC_NA_1

Parameter activation.

- obj					ASDU object [▶ 37]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 37]
|	|---	eType	= 0x71 (113)	= P_AC_NA_1	Type identification [▶ 37]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 37]
|	|---	nORG											Originator address
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 37]
|---	- info												INFORMATION OBJECT [▶ 37]
	|---	objAddr		= 0									Information object address
	|---	stream											Information element/ object data [▶ 37]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	QPA [▶ 37]								Qualifier of parameter activation
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.25 Process information in monitoring direction

4.25.1 M_SP_TA_2

Single-point information with time tag.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x1 (1)	= M_SP_TA_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]
	|---	rcdAddr			Record address
	|---	stream			Information element/ object data [▶ 215]
		|---	length	= i * 9	
		|---	data	7 6 5 4 3 2 1 0	

			|---	data[0] =	SPA [▶ 260]		Single-point address	Information object 1
			|---	data[1] =	SPQ [▶ 260]	SPI [▶ 215]	SPQ = Single-point qualifier SPI = Single-point information	
			|---	data[2..8] =	CP56Time2b [▶ 430]		Seven octets binary time tag	
			|---	data[9..length - 1] =				Information object 2..i
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =			Reserved	

4.25.2 M_IT_TA_2

Accounting integrated totals, 4 octets each

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x2 (2)	= M_IT_TA_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= i									Number of objects	
|	|---	bT											Test	
|	|---	bPN											Positive/ negative confirmation /activation	
|	|---	eCOT											Cause of transmission [▶ 428]	
|	|---	asdu Addr											Common address of asdu	
|	|---	eClass											Fifo priority class [▶ 215]	
|---	- info												INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 215]	
		|---	length	= i * (6 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..4] =	CR4								Counter reading, 4 octets	
			|---	data[5] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[6] =	Signature								Signature (optional)	
			|---	data[7..length - 6] =										Information object 2..i

			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]	Five octets binary time tag
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.25.3 M_IT_TB_2

Accounting integrated totals, 3 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x3 (3)	= M_IT_TB_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]
	|---	rcdAddr			Record address

	|---	stream											Information element/ object data [▶ 215]	
		|---	length	= i * (5 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..3] =	CR3								Counter reading, 3 octets	
			|---	data[4] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[5] =	Signature								Signature (optional)	
			|---	data[6..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]								Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved	

4.25.4 M_IT_TC_2

Accounting integrated totals, 2 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved

|---	- ident												DATA UNIT IDENTIFIER [▶ 426]	
|	|---	eType	= 0x4 (4)	= M_IT_TC_2								Type identification [▶ 427]	
|	|---	bSQ		= FALSE								Sequence of information objects	
|	|---	nObj		= i								Number of objects	
|	|---	bT										Test	
|	|---	bPN										Positive/ negative confirmation /activation	
|	|---	eCOT										Cause of transmission [▶ 428]	
|	|---	asduAddr										Common address of asdu	
|	|---	eClass										Fifo priority class [▶ 215]	
|---	- info											INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr										Record address	
	|---	stream										Information element/ object data [▶ 215]	
		|---	length	= i * (4 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..2] =	CR2								Counter reading, 2 octets	
			|---	data[3] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	

			|---	data[5..length - 6] =			Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]	Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved	

4.25.5 M_IT_TD_2

Periodical reset accounting integrated totals, 4 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x5 (5)	= M_IT_TD_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 215]	
		|---	length	= i * (6 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..4] =	CR4								Counter reading, 4 octets	
			|---	data[5] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[6] =	Signature								Signature (optional)	
			|---	data[7..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]								Five octets binary time tag	
			|---	data[length ..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved	

4.25.6 M_IT_TE_2

Periodical reset accounting integrated totals, 3 octets each.

- obj													ASDU object [▶ 424]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x6 (6)	= M_IT_TE_2									Type identification [▶ 427]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= i									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation /activation
|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= i * (5 + [1 (signature)]) + 5									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	IOA				Information object address	Information object 1
			|---	data[1..3] =	CR3				Counter reading, 3 octets	
			|---	data[4] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence	Quality descriptor	
			|---	data[5] =	Signature				Signature (optional)	
			|---	data[6..length - 6] =						Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]				Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =					Reserved	

4.25.7 M_IT_TF_2

Periodical reset accounting integrated totals, 2 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x7 (7)	= M_IT_TF_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= i									Number of objects	
|	|---	bT											Test	
|	|---	bPN											Positive/ negative confirmation /activation	
|	|---	eCOT											Cause of transmission [▶ 428]	
|	|---	asdu Addr											Common address of asdu	
|	|---	eClass											Fifo priority class [▶ 215]	
|---	- info												INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 215]	
		|---	length	= i * (4 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..2] =	CR2								Counter reading, 2 octets	
			|---	data[3] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	
			|---	data[5..length - 6] =										Information object 2..i

			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]	Five octets binary time tag
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.25.8 M_IT_TG_2

Operational integrated totals, 4 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0x8 (8)	= M_IT_TG_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]
|---	- info				INFORMATION OBJECT [▶ 425]
	|---	rcdAddr			Record address

	|---	stream											Information element/ object data [▶ 215]	
		|---	length	= i * (6 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..4] =	CR4								Counter reading, 4 octets	
			|---	data[5] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[6] =	Signature								Signature (optional)	
			|---	data[7..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]								Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved	

4.25.9 M_IT_TH_2

Operational integrated totals, 3 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved

|---	- ident												DATA UNIT IDENTIFIER [▶ 426]	
|	|---	eType	= 0x9 (9)	= M_IT_TH_2								Type identification [▶ 427]	
|	|---	bSQ		= FALSE								Sequence of information objects	
|	|---	nObj		= i								Number of objects	
|	|---	bT										Test	
|	|---	bPN										Positive/ negative confirmation /activation	
|	|---	eCOT										Cause of transmission [▶ 428]	
|	|---	asduAddr										Common address of asdu	
|	|---	eClass										Fifo priority class [▶ 215]	
|---	- info											INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr										Record address	
	|---	stream										Information element/ object data [▶ 215]	
		|---	length	= i * (5 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..3] =	CR3								Counter reading, 3 octets	
			|---	data[4] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[5] =	Signature								Signature (optional)	

			|---	data[6..length - 6] =			Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]	Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved	

4.25.10 M_IT_TI_2

Operational integrated totals, 2 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0xA (10)	= M_IT_TI_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 215]	
		|---	length	= i * (4 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..2] =	CR2								Counter reading, 2 octets	
			|---	data[3] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	
			|---	data[5..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]								Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved	

4.25.11 M_IT_TK_2

Periodical reset operational integrated totals, 4 octets each.

- obj													ASDU object [▶ 424]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0xB (11)	= M_IT_TK_2									Type identification [▶ 427]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= i									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation /activation
|	|---	eCOT											Cause of transmission [▶ 428]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 215]
|---	- info												INFORMATION OBJECT [▶ 425]
	|---	rcdAddr											Record address
	|---	stream											Information element/ object data [▶ 215]
		|---	length	= i * (6 + [1 (signature)]) + 5									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	IOA				Information object address	Information object 1
			|---	data[1..4] =	CR4				Counter reading, 4 octets	
			|---	data[5] =	IV	CA	CY	Sequence	Quality descriptor	
			|---	data[6] =	Signature				Signature (optional)	
			|---	data[7..length - 6] =						Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]				Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =					Reserved	

Also see about this

- Other information elements [▶ 264]
- Other information elements [▶ 264]
- Other information elements [▶ 264]

4.25.12 M_IT_TL_2

Periodical reset operational integrated totals, 3 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]

|	|---	eType	= 0xC (12)	= M_IT_TL_2										Type identification [▶ 427]	
|	|---	bSQ		= FALSE										Sequence of information objects	
|	|---	nObj		= i										Number of objects	
|	|---	bT												Test	
|	|---	bPN												Positive/ negative confirmation /activation	
|	|---	eCOT												Cause of transmission [▶ 428]	
|	|---	asdu Addr												Common address of asdu	
|	|---	eClass												Fifo priority class [▶ 215]	
|---	- info													INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr												Record address	
	|---	stream												Information element/ object data [▶ 215]	
		|---	length	= i * (5 + [1 (signature)]) + 5											
		|---	data		7	6	5	4	3	2	1	0			
			|---	data[0] =	IOA									Information object address	Information object 1
			|---	data[1..3] =	CR3									Counter reading, 3 octets	
			|---	data[4] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence						Quality descriptor	
			|---	data[5] =	Signature									Signature (optional)	

			|---	data[6..length - 6] =			Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]	Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved	

4.25.13 M_IT_TM_2

Periodical reset operational integrated totals, 2 octets each.

- obj					ASDU object [▶ 424]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 426]
|	|---	eType	= 0xD (13)	= M_IT_TM_2	Type identification [▶ 427]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 428]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 215]

|---	- info												INFORMATION OBJECT [▶ 425]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 215]	
		|---	length	= i * (4 + [1 (signature)]) + 5										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	IOA								Information object address	Information object 1
			|---	data[1..2] =	CR2								Counter reading, 2 octets	
			|---	data[3] =	IV [▶ 264]	CA [▶ 264]	CY [▶ 264]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	
			|---	data[5..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 430]								Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved	

4.25.14 M_TTM_TA_3

Time-tagged message.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x1 (1)	= M_TTM_TA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]

	|---	fc										Function code/ type [▶ 270]
	|---	n										Information number [▶ 270]
	|---	stream										Information element/ object data [▶ 316]
		|---	length	= 6								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0	0	0	0	0	0	DPI [▶ 342]		DPI = Double-point information
			|---	data[1..4] =	CP32Time2a [▶ 344]								Four octets binary time tag
			|---	data[5] =	SIN [▶ 263]								Supplementary information
			|---	data[6..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.25.15 M_TMR_TA_3

Time-tagged message with relative time.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x2 (2)	= M_TMR_TA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]

	|---	fc										Function code/ type [▶ 270]
	|---	n										Information number [▶ 270]
	|---	stream										Information element/ object data [▶ 316]
		|---	length	= 10								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0	0	0	0	0	0	DPI [▶ 342]		DPI = Double-point information
			|---	data[1..2] =	RET [▶ 263]								Relative time
			|---	data[3..4] =	FAN [▶ 261]								Fault number
			|---	data[5..8] =	CP32Time2a [▶ 344]								Four octets binary time tag
			|---	data[9] =	SIN [▶ 263]								Supplementary information
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.25.16 M_MEI_NA_3

Measurands I.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x3 (3)	= M_MEI_NA_3	Type identification [▶ 439]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]

|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc											Function code/ type [▶ 270]
	|---	n											Information number [▶ 270]
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 8 * i									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[1] =	MVAL [▶ 262]								Current L2
			|---	data[2] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[3] =	MVAL [▶ 262]								Voltage L1-L2

			|---	data[4] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[5] =	MVAL [▶ 262]				Active power P
			|---	data[6] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[7] =	MVAL [▶ 262]				Reactive power Q
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =					Reserved

4.25.17 M_TME_TA_3

Time-tagged measurands with relative time

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x44	= M_TME_TA_3	Type identification [▶ 439]

|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]
	|---	fc			Function code/ type [▶ 270]
	|---	n			Information number [▶ 270]

	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 265]								Short circuit location
			|---	data[4..5] =	RET [▶ 263]								Relative time
			|---	data[6..7] =	FAN [▶ 261]								Fault number
			|---	data[8..11] =	CP32Time2a [▶ 344]								Four octets binary time tag
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.25.18 M_MEII_NA_3

Measurands II.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]

|	|---	eType	= 0x9 (9)	= M_MEII_NA_3	Type identification [▶ 439]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]
	|---	fc			Function code/ type [▶ 270]
	|---	n			Information number [▶ 270]

	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 18 * i									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[1] =	MVAL [▶ 262]								Current L1
			|---	data[2] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[3] =	MVAL [▶ 262]								Current L2
			|---	data[4] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[5] =	MVAL [▶ 262]								Current L3
			|---	data[6] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[7] =	MVAL [▶ 262]								Voltage L1-E

			|---	data[8] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[9] =	MVAL [▶ 262]				Voltage L2-E
			|---	data[10] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[11] =	MVAL [▶ 262]				Voltage L3-E
			|---	data[12] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[13] =	MVAL [▶ 262]				Active power P
			|---	data[14] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[15] =	MVAL [▶ 262]				Reactive power Q
			|---	data[16] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[17] =	MVAL [▶ 262]				Frequency f

			|---	data[18..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.25.19 M_LRD_TA_3

List of recorded disturbances.

- obj													ASDU object [▶ 438]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x17 (23)	= M_LRD_TA_3									Type identification [▶ 439]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= i									Number of objects
|	|---	eCOT											Cause of transmission [▶ 440]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc											Function code/type [▶ 270]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= i * 10									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0..1] =	FAN [▶ 261]					Fault number	Dataset 1
			|---	data[2] =	RES	OTEV	TEST	TM	TP	SOF = Status of fault	
			|---	data[3..9] =	CP56Time2a [▶ 344]					Seven octets binary time tag	
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =							Dataset 2..i

4.25.20 M_RTD_TA_3

Ready for transmission of disturbance data.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	0x1A (26)	= M_RTD_TA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	eCOT											Cause of transmission [▶ 440]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc											Function code/ type [▶ 270]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 15									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0								not used
			|---	data[1] =	TOV [▶ 263]								Type of disturbance values
			|---	data[2..3] =	FAN [▶ 261]								Fault number

			|---	data[4..5] =	NOF [▶ 263]	Number of grid faults
			|---	data[6] =	NOC [▶ 262]	Number of channels
			|---	data[7..8] =	NOE [▶ 262]	Number of information elements of a channel
			|---	data[9..10] =	INT [▶ 261]	Interval
			|---	data[11..14] =	CP32Time2a [▶ 344]	Four octets binary time tag
			|---	data[15..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.25.21 M_RTC_NA_3

Ready for transmission of channel.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x1B (27)	= M_RTC_NA_3	Type identification [▶ 439]

|	|---	bSQ		= TRUE									Sequence of information objects
|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 440]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc											Function code/ type [▶ 270]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 17									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	0	not used
			|---	data[1] =	TOV [▶ 263]	Type of disturbance value
			|---	data[2..3] =	FAN [▶ 261]	Fault number
			|---	data[4] =	ACC [▶ 260]	Actual channel
			|---	data[5..8] =	R32 [▶ 265]	RPV = Rated primary value
			|---	data[9..12] =	R32	RSV = Rated secondary value
			|---	data[13..16] =	R32	RFA = Reference factor
			|---	data[17..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.25.22 M_RTT_NA_3

Ready for transmission of tags.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]

|	|---	eType	0x1C (28)	= M_RTT_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]
	|---	fc			Function code/ type [▶ 270]
	|---	n		= 0	not used

	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 4									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0								not used
			|---	data[1] =	0								not used
			|---	data[2..3] =	FAN [▶ 261]								Fault number
			|---	data[4..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.25.23 M_TOT_NA_3

Transmission of tags.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x1D (29)	= M_TOT_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]

|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc											Function code/type [▶ 270]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 5 + (i * 3)									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	FAN [▶ 261]								Fault number
			|---	data[2] =	NOT [▶ 263]								Number of tags
			|---	data[3..4] =	TAP [▶ 263]								Tag position
			|---	data[5] =									Function code/ type	Tag 1
			|---	data[6] =									Information number	
			|---	data[7] =	0						DPI [▶ 342]		Double point information	
			|---	data[8..IEC870_MAX_ASDU_DATA_BYTE] =										Tag 2..i

4.25.24 M_TOV_NA_3

Transmission of disturbance values.

- obj					ASDU object [▶ 438]

|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x1E (30)	= M_TOV_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 333]
|---	- info				INFORMATION OBJECT [▶ 438]
	|---	fc			Function code/ type [▶ 270]

	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 8 + (i *2)									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0								not used
			|---	data[1] =	TOV [▶ 263]								Type of disturbance values
			|---	data[2..3] =	FAN [▶ 261]								Fault number
			|---	data[4] =	ACC [▶ 260]								Actual channel
			|---	data[5] =	NDV [▶ 262]								Number of relevant disturbance values per ASDU
			|---	data[6..7] =	NFE [▶ 262]								Number of the ASDU's first information element
			|---	data[8..9] =	SDV [▶ 263]								Single disturbance value 1
			|---	data[10..11] =	SDV [▶ 263]								Single disturbance value 2

			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =	SDV [▶ 263]	Single disturbance value 3..i

4.25.25 M_EOT_NA_3

End of transmission.

- obj					ASDU object [▶ 438]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 439]
|	|---	eType	= 0x1F (31)	= M_EOT_NA_3	Type identification [▶ 439]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 440]
|	|---	asduAddr			Common address of asdu

|	|---	eClass											Fifo priority class [▶ 333]
|---	- info												INFORMATION OBJECT [▶ 438]
	|---	fc											Function code/ type [▶ 270]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 316]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	TOO [▶ 263]								Type of order
			|---	data[1] =	TOV [▶ 263]								Type of disturbance values
			|---	data[2..3] =	FAN [▶ 261]								Fault number
			|---	data[4] =	ACC [▶ 260]								Actual channel
			|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.26 Information elements

4.26.1 SPA

Single-point address: <0..255>

- <0..127> compatible range
- <128..255> private range

4.26.2 SPQ

Single-point qualifier: <0..127>

- <0> not specified
- <1..127> vendor specific

4.26.3 ACC

Actual channel: <0..255>

4.26.4 ASC

ASCII 8 bit code: <0..255>

4.26.5 COL

Compatibility level: <0..255>

4.26.6 CONT

Continuous flag

- <0> no following ASDU with the same RII;
- <1> following ASDU has the same RII;

4.26.7 COUNT

One bit counter for ASDU with equal RII: <0..1>

4.26.8 DATASIZE

Data size: <1..255>

4.26.9 DATATYPE

Data type: <0..255>

- <0> no data
- <1> OS8ASCII
- <2> PACKEDBITSTRING
- <3> UI
- <4> I

- <5> UF
- <6> F
- <7> R32
- <8> R64
- <9> Double poing information
- <10> Single poing information
- <11>
- <12>
- <13>
- <14>
- <15>
- <16>
- <17>
- <18>
- <19>
- <20>
- <21>
- <22>
- <23> data struct
- <24> index
- <25..255> reserved

4.26.10 ENTRY

Entry identification: <0..255>

4.26.11 FAN

Fault number: <0..65535>

4.26.12 GROUP

Group identification: <0..255>

4.26.13 INT

Interval: <1..65535> [µs]

4.26.14 KOD

Kind of description: <0..255>

- <0> no KOD specified
- <1> actual value
- <2> default value
- <3> range (min, max, step size)
- <4> reserved
- <5> precision

- <6> factor
- <7> % reference
- <8> enumeration
- <9> dimension
- <10> description
- <11> reserved
- <12> password entry
- <13> is read only
- <14> is write only
- <15> reserved
- <16> reserved
- <17> reserved
- <18> reserved
- <19> corresponding function type and information number
- <20> corresponding event
- <21> enumerated text array
- <22> enumerated value array
- <23> related entries
- <24..255> reserved

4.26.15 MVAL

Measured value:<-1..+1.-2E-12>

4.26.16 NDV

Number of relevant disturbance values per ASDU: <1..255>

- <1..25> used
- <26.255> not used

4.26.17 NFE

Number of the ASDU's first information element: <0..65535>

4.26.18 NO

Number of generic data sets: <0..63>

4.26.19 NOC

Number of channels: <0..255>

4.26.20 NOE

Number of information elements of a channel: <0..65535>

4.26.21 NOF

Number of grid faluts: <0..65535>

4.26.22 NOG

Number of generic identifications: <0..255>

4.26.23 NOT

Number of tags: <1..255>

4.26.24 RET

Relative time: <0..65535>

4.26.25 RII

Return information identifier.

USINT <0..255>

4.26.26 SCN

Scan number: <0..255>

4.26.27 SDV

Single disturbance value: <-1..+1-2E-15>

4.26.28 SIN

Supplementary information: <0..255>

4.26.29 TAP

Tag position: <0..65535>

4.26.30 TOO

Type of order: <1..255>

4.26.31 TOV

Type of disturbance value: <0..255>

4.26.32 Other information elements

RES

ER

OV

OTEV

TEST

TM

TP

BL

BL

Blocked quality flag:

- <0> = not blocked;
- <1> = blocked;

CA

CA

Adjusted flag:

- <0> = Counter was not adjusted;
- <1> = Counter was adjusted;

CY

CY

Carry flag:

- <0> = no carry;
- <1> = carry;

EI

EI

Elapsed flag:

- <0> = Elapsed time valid;
- <1> = Elapsed time not valid;

IV

IV

Invalid quality flag:

- <0> = valid;
- <1> = invalid;

NT

NT

Topical quality flag:

- <0> = topical;
- <1> = not topical;

OV

OV

Overflow quality flag:

- <0> = no overflow;
- <1> = overflow;

QDS

QDS

Quality descriptor

Quality descriptor

SB

SB

Substituted quality flag:

- <0> = not substituted;
- <1> = substituted;

4.26.33 NVA

Normalized value.

4.26.34 S/E

Select/execute state.

- <0> = Execute;
- <1> = Select;

4.26.35 BSI

Bitstring of 32 bits.

4.26.36 SVA

Scaled value.

4.26.37 R32

Short floating point value.

4.26.38 TSC

Test command counter.

4.26.39 LPC

Local parameter change flag.

- <0> = No change;
- <1> = Changed;

4.26.40 BCR

Binary counter reading.

4.26.41 VTI

Value with transient state indication (8 bits).

Transient state (bit 7):

- <0> = equipment is not in transient state;
- <1> = equipment is in transient state;

Value (bits 0..6) = <-64..63>;

4.27 Step position information

4.27.1 M_ST_NA_1

Step position information without time tag.

- obj					ASDU object
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER
|	|---	eType	= 0x05 (5)	= M_ST_NA_1	Type identification
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= 1								Number of objects
|	|---	bT										Test
|	|---	bPN										Positive/ negative confirmation/ activation
|	|---	eCOT										Cause of transmission
|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class
|---	- info											INFORMATION OBJECT
	|---	objAddr										Information object address
	|---	stream										Information element/ object data
		|---	length	= 2								
		|---	data		7	6	5	4	3	2	1	0
			|---	data[0] =	VTI							Value with transient state indication

			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.27.2 M_ST_TB_1

Step position information with CP56Time2a time tag.

- obj					ASDU object
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER
|	|---	eType	= 0x20 (32)	= M_ST_TB_1	Type identification
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission

|	|---	nORG										Originator address
|	|---	asduAddr										Common address of asdu
|	|---	eClass										Fifo priority class
|---	- info											INFORMATION OBJECT
	|---	objAddr										Information object address
	|---	stream										Information element/ object data
		|---	length	= 9								
		|---	data		7	6	5	4	3	2	1	0
			|---	data[0] =	VTI							Value with transient state indication
			|---	data[1] =	IV [▶ 264]	NT [▶ 265]	SB [▶ 265]	BL [▶ 264]	0	0	0	OV [▶ 265]	QDS = Quality descriptor
			|---	data[2..8] =	CP56Time2a							Seven octets binary time tag
			|---	data[9..IEC870_MAX_ASDU_DATA_BYTE] =								Reserved

4.28 Function type (code)

Type	Description
0..127	Reserved (private area)
128	Distance protection
129	Not used (compatible area)
130..143	Reserved (private area)
144..145	Not used (compatible area)
146..159	Reserved (private area)
160	Overcurrent protection
161	Not used (compatible area)
162..175	Reserved (private area)
176	Transformer differential protection
177	Not used (compatible area)
178..191	Reserved (private area)
192	Line dirrerential protection
193	Not used (compatible area)
194..207	Reserved (private area)
208	Not used (compatible area)
209	Not used (compatible area)
210..223	Reserved (private area)
224	Not used (compatible area)
225	Not used (compatible area)
226..239	Reserved (private area)
240	Not used (compatible area)
241	Not used (compatible area)
242..253	Reserved (private area)
254	Generic function (GEN)
255	Global function (GLB)

4.29 Information number

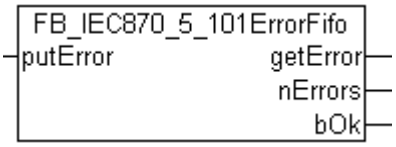
Number	Monitoring direction	Control direction
0..15	System functions	System functions
16..31	State	General commands
32..47	Control	Not used
48..63	Earth faults	Not used
64..127	Short-circuit faults	Not used
128..143	Automatic reclose	Not used
144..159	Operating measured values	Not used
160..239	Not used	Not used
240..255	Generic functions	Generic functions

5 TcIEC870_5_101: IEC 60870-5-101 Common Data Types

The TwinCAT PLC Library **TcIEC870_5_101.Lib** implements the connection functions and data structures according to IEC60870-5-101.

5.1 Function blocks

5.1.1 FB_IEC870_5_101ErrorFifo



IEC60870-5-10x error FIFO. The oldest entry is overwritten. The FIFO has a constant size, determined by the constant: *IEC870_MAX_ERROR_FIFO_SIZE* (default: 10 elements).

The function block features three tasks:

- **AddError** (adds a new error message to the FIFO);
- **RemoveError** (removes the oldest error message from the FIFO);
- **Reset** (clears all error messages and resets the FIFO);

Error messages are normally added to the FIFO via the internal IEC60870-5-10x device functions. The PLC application can read and analyse these error messages by calling the task: **RemoveError**.

VAR_INPUT

```
VAR_INPUT
    putError    : ST_IEC870_5_101ErrorFifoEntry;
END_VAR
```

putError: Error message to be added to the FIFO [▶ 320].

VAR_OUTPUT

```
VAR_OUTPUT
    getError    : ST_IEC870_5_101ErrorFifoEntry;
    nErrors     : UDINT; (* Error counter *)
    bOk         : BOOL; (* TRUE = new entry added or removed succesfully, FALSE = fifo empty *)
END_VAR
```

getError: Error message to be removed from the FIFO [▶ 320].

nErrors: Returns the current number of FIFO entries (error messages in the FIFO).

bOk: This variable becomes TRUE if a new entry was successfully added to or removed from the FIFO.

Example of call in ST:

In the following structured text example the device error FIFO is read, and the registered errors are written into the Windows Application Log.

```
PROGRAM P_LogErrors
VAR_IN_OUT
    fbErrors : FB_IEC870_5_101ErrorFifo;
END_VAR
```

```
REPEAT
    fbErrors.RemoveError();
    IF fbErrors.bOk THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
```

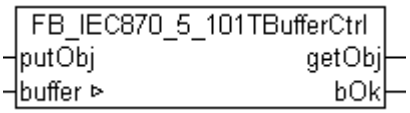
```
            'IEC60870-5-10x device error: 0x%s',
            DWORD_TO_HEXSTR( fbErrors.getError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT fbErrors.bOk
END_REPEAT
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.1.2 FB_IEC870_5_101TBufferCtrl

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 / IEC60870-5-101 substation v2.0.2 and higher.



This function block changes the content of the TX/RX data buffer. This data buffer ist used by the communication via the IEC60870-5-104/101 Transport Interface.

The function block has the following actions:

- **RxRemoveObj** (removes the oldest Fifo entry from the RX-Fifo);
- **RxReset** (deletes all RX-Fifo entries,resets the RX-Fifo);
- **TxAddObj** (inserts a new Fifo entry in the TX-Fifo);
- **TxReset** (deletes all TX-Fifo entries, resets the TX-Fifo)

The content of the TX/RX data buffer can be changed by the call of one of the listed actions above.

VAR_IN_OUT

```
VAR_IN_OUT
    buffer  : ST_IEC870_5_101TBuffer; (* TX/RX data buffer *)
END_VAR
```

buffer: TX/RX data buffer [▶ 312]. The TX/RX buffer parameters (like e.g. asduSize) have to be configured before using.

VAR_INPUT

```
VAR_INPUT
    putObj  : ST_IEC870_5_101AOGen; (* ASDU to send *)
END_VAR
```

putObj: Data unit [▶ 314] (ASDU) to be sent.

VAR_OUTPUT

```
VAR_OUTPUT
    getObj  : ST_IEC870_5_101AOGen; (* received ASDU *)
    bOk     : BOOL; (* TRUE = action succesfully, FALSE=Fifo overflow or fifo empty *)
END_VAR
```

getObj: Received data unit [▶ 314] (ASDU).

bOk: This variable becomes TRUE, if a new entry was inserted or removed successfully from the Fifo. This variable becomes FALSE at a buffer overflow and if no entry could be removed, because the Fifo was already emty.

Example in ST:

The following example code shows the using of theactions. Each ~100ms (tCycle) a new ASDU (M_BO_TB_1) with the transmission cause spontaneous and time stamp will be created and stored in the TX-Fifo.
The received test commands (C_TS_TA_1) and time synchronisation commands (C_CS_NA_1) are removed from the RX-Fifo and answered with mirrored ASDUs

```
PROGRAM P_SAMPLE_1ms
VAR
    (* TX/RX data buffer *)
 fbBuffer    : FB_IEC870_5_101TBufferCtrl;
    buffer      : ST_IEC870_5_101TBuffer := ( asduSize := 253 );

    tCycle  : TIME := T#100ms;
    timer   : TON;
    dtStart     : DT := DT#2006-07-05-12:34:56;

    txAsdu  : ST_IEC870_5_101AOGen;
    rxAsdu  : ST_IEC870_5_101AOGen;

    txBSI   : DWORD := 1;
    txQDS   : BYTE;
    txTT    : T_CP56Time2a;
    rxTT    : T_CP56Time2a;
END_VAR
```

```
timer( IN := TRUE, PT := tCycle );
IF timer.Q THEN
    timer( IN := FALSE );

    txAsdu.ident.eType := M_BO_TB_1; (* Bit string with time tag *)
    txAsdu.ident.bSQ := FALSE;
    txAsdu.ident.nObj := 1;
    txAsdu.ident.eCOT := eIEC870_COT_SPONTAN;
    txAsdu.ident.nORG := 1;
    txAsdu.ident.bPN := FALSE;
    txAsdu.ident.bT := FALSE;
    txAsdu.ident.eClass := eIEC870_Class_1;
    txAsdu.ident.asduAddr := 7;
    txAsdu.info.objAddr := 100;


    txBSI := ROL( txBSI, 1); (* Modify bit string value *)
    txQDS.7 := NOT txQDS.7; (* Toggle IV quality flag *)(* create dummy time tag *)
    dtStart := dtStart + tCycle;
    txTT := SYSTEMTIME_TO_CP56Time2a( DT_TO_SYSTEMTIME( dtStart ), TRUE );

    F_iecResetStream( 0, txAsdu.info.stream ); (* clear previous data (this sets the stream length =
 0 *)
 F_iecCopyBufferToStream( ADR( txBSI ), SIZEOF( txBSI ), txAsdu.info.stream ); (* put BSI to stream
*)
 F_iecCopyBufferToStream( ADR( txQDS ), SIZEOF( txQDS ), txAsdu.info.stream ); (* put QDS to stream
*)
 F_iecCopyBufferToStream( ADR( txTT ), SIZEOF( txTT ), txAsdu.info.stream ); (* put time tag to stre
am *)

    fbBuffer.TxAddObj( putObj := txAsdu, buffer := buffer ); (* put asdu to the TX fifo *)
 IF NOT fbBuffer.bOk THEN
        ;(* Report send buffer overflow error *)
 END_IF
END_IF



REPEAT
fbBuffer.RxRemoveObj( getObj=>rxAsdu, buffer := buffer ); (* Try to remove asdu from RX fifo *)
IF fbBuffer.bOk THEN (* success *)
```

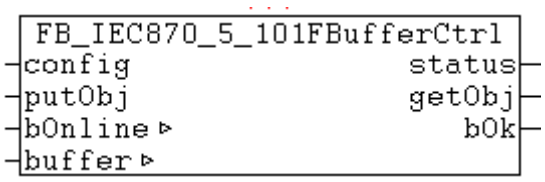
```
    CASE rxAsdu.ident.eType OF
    C_TS_TA_1: (* Test command *)

        txAsdu := rxAsdu;
        txAsdu.ident.eCOT := eIEC870_COT_ACT_CON; (* send activation confirmation *)

        fbBuffer.TxAddObj( putObj := txAsdu, buffer := buffer ); (* put asdu to the TX fifo *)
     IF NOT fbBuffer.bOk THEN
            ;(* Report send buffer overflow error *)
     END_IF


    C_CS_NA_1: (* clock synchronisation *)

        F_iecCopyStreamToBuffer( ADR( rxTT ), SIZEOF( rxTT ), rxAsdu.info.stream );

        (*... *)

        txAsdu := rxAsdu; (* dummy old time value *)
     txAsdu.ident.eCOT := eIEC870_COT_ACT_CON; (* send activation confirmation *)

        fbBuffer.TxAddObj( putObj := txAsdu, buffer := buffer ); (* put asdu to the TX fifo *)
     IF NOT fbBuffer.bOk THEN
            ;(* Report send buffer overflow error *)
     END_IF

    END_CASE

END_IF
UNTIL NOT fbBuffer.bOk (* RX fifo is empty *)
END_REPEAT
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.1.3 FB_IEC870_5_101FBufferCtrl

From product version: TwinCAT PLC Library IEC60870-5-101/104 substation v3.0.2 / IEC60870-5-104 controlling station v1.0.2 and higher.



This function block allows the contents of the TX/RX data buffer that is used for communication through the IEC60870-5-104/101 (low level) transport interface to be manipulated. In addition, the ASDUs that are to be transmitted (only in the TX direction) are buffered in the file if the connection to the central station is interrupted (in offline mode). The functionality is like that of the FB_IEC870_5_101TBufferCtrl function block.

The function block features the following tasks:

- **RxRemoveObj** (removes the oldest FIFO entry from the RX FIFO);
- **RxReset** (clears all the RX FIFO entries, resets the RX FIFO);
- **TxAddObj** (inserts a new FIFO entry into the TX FIFO);

- **TxReset** (clears all the TX FIFO entries, resets the TX FIFO);

The content of the TX/RX data buffer can be changed by calling the actions listed above.

VAR_IN_OUT

```
VAR_IN_OUT
    bOnline : BOOL; (* TRUE => route frames to the link layer, FALSE => route frames to the file buf
fer *)
    buffer  : ST_IEC870_5_101TBuffer; (* TX/RX data buffer *)
END_VAR
```

bOnline: This input tells the function block whether the connection is in offline or online mode. TRUE = online, FALSE = offline. In offline mode, the ASDUs that are to be transmitted are buffered in a file. In online mode, the ASDUs that are buffered in the file are taken out of the file and sent to the central station.

buffer: TX/RX data buffer [▶ 312]. The TX/RX buffer parameters (such as asduSize) must be configured before use.

VAR_INPUT

```
VAR_INPUT
    config  : ST_IEC870_5_101FBufferCfg; (* File buffer configuration settings *)
    putObj  : ST_IEC870_5_101AOGen; (* ASDU to send *)
END_VAR
```

config: offline file buffer configuration [▶ 326] settings.

putObj: data unit [▶ 314] (ASDU) that is to be transmitted.

VAR_OUTPUT

```
VAR_OUTPUT
    status  : ST_IEC870_5_101FBufferStatus; (* File buffer status *)
    getObj  : ST_IEC870_5_101AOGen; (* received ASDU *)
    bOk     : BOOL; (* TRUE = action succesfully, FALSE=Fifo overflow or fifo empty *)
END_VAR
```

status: Offline data buffer status information.

getObj:: Received data unit (ASDU).

bOk: This variable becomes TRUE if a new entry was successfully added to or removed from the FIFO. This variable becomes FALSE if the buffer overflows and if it is not possible to remove an entry because the FIFO is already empty.

Example in ST:

The following fragment of example code demonstrates the use of the function block actions. Approximately every 1s (tCycle) a new ASDU (M_BO_TB_1) is generated with "spontaneous" as the transmission cause and with a timestamp and is placed in the TX FIFO.
The test command (C_TS_TA_1) and clock synchronization command (C_CS_NA_1) received are removed from the RX FIFO, and replied to with reflected ASDUs.

The VAR_IN_OUT variable *bOnline* is used to control the storage or loading of the ASDUs in the file. If *bOnline* = FALSE the file is opened, and all the TX ASDUs that arise are written into the file in the background. If bOnline = TRUE the ASDUs are taken out of the file in the background and transmitted. For saving and loading of the buffer file in the background to be possible, the FB_IEC870_5_101FBufferCtrl function block must be called cyclically. The insertion of new TX ASDUs or the editing of old RX ASDUs is not affected by saving or loading data into or out of the file.

```
PROGRAM P_ProcessSlaveBufferData
VAR_IN_OUT
    bOnline : BOOL;
    buffer : ST_IEC870_5_101TBuffer;
END_VAR
```

```
VAR
    asduAddr : DWORD := 7; (* Common asdu address *)

    fbBuffer : FB_IEC870_5_101FBufferCtrl :=( config := ( sPathName := 'c:\tmp\OfflineAsdu.dat',
                          bOverwrite := TRUE,
                        cbBuffer := 16#100000 )); (* RX/TX buffer control function block *)

    txAsdu : ST_IEC870_5_101AOGen; (* asdu to send *)
    txTT : T_CP56Time2a; (* time tag to send *)

    rxAsdu : ST_IEC870_5_101AOGen; (* received asdu *)
    rxTT : T_CP56Time2a; (* received time tag *)

    rxQOI : BYTE; (* qualifier of interrogation command *)
    txBSI : DWORD := 1; (* bit string value *)
    txQDS : BYTE; (* bit string quality descriptor *)
    tCycle : TIME := T#1s;
    bSpont : BOOL := TRUE;
    timer : TON;
    fbRTC : RTC_EX2 := ( EN := TRUE, PDT := ( wYear := 2006, wMonth := 8, wDay := 17, wHour := 12, w
Minute := 23 ) );
END_VAR
```

```
timer( IN := bSpont, PT := tCycle );
IF timer.Q THEN
    timer( IN := FALSE ); timer( IN := bSpont );

    txBSI := ROL( txBSI, 1); (* Modify bit string value *)
    txQDS.7 := NOT txQDS.7; (* Toggle IV quality flag *)(* create dummy time tag *)
    fbRTC();
    txTT := SYSTEMTIME_TO_CP56Time2a( fbRTC.CDT, TRUE );

    (* create asdu *)
    txAsdu.ident.eType := M_BO_TB_1; (* Bit string with time tag *)
    txAsdu.ident.bSQ := FALSE;
    txAsdu.ident.nObj := 1;
    txAsdu.ident.eCOT := eIEC870_COT_SPONTAN;
    txAsdu.ident.nORG := 1;
    txAsdu.ident.bPN := FALSE;
    txAsdu.ident.bT := FALSE;
    txAsdu.ident.eClass := eIEC870_Class_1;
    txAsdu.ident.asduAddr := asduAddr;
    txAsdu.info.objAddr := 100;
    F_iecResetStream( 0, txAsdu.info.stream ); (* clear previous data (this sets the stream length =
 0 *)
    F_iecCopyBufferToStream( ADR( txBSI ), SIZEOF( txBSI ), txAsdu.info.stream ); (* put BSI to stre
am *)
    F_iecCopyBufferToStream( ADR( txQDS ), SIZEOF( txQDS ), txAsdu.info.stream ); (* put QDS to stre
am *)
    F_iecCopyBufferToStream( ADR( txTT ), SIZEOF( txTT ), txAsdu.info.stream ); (* put time tag to s
tream *)

    fbBuffer.TxAddObj( bOnline := bOnline, putObj := txAsdu, buffer := buffer ); (* put asdu to the
TX fifo *)
    IF NOT fbBuffer.bOk THEN
        RETURN;
        (* TODO: Report send buffer overflow error *)
    END_IF
END_IF




REPEAT
    fbBuffer.RxRemoveObj( bOnline := bOnline, getObj=>rxAsdu, buffer := buffer ); (* Try to remove a
sdu from RX fifo *)
    IF fbBuffer.RxRemoveObj.bOk THEN (* success *)
        CASE rxAsdu.ident.eType OF

        C_TS_NA_1: (* Simple test command implementation *)

        txAsdu := rxAsdu;
        txAsdu.ident.eCOT := eIEC870_COT_ACT_CON; (* send activation confirmation *)
        fbBuffer.TxAddObj( bOnline := bOnline, putObj := txAsdu, buffer := buffer ); (* put asdu to
the TX fifo *)
        IF NOT fbBuffer.bOk THEN
            EXIT;
            (* TODO: Report send buffer overflow error *)
        END_IF
```

```
        C_CS_NA_1: (* Simple clock synchronisation command implementation *)

        F_iecCopyStreamToBuffer( ADR( rxTT ), SIZEOF( rxTT ), rxAsdu.info.stream );

        (*... *)

        txAsdu := rxAsdu; (* dummy old time value *)
        txAsdu.ident.eCOT := eIEC870_COT_ACT_CON; (* send activation confirmation *)
        fbBuffer.TxAddObj( bOnline := bOnline, putObj := txAsdu, buffer := buffer ); (* put asdu to
the TX fifo *)
        IF NOT fbBuffer.bOk THEN
            EXIT;
            (* TODO: Report send buffer overflow error *)
        END_IF

        C_IC_NA_1: (* Simple interrogation command implementation *)

        txAsdu := rxAsdu;
        txAsdu.ident.eCOT := eIEC870_COT_ACT_CON; (* send activation confirmation *)
        fbBuffer.TxAddObj( bOnline := bOnline, putObj := txAsdu, buffer := buffer ); (* put asdu to
the TX fifo *)
        IF NOT fbBuffer.bOk THEN
            EXIT;
            (* TODO: Report send buffer overflow error *)
        END_IF

        F_iecCopyStreamToBuffer( ADR( rxQOI ), SIZEOF(rxQOI), rxAsdu.info.stream );

        (* create asdu *)
        txAsdu.ident.eType := M_BO_NA_1; (* Bit string without time tag! *)
        txAsdu.ident.bSQ := FALSE;
        txAsdu.ident.nObj := 1;
        txAsdu.ident.eCOT := BYTE_TO_INT( rxQOI );
        txAsdu.ident.nORG := 1;
        txAsdu.ident.bPN := FALSE;
        txAsdu.ident.bT := FALSE;
        txAsdu.ident.eClass := eIEC870_Class_1;
        txAsdu.ident.asduAddr := asduAddr;
        txAsdu.info.objAddr := 100;
        F_iecResetStream( 0, txAsdu.info.stream ); (* clear previous data (this sets the stream leng
th = 0 *)
        F_iecCopyBufferToStream( ADR( txBSI ), SIZEOF( txBSI ), txAsdu.info.stream ); (* put BSI to
stream *)
        F_iecCopyBufferToStream( ADR( txQDS ), SIZEOF( txQDS ), txAsdu.info.stream ); (* put QDS to
stream *)
        fbBuffer.TxAddObj( bOnline := bOnline, putObj := txAsdu, buffer := buffer ); (* put asdu to
the TX fifo *)
        IF NOT fbBuffer.bOk THEN
            EXIT;
            (* TODO: Report send buffer overflow error *)
        END_IF


        txAsdu := rxAsdu;
        txAsdu.ident.eCOT := eIEC870_COT_ACT_TERM; (* send activation termination *)
        fbBuffer.TxAddObj( bOnline := bOnline, putObj := txAsdu, buffer := buffer ); (* put asdu to
the TX fifo *)
        IF NOT fbBuffer.bOk THEN
            EXIT;
            (* TODO: Report send buffer overflow error *)
        END_IF

        ELSE
            (* TODO: Report invalid asdu type...*)
            EXIT;
        END_CASE

    END_IF
UNTIL NOT fbBuffer.bOk (* RX fifo is empty *)
END_REPEAT



(* Offline frames are written to the file. Execute this function block in every cycle! *)
fbBuffer(bOnline := bOnline, buffer:= buffer );
IF fbBuffer.status.eState = eIEC870_FBUFFER_ERROR THEN
    (*TODO: Report file access error *)
    ;
END_IF
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

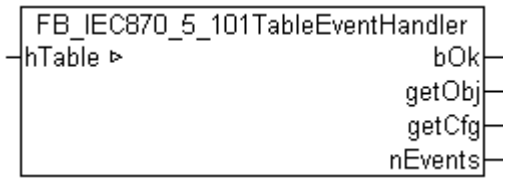
Also see about this

- ST_IEC870_5_101FBufferStatus [▶ 327]
- ST_IEC870_5_101AOGen [▶ 314]

5.1.4 FB_IEC870_5_101TableEventHandler

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

Reading of the events is optional, i.e., the PLC application does not necessarily have to instance this block. This functionality is currently only supported by the IEC 60870-5-104 master!



This function block enables the PLC application to detect certain changes in the IEC application object database and respond accordingly, if required. The changes are referred to as events. Each event type is managed in a separate internal list. The PLC application can read the pending events from one of the lists by calling the block actions. The events are counted internally since several events may occur during a PLC cycle. The counter is incremented whenever an event occurs. At the block output only the last value and the counter reading is issued.

The following events are registered by the function block:

- **OnCreate events** are reported whenever a new application object (single point, double point, measured value...) was added to the application database.
- **OnChange events** are reported when an application object is received by the lower transport layer (Rx frames) or sent (Tx frames), irrespective of whether the value of the information object has changed or not. For a direct command, e.g., C_SC_NA_1, in control direction (master->slave), events are usually reported for the following causes of transmission: eIEC870_COT_ACT (activation), eIEC870_COT_ACT_CON (confirmation of activation) and eIEC870_COT_TERM (completion of activation). For a data point in monitoring direction (slave->master), e.g., M_SP_NA_1, events may be reported for the following causes of transmission: eIEC870_COT_SPONTAN, eIEC870_COT_INROGEN, eIEC870_COT_BACKGROUND etc.

The function block features two tasks:

- **RemoveOnCreateEvent** (reads an entry from the OnCreate event list);
- **RemoveOnChangeEvent** (reads an entry from the OnChange event list);

VAR_IN_OUT

```
VAR_IN_OUT
    hTable : T_HAODBTable;
END_VAR
```

hTable: Application object database handle [▶ 343] (hash table handle). The table handle must be initialized once with the function F_iecCreateTableHnd [▶ 295] before it can be used.

VAR_OUTPUT

```
VAR_OUTPUT
    bOk     : BOOL := FALSE;
    getObj  : ST_IEC870_5_101AOGen;
    getCfg  : ST_IEC870_5_101AOCfg;
    nEvents : DWORD := 0;
END_VAR
```

bOk: This variable becomes TRUE if a new event was read successfully. If FALSE, the event list read last is empty.

getObj: The current value of the data unit [▶ 314] (ASDU).

getCfg: The current configuration [▶ 314] parameters of the data unit (ASDU).

nEvents: Event counter (multiplier). Value range: (0 to 16#FFFFFFFF). Incrementation stops when the maximum value is reached.

Example in ST:

In the following program section, the pending events are read via REPEAT loops and written to the Windows Application log. The associated data points have already been configured as hash table entries. See function description: F_iecAddTableEntry [▶ 296].

```
PROGRAM P_LogEvents
VAR_IN_OUT
    hTable : T_HAODBTable;
END_VAR
VAR
    fbHandler : FB_IEC870_5_101TableEventHandler;
END_VAR
```

```
REPEAT
    fbHandler.RemoveOnChangeEvent( hTable := hTable );
    IF fbHandler.bOk THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
            'RemoveOnChangeEvent(), IOA: %s',
            DWORD_TO_STRING( fbHandler.getObj.info.objAddr ) );
    END_IF
UNTIL NOT fbHandler.bOk
END_REPEAT


REPEAT
    fbHandler.RemoveOnCreateEvent( hTable := hTable );
    IF fbHandler.bOk THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
            'RemoveOnCreateEvent(), IOA: %s',
            DWORD_TO_STRING( fbHandler.getObj.info.objAddr ) );
    END_IF
UNTIL NOT fbHandler.bOk
END_REPEAT
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2 Functions

5.2.1 F_iecInitAOEntry

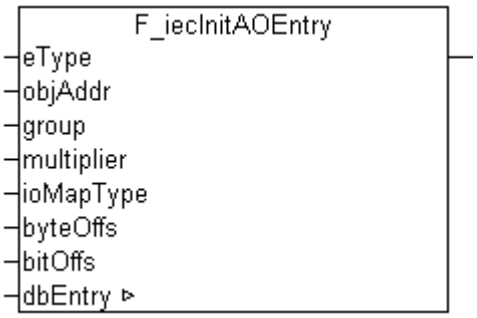


Fig. 1: F_iecInitAOEntry

The function F_iecInitAOEntry configures application objects (Single Points, Double Points, Measured Values...) as linear table entries. The table element (array element) to be configured has to be given to the function as VAR_IN_OUT function parameter.

FUNCTION F_iecInitAOEntry: UDINT

```
VAR_INPUT
    eType       : E_IEC870_5_101TcTypeID := ASDU_TYPEUNDEF;
    objAddr     : DWORD := 0;
    group       : DWORD := 0;
    multiplier  : BYTE := 0;
    ioMapType   : E_IEC870_5_101IOMappingType := MAP_AREA_NONE;
    byteOffs    : UDINT := 0;
    bitOffs     : UDINT := 0;
END_VAR
VAR_IN_OUT
    dbEntry     : ST_IEC870_5_101AODBEntry;
END_VAR
```

eType: Type of application object, ASDU identifier [▶ 327] (e.g..: M_SP_NA_1 for Single Point or M_DP_NA_1 for Double Point).

objAddr : Object address, any address can be given.

group: Object group configuration flags. See: Description of all Group Flags [▶ 348]. The flags can be combined with OR-Linking. Not all combinations are sensefull!

multiplier: Base cycle time multiplicator for cyclic/periodic data transfer. 0 = disabled. The base cycle time can be configured by the *tPerCyclicBase*- parameter in the system parameters [▶ 317].

ioMapType: TwinCAT PLC process data range. This parameter [▶ 331] defines how the TwinCAT PLC and IEC applicatation object process data are to be mapped.

byteOffs: TwinCAT PLC process data byte offset.

bitOffs: TwinCAT PLC process data bit offset.

dbEntry: Application object [▶ 313] to be configured (array element).

Return value	Description
0	No error.
<> 0	Error: IEC60870-5-10x error code

Example in ST:

Configuration of three datapoints as linear table entries.

eType	objAddr	group	multiplier	ioMapType	byteOffs	bitOffs
M_SP_NA_1	100	IEC870_GRP_INRO1	0	MAP_AREA_MEMORY	100	0
M_DP_NA_1	200	IEC870_GRP_INROGEN	0	MAP_AREA_DATA	200	0
M_IT_NA_1	800	IEC870_GRP_REQCOGEN	0	MAP_AREA_MEMORY	800	0

```
VAR_GLOBAL CONSTANT
    MIN_TABLE_IDX : INT := 0;
    MAX_TABLE_IDX : INT := 49;
END_VAR
```

```
PROGRAM P_LinearTableConfig
VAR_IN_OUT
    AODB : ARRAY[MIN_TABLE_IDX..MAX_TABLE_IDX] OF ST_IEC870_5_101AODBEntry;
END_VAR
VAR
    init        : BOOL := TRUE;
    initError       : UDINT;
END_VAR
```

```
IF init THEN
    init := FALSE;
    initError := F_iecInitAOEntry(  eType       := M_SP_NA_1,
                    objAddr         := 100,
                    group       := IEC870_GRP_INRO1,
                    multiplier      := 0,
                    ioMapType       := MAP_AREA_MEMORY,
                    byteOffs    := 100,
                    bitOffs         := 0,
                    dbEntry         := AODB[0] );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'F_iecInitAOEntry() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF

    initError := F_iecInitAOEntry(  eType       := M_DP_NA_1,
                    objAddr         := 200,
                    group       := IEC870_GRP_INROGEN,
                    multiplier      := 0,
                    ioMapType       := MAP_AREA_DATA,
                    byteOffs    := 200,
                    bitOffs         := 0,
                    dbEntry         := AODB[1] );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'F_iecInitAOEntry() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF

    initError := F_iecInitAOEntry(  eType       := M_IT_NA_1,
                    objAddr         := 800,
                    group       := IEC870_GRP_REQCOGEN,
                    multiplier      := 0,
                    ioMapType       := MAP_AREA_MEMORY,
                    byteOffs    := 800,
                    bitOffs         := 0,
                    dbEntry         := AODB[2] );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'F_iecInitAOEntry() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF

END_IF
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.2 F_iecSetAOQuality

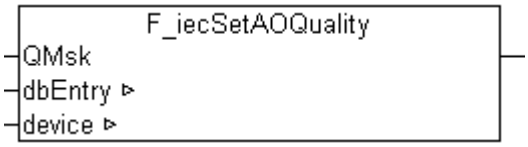


Fig. 2: F_iecSetAOQuality

This function sets / resets the quality flags of an application object to a defined value.

FUNCTION F_iecSetAOQuality: UDINT

```
VAR_INPUT
    QMsk : DWORD;
END_VAR
VAR_IN_OUT
    dbEntry : ST_IEC870_5_101AODBEntry;
    device  : ST_IEC870_5_101DeviceInterface;
END_VAR
```

QMsk : Quality-Flags. The Quality-Flags can be combined with OR Linking. If QMsk = zero, no flags are set/ reset. See: List of available quality-flags [▶ 349].

dbEntry: Application object [▶ 313] whose status of the Quality-Flags are to be set.

device: Communication interface [▶ 319] of IEC device.

Return parameter	Description
0	No error
<> 0	error: IEC60870-5-10x error code

Example for a call in ST:

```
PROGRAM MAIN
VAR
    slave1AODB : ARRAY[1..199] OF ST_IEC870_5_101AODBEntry;

    server1 : FB_IEC870_5_104Slave;
...

    bBlock : BOOL;
    bUnblock : BOOL;
    bIsBlocked : BOOL;
END_VAR
```

Program code:

```
IF bBlock THEN
    bBlock := FALSE;
    F_iecSetAOQuality( IECQ_BL_ON, slave1AODB[1], server1.system.device );
END_IF

IF bUnblock THEN
    bUnblock := FALSE;
    F_iecSetAOQuality( IECQ_BL_OFF, slave1AODB[1], server1.system.device );
END_IF

bIsBlocked := SEL( ( F_iecGetAOQuality( slave1AODB[1], server1.system.device ) AND IECQ_BL_ON ) = IE
CQ_BL_ON, FALSE, TRUE );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.3 F_iecGetAOQuality

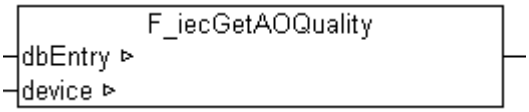


Fig. 3: F_iecGetAOQuality

This function reads the status of the qualitiy flag of an application object.

FUNCTION F_iecGetAOQuality: DWORD

```
VAR_IN_OUT
    dbEntry : ST_IEC870_5_101AODBEntry;
    device  : ST_IEC870_5_101DeviceInterface;
END_VAR
```

dbEntry : Application object [▶ 313] whose qualitiy flags are to be read.

device: Communication interface [▶ 319] to IEC device.

Return parameter	Description
0	error , no qualtiy flag for this application object available.
<> 0	No error. The return parameter delivers the status of the Quality-Flags. See: available Quality-Flags [▶ 349].

Example for a call in ST:

```
PROGRAM MAIN
VAR
    slave1AODB : ARRAY[1..199] OF ST_IEC870_5_101AODBEntry;

    server1 : FB_IEC870_5_104Slave;
...

    bBlock : BOOL;
    bUnblock : BOOL;
    bIsBlocked : BOOL;
END_VAR
```

Program code:

```
IF bBlock THEN
    bBlock := FALSE;
    F_iecSetAOQuality( IECQ_BL_ON, slave1AODB[1], server1.system.device );
END_IF

IF bUnblock THEN
    bUnblock := FALSE;
    F_iecSetAOQuality( IECQ_BL_OFF, slave1AODB[1], server1.system.device );
END_IF
```

```
bIsBlocked := SEL( ( F_iecGetAOQuality( slave1AODB[1], server1.system.device ) AND IECQ_BL_ON ) = IE
CQ_BL_ON, FALSE, TRUE );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.4 F_iecGetAOTimeTag

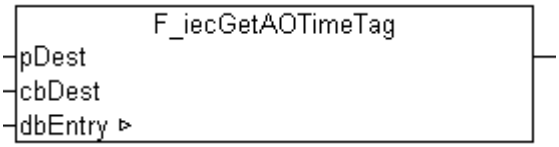


Fig. 4: F_iecGetAOTimeTag

This function reads the current timestamp in a byte buffer of an application object.

FUNCTION F_iecGetAOTimeTag: UDINT

```
VAR_INPUT
    pDest   : DWORD; (* Pointer to time tag destination buffer *)
 cbDest  : UDINT; (* Byte size of time tag destination buffer *)
END_VAR
VAR_IN_OUT
    dbEntry     : ST_IEC870_5_101AODBEntry;
END_VAR
```

pDest : Buffer address.

cbDest: Byte size of the buffer.

dbEntry: Application object [▶ 313] whose timestamp is to be read.

Return parameter	Description
0	Error, no timestamp for this application object available.
<> 0	Number of successfull copied time stamp data bytes. These are 3 bytes for the CP24Time2a time stamp format and 7 bytes for the CP56Time2a time stamp format.

Example for a call in ST:

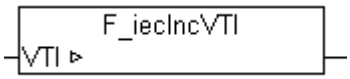
```
PROGRAM MAIN
VAR
(*...*)
    TT1     : T_CP56Time2a;
    TTSize      : UDINT;
    bGetTT      : BOOL;
END_VAR
```

```
IF bGetTT THEN
    bGetTT  := FALSE;
    TTSize  := F_iecGetAOTimeTag( ADR( TT1 ), SIZEOF( TT1 ), slave1AODB[1] );
END_IF
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.5 F_iecIncVTI



This function increments the INT7 regulating step value. Transient bit is not changed!

FUNCTION F_iecIncVTI: BOOL

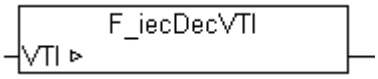
```
VAR_IN_OUT
    VTI : BYTE;
END_VAR
```

VTI: Byte to be incremented.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.6 F_iecDecVTI



This function decrements the INT7 regulating step value. The transient flag is not changed!

FUNCTION F_iecDecVTI: BOOL

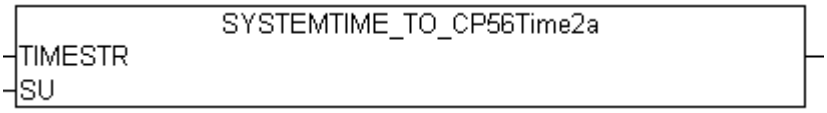
```
VAR_IN_OUT
    VTI     : BYTE;
END_VAR
```

VTI: Byte to be decremented.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.7 SYSTEMTIME_TO_CP56Time2a



The function converts the Windows Systemtime format to the CP56Time2a-Format. All reserved bits are zero.

FUNCTION SYSTEMTIME_TO_CP56Time2a: T_CP56Time2a

T_CP56Time2a [▶ 344]

```
VAR_INPUT
    TIMESTR     : TIMESTRUCT;
    SU          : BOOL;
END_VAR
```

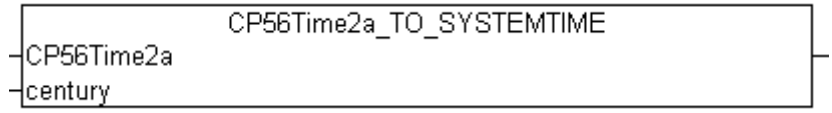
TIMESTR : System time to be converted.

SU: Summer and winter time format. This information is not included in the TIMESTR format. It must be inserted additionally! TRUE = Summer time, FALSE = Winter time.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.8 CP56Time2a_TO_SYSTEMTIME



This function converts the CP56Time2a- time format to the Windows system time format. The SU flag is not used.

FUNCTION CP56Time2a_TO_SYSTEMTIME: TIMESTRUCT

TIMESTRUCT

```
VAR_INPUT
    CP56Time2a      : T_CP56Time2a;
    century         : WORD;
END_VAR
```

CP56Time2a : CP56Time2a format [▶ 344] to be converted.

century: The century (e.g., 20 for the year 2005). This information is not included in the CP56Time2a format, it must be inserted additionally!

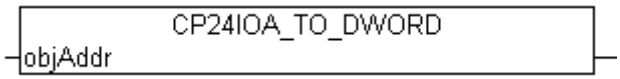
Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Development Environment	Target System	PLC libraries to include
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.9 CP24IOA_TO_DWORD

From product version: TwinCAT PLC Library IEC60870-5-104 substation v3.0.0 / IEC60870-5-104 controlling station v1.0.0 and higher.



The funktion creates a structured TwinCAT object address (3 octets). See: DWORD_TO_CP24IOA [▶ 287].

FUNCTION CP24IOA_TO_DWORD: DWORD

```
VAR_INPUT
    objAddr : T_CP24IOA;
END_VAR
```

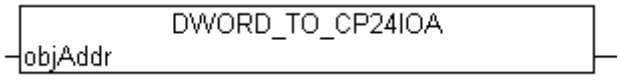
objAddr : Parameter [▶ 345] of structured TwinCAT object addresse.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.10 DWORD_TO_CP24IOA

From product version: TwinCAT PLC Library IEC60870-5-104 substation v3.0.0 / IEC60870-5-104 controlling station v1.0.0 and higher.



The function converts a structured TwinCAT object address into individual address parameters. See also: CP24IOA_TO_DWORD [▶ 287].

FUNCTION DWORD_TO_CP24IOA: T_CP24IOA

T_CP24IOA [▶ 345]

```
VAR_INPUT
    objAddr : DWORD(0..16777215);
END_VAR
```

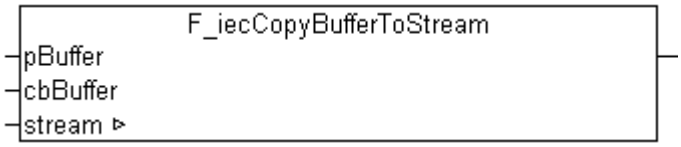
objAddr : structured TwinCAT object address (3 octets).

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.11 F_iecCopyBufferToStream

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 and higher.



This function copies data bytes from an external buffer variable to the *stream* variable.The memory content of the *stream v*ariable increases. The return parameter of the function delivers the number of successfull copied data bytes.

FUNCTION F_iecCopyBufferToStream: UDINT

```
VAR_INPUT
    pBuffer     : DWORD;
    cbBuffer    : UDINT;
END_VAR
VAR_IN_OUT
    stream      : ST_IEC870_5_101Stream;
END_VAR
```

pBuffer: Pointer (address) of external buffer variable.

cbBuffer: Number of data bytes to be copied from the external buffer variable to the *stream* variable.

stream: Data buffer [▶ 316].

Exmple in ST:

The 4 data bytes of the *txBuffer* variable are copied to the *stream* variable by a rising edge at *bTx*.

```
PROGRAM P_CopyBufferToStream
VAR
    stream  : ST_IEC870_5_101Stream;
    txBuffer : ARRAY[0..3] OF BYTE := 1, 2, 3, 4;
    cbTx    : UDINT;
    bTx     : BOOL;
END_VAR
```

```
IF bTx THEN
    bTx     := FALSE;
    cbTx    := cbTx + F_iecCopyBufferToStream( ADR( txBuffer ), SIZEOF( txBuffer ), stream );
    txBuffer[0] := txBuffer[0] + 1;
    txBuffer[1] := txBuffer[1] + 1;
    txBuffer[2] := txBuffer[2] + 1;
    txBuffer[3] := txBuffer[3] + 1;
END_IF
```

Memory description of *stream* variable before the first function call:

length	data											
0	16#00	16#00	16#00	16#00	16#00	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Memory description of *stream* variable after the first function call:

length	data											
4	16#01	16#02	16#03	16#04	...	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Memory description of *stream* variable after the second function call:

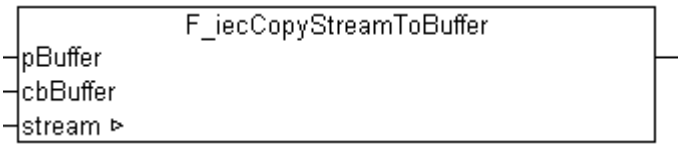
length	data											
8	16#01	16#02	16#03	16#04	16#02	16#03	16#04	16#05	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.12 F_iecCopyStreamToBuffer

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 and higher.



This function copies data bytes from *the stream* variable to an external buffer variable. The memory content of the *stream v*ariable remains unchanged. The return parameter of the function delivers the number of successfull copied data bytes.

FUNCTION F_iecCopyStreamToBuffer: UDINT

```
VAR_INPUT
    pBuffer     : DWORD;
    cbBuffer    : UDINT;
END_VAR
VAR_IN_OUT
    stream  : ST_IEC870_5_101Stream;
END_VAR
```

pBuffer: Pointer (address) of external buffer variable.

cbBuffer: Maximum number of data bytes to be copied from the *stream* variable to the external buffer.

stream: Data buffer [▶ 316].

Example in ST:

After run 4 data bytes are copied to the *stream* variable. The first 4 data bytes of the *stream* variable are copied to the *rxBuffer* variable by a rising edge at bRx .

```
PROGRAM P_CopyStreamToBuffer
VAR
    stream  : ST_IEC870_5_101Stream;
    txBuffer    : ARRAY[0..3] OF BYTE := 1, 2, 3, 4;
    cbTx    : UDINT;
    bTx     : BOOL := TRUE;

    rxBuffer    : ARRAY[0..3] OF BYTE;
    cbRx    : UDINT;
    bRx     : BOOL;
END_VAR
```

```
IF bTx THEN
    bTx     := FALSE;
    cbTx    := cbTx + F_iecCopyBufferToStream( ADR( txBuffer ), SIZEOF( txBuffer ), stream );
    txBuffer[0] := txBuffer[0] + 1;
    txBuffer[1] := txBuffer[1] + 1;
    txBuffer[2] := txBuffer[2] + 1;
    txBuffer[3] := txBuffer[3] + 1;
END_IF

IF bRx THEN
    bRx     := FALSE;
    cbRx    := F_iecCopyStreamToBuffer( ADR( rxBuffer ), SIZEOF( rxBuffer ), stream );
END_IF
```

Memory description of *stream* variable after run:

length	data											
4	16#01	16#02	16#03	16#04	...	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Memory description of *stream* variable after first and each further F_CopyStreamToBuffer function call:

length	data											
4	16#01	16#02	16#03	16#04	...	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.13 F_iecCopyStreamToStream

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 and higher.

This function copies data bytes from the *source* variable to the *target* variable. The memory content of *source* variable remains unchanged. The memory content of the *target* variable increases. The return parameter of the function delivers the number of successfully copied data bytes.

FUNCTION F_iecCopyStreamToStream: UDINT

```
VAR_INPUT
    cbCopy      : UDINT;
END_VAR
VAR_IN_OUT
    target      : ST_IEC870_5_101Stream;
    source      : ST_IEC870_5_101Stream;
END_VAR
```

cbCopy: Number of bytes to be copied from the *source* variable to the *target* variable.

target: Target data buffer [▶ 316]

source: Source data buffer

Example in ST:

At a rising edge at *bCopy* first the *srcValue* is incremented and copied to *srcStream* . After that the first 4 data bytes are copied from *srcStream* to *dstStream.* At the end the first 4 data bytes are copied from *dstStream* into the *dstValue* variable.

```
PROGRAM P_iecCopyStreamToStream
VAR
    srcStream       : ST_IEC870_5_101Stream;
    srcValue    : DWORD;

    dstStream       : ST_IEC870_5_101Stream;
    dstValue    : DWORD;

    bCopy       : BOOL;
END_VAR
```

```
IF bCopy THEN
    bCopy := FALSE;
    srcValue := srcValue + 1;

    F_iecCopyBufferToStream( ADR( srcValue ), SIZEOF( srcValue ), srcStream );

    F_iecCopyStreamToStream( SIZEOF( srcValue ), dstStream, srcStream );

    F_iecCopyStreamToBuffer( ADR( dstValue ), SIZEOF( dstValue ), dstStream );

END_IF
```

Memory description of *srcStream* and *dstStream* variable after the first F_iecCopyStreamToStream function call:

Table 1: srcStream:

length	data											
4	16#01	16#00	16#00	16#00	...	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Table 2: dstStream:

length	data										
4	16#01	16#00	16#00	16#00	...	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Memory description of *srcStream* and *dstStream* variable after thesecond F_iecCopyStreamToStream function call:

Table 3: srcStream:

length	data										
8	16#01	16#00	16#00	16#00	16#02	16#00	16#00	16#00	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Table 4: dstStream:

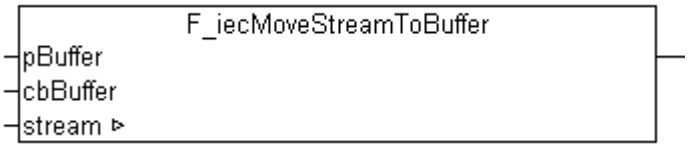
length	data										
8	16#01	16#00	16#00	16#00	16#01	16#00	16#00	16#00	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.14 F_iecMoveStreamToBuffer

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 and higher.



This function copies data bytes from *the stream* variable to an external buffer variable and deletes afterwards the copied data bytes from the *stream* variable. The memory content of the *stream* variable decreases . The return parameter of the function delivers the number of successfully copied data bytes.

FUNCTION F_iecMoveStreamToBuffer : UDINT

```
VAR_INPUT
    pBuffer     : DWORD;
    cbBuffer    : UDINT;
END_VAR
VAR_IN_OUT
    stream  : ST_IEC870_5_101Stream;
END_VAR
```

pBuffer: Pointer (address) of external buffer variable.

cbBuffer: Maximum number of data bytes to be copied from the *stream* variable to the external buffer.

stream: Data buffer [▶ 316].

Example in ST:

After run two DWORD values are copied to the *stream* variable. Each time 4 data bytes of the *stream* variable are copied to the *rxBuffer* variable by a rising edge at bRx .

```
PROGRAM P_MoveStreamToBuffer
VAR
    stream  : ST_IEC870_5_101Stream;
    txBuffer    : ARRAY[0..1] OF DWORD := 16#12345678, 16#ABCDEF01;
    cbTx    : UDINT;
    bTx     : BOOL := TRUE;

    rxBuffer    : DWORD;
    cbRx    : UDINT;
    bRx     : BOOL;
END_VAR
```

```
IF bTx THEN
    bTx     := FALSE;
    cbTx    := F_iecCopyBufferToStream( ADR( txBuffer ), SIZEOF( txBuffer ), stream );
END_IF

IF bRx THEN
    bRx     := FALSE;
    cbRx    := F_iecMoveStreamToBuffer( ADR( rxBuffer ), SIZEOF( rxBuffer ), stream );
END_IF
```

Memory description of *stream* variable after run:

length	data											
8	16#78	16#56	16#34	16#12	16#01	16#EF	16#CD	16#AB	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Memory description of *stream* variable after the first F_iecMoveStreamToBuffer function call:

length	data											
4	16#01	16#EF	16#CD	16#AB	16#01	16#EF	16#CD	16#AB	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Memory description of *stream* variable after the second F_iecMoveStreamToBuffer function call:

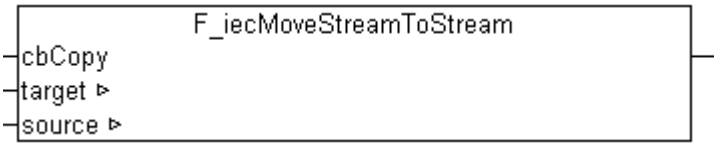
length	data											
0	16#01	16#EF	16#CD	16#AB	16#01	16#EF	16#CD	16#AB	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.15 F_iecMoveStreamToStream

From product version TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 and higher.



This function copies data bytes from the *source* variable to the target variable and deletes afterwards the copied data bytes from the *source* variable. The memory content of the *source* variable decreases. The memory content of the *target* variable increases

FUNCTION F_iecMoveStreamToStream : UDINT

```
VAR_INPUT
    cbCopy   : UDINT;
END_VAR
VAR_IN_OUT
    target   : ST_IEC870_5_101Stream;
    source   : ST_IEC870_5_101Stream;
END_VAR
```

cbCopy: Number of data bytes to be copied from the *source* variable to the *target* variable.

target: Target data buffer [▶ 316]

source: Source data buffer.

Example in ST:

under construction

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.16 F_iecResetStream

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 and higher.

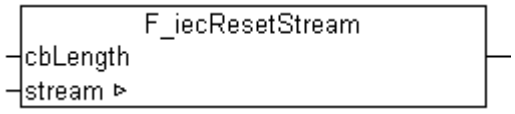


Fig. 5: F_iecResetStream

The function initialises ans resets the *stream* variable. As an option the internal data buffer of the *stream* variable can be initialised with a defined number of zero bytes. The return value of the function delivers the number of successfull initialised zero bytes.

FUNCTION F_iecResetStream: UDINT

```
VAR_INPUT
    cbLength : UDINT; (* number of init data bytes *)
END_VAR
VAR_IN_OUT
    stream  : ST_IEC870_5_101Stream;
END_VAR
```

cbLength: Number of data bytes to be initialised.

stream: Variable [▶ 316] to be initialized.

Example in ST:

After run the internal buffer odf the *stream* variable is reset and initialised with 5 zero bytes.

```
PROGRAM P_ResetStream
VAR
    stream : ST_IEC870_5_101Stream;
    bReset : BOOL := TRUE;
END_VAR
```

```
IF bReset THEN
    bReset := FALSE;
    F_iecResetStream( 5, stream );
END_IF
```

Memory description of stream variable after run:

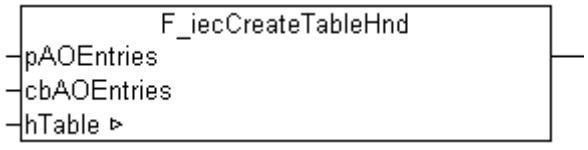
length	data											
5	16#00	16#00	16#00	16#00	16#00	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically

5.2.17 F_iecCreateTableHnd

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.



The function F_iecCreateTableHnd initializes the application object data base handle (hash tabele handle). The table handle has to be initialized once before using.

FUNCTION F_iecCreateTableHnd: UDINT

```
VAR_INPUT
    pAOEntries      : POINTER TO ST_IEC870_5_101AODBEntry := 0;
    cbAOEntries     : UDINT := 0;
```

```
END_VAR
VAR_IN_OUT
    hTable      : T_HAODBTable;
ENd_VAR
```

pAOEntries: Address of application object data base variable.

cbAOEntries: Byte size of application object data base variable.

hTable: Application object data base handle [▶ 343] (hash table handle) to be initialized.

Return parameter	Description
0	No error
<> 0	Error: IEC60870-5-10x error code

Example in ST:

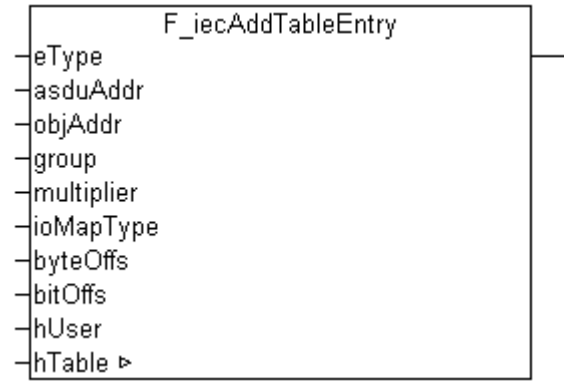
See description of F_iecAddTableEntry [▶ 296]

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.18 F_iecAddTableEntry

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.



The function F_iecAddTableEntry configures the application objects (Single Points, Double Points, Measured Values...) in the application data base as hash table entries. The function searches automatically a free, not used table element (array element) und sets whose configuration parameter.

FUNCTION F_iecAddTableEntry: UDINT

```
VAR_INPUT
    eType       : E_IEC870_5_101TcTypeID := ASDU_TYPEUNDEF;
    asduAddr    : DWORD := 0;
    objAddr     : DWORD := 0;
    group       : DWORD := 0;
    multiplier  : BYTE := 0;
    ioMapType   : E_IEC870_5_101IOMappingType := MAP_AREA_NONE;
    byteOffs    : UDINT := 0;
    bitOffs     : UDINT := 0;
    hUser       : DWORD := 0;
END_VAR
VAR_IN_OUT
    hTable      : T_HAODBTable;
ENd_VAR
```

eType: Application object type, ASDU identifier [▶ 327] (e.g.: M_SP_NA_1 for Single-Point or M_DP_NA_1 for Double-Point).

asduAddr: Shared ASDU address.

objAddr: Object address, arbitrary.

group: Object group configuration flag (description) [▶ 348]. The flags can be combined with OR connection. Note: Not all combinations make sense!

multiplier: Base cycle time multiplicator for cyclic/periodic data transfer. 0 = Disabled. The base cycle time can be configured by the *tPerCyclicBase* parameter in the system parameter [▶ 317].

ioMapType: TwinCAT PLC process data range. This parameter [▶ 331] defines the mapping of TwinCAT PLC and IEC application object process data.

byteOffs: TwinCAT PLC process data byte offset.

bitOffs: TwinCAT PLC process data bit offset.

hUser: Free definable 32 bit value. This value is stored in the configuration data of the application object.

hTable: Application object data base handle [▶ 343] (hash table handle). The table handle has to be initialized once before using with the function F_iecCreateTableHnd [▶ 295].

Return parameter	Description
0	No error
<> 0	Error: IEC60870-5-10x error code

Example in ST:

Three data points are inserted to the application data base as hash table entries:

eType	asduAddr	objAddr	group	multiplier	ioMap-Type	byteOffs	bitOffs	hUser
M_SP_NA_1	11	100	IEC870_GRP_INRO1	0	MAP_AREA_MEMORY	100	0	16#00BECF01
M_DP_NA_1	11	200	IEC870_GRP_INROGEN	0	MAP_AREA_DATA	200	0	16#00BECF02
M_IT_NA_1	11	800	IEC870_GRP_REQCOGEN	0	MAP_AREA_MEMORY	800	0	16#00BECF03

```
VAR_GLOBAL CONSTANT
    MIN_TABLE_IDX : INT := 0;
    MAX_TABLE_IDX : INT := 49;
END_VAR
```

```
PROGRAM P_HashTableConfig
VAR_IN_OUT
    hTable  : T_HAODBTable;
    AODB    : ARRAY[MIN_TABLE_IDX..MAX_TABLE_IDX] OF ST_IEC870_5_101AODBEntry;
END_VAR
VAR
    init    : BOOL := TRUE;
    initError : UDINT;
END_VAR
```

```
IF init THEN
    init := FALSE;
    initError := F_iecCreateTableHnd( ADR( AODB ), SIZEOF( AODB ), hTable );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
            'F_iecCreateTableHnd() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
```

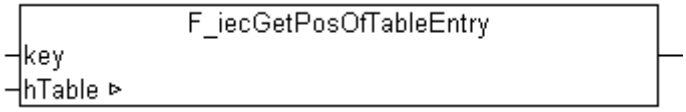
```
    END_IF
    initError := F_iecAddTableEntry(    eType       := M_SP_NA_1,
                    asduAddr    := 11,
                    objAddr     := 100,
                    group       := IEC870_GRP_INRO1,
                    multiplier      := 0,
                    ioMapType       := MAP_AREA_MEMORY,
                    byteOffs    := 100,
                    bitOffs         := 0,
                    hUser       := 16#00BECF01,
                    hTable      := hTable );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'F_iecAddTableEntry() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF

    initError := F_iecAddTableEntry(    eType       := M_DP_NA_1,
                    asduAddr    := 11,
                    objAddr         := 200,
                    group       := IEC870_GRP_INROGEN,
                    multiplier      := 0,
                    ioMapType       := MAP_AREA_DATA,
                    byteOffs    := 200,
                    bitOffs         := 0,
                    hUser       := 16#00BECF02,
                    hTable      := hTable );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'F_iecAddTableEntry() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF

    initError := F_iecAddTableEntry(    eType       := M_IT_NA_1,
                    asduAddr    := 11,
                    objAddr         := 800,
                    group       := IEC870_GRP_REQCOGEN,
                    multiplier      := 0,
                    ioMapType       := MAP_AREA_MEMORY,
                    byteOffs    := 800,
                    bitOffs         := 0,
                    hUser       := 16#00BECF03,
                    hTable      := hTable );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'F_iecAddTableEntry() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF
END_IF
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.19 F_iecGetPosOfTableEntry

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.



The function F_iecGetPosOfTableEntry supplies the table index (array index) for a hash table entry that matches the lookup key. The first array element has position number one (non-zero-based array position).

FUNCTION F_iecGetPosOfTableEntry: UDINT

```
VAR_INPUT
    key : ST_IEC870_5_101HashTableKey;
END_VAR
VAR_IN_OUT
    hTable  : T_HAODBTable;
ENd_VAR
```

key: Lookup key [▶ 325].

hTable: Application object database handle [▶ 343] (hash table handle). The table handle must be initialised once with the function F_iecCreateTableHnd [▶ 295] before it can be used.

Return parameter	Description
0	No table entry matching the key was found.
<> 0	No error. The return parameter supplies the required table index (non-zero-based array position).

Example in ST:
The system searches for the linear table index for three data points. The associated data points have already been configured as hash table entries. See function description: F_iecAddTableEntry [▶ 296].

eType	objAddr	asduAddr	group
M_SP_NA_1	100	11	IEC870_GRP_INRO1
M_DP_NA_1	200	11	IEC870_GRP_INROGEN
M_IT_NA_1	800	11	IEC870_GRP_REQCOGEN

```
VAR_GLOBAL CONSTANT
    MIN_TABLE_IDX : INT := 0;
    MAX_TABLE_IDX : INT := 49;
END_VAR
```

```
PROGRAM P_GetPosOfTableEntry
VAR_IN_OUT
    hTable  : T_HAODBTable;
    AODB    : ARRAY[MIN_TABLE_IDX..MAX_TABLE_IDX] OF ST_IEC870_5_101AODBEntry;
END_VAR
VAR
    bGetPos     : BOOL;
    position    : UDINT;
    key     : ST_IEC870_5_101HashTableKey;
    hUser   : UDINT;
END_VAR
```

```
IF bGetPos      THEN
    bGetPos         := FALSE;

    key.eType       := M_SP_NA_1;
    key.asduAddr    := 11;
    key.objAddr     := 100;
    key.group       := IEC870_GRP_INRO1;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    position    := F_iecGetPosOfTableEntry( key := key, hTable := hTable );
    IF position <> 0 THEN
        hUser := AODB[MIN_TABLE_IDX+position-1].aObj.cfg.hUser;
    ELSE
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
        'F_iecGetPosOfTableEntry() error: %s', '' );
    END_IF

    key.eType       := M_DP_NA_1;
    key.objAddr     := 200;
    key.group       := IEC870_GRP_INROGEN;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    position    := F_iecGetPosOfTableEntry( key := key, hTable := hTable );
    IF position <> 0 THEN
        hUser := AODB[MIN_TABLE_IDX+position-1].aObj.cfg.hUser;
    ELSE
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
        'F_iecGetPosOfTableEntry() error: %s', '' );
    END_IF
```

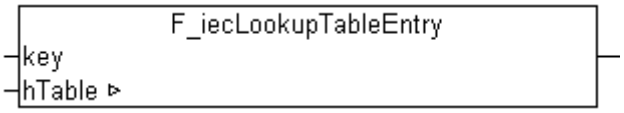
```
    key.eType       := M_IT_NA_1;
    key.objAddr     := 800;
    key.group       := IEC870_GRP_REQCOGEN;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    position    := F_iecGetPosOfTableEntry( key := key, hTable := hTable );
    IF position <> 0 THEN
        hUser := AODB[MIN_TABLE_IDX+position-1].aObj.cfg.hUser;
    ELSE
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
            'F_iecGetPosOfTableEntry() error: %s', '' );
    END_IF
END_IF
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.20 F_iecLookupTableEntry

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.



The function F_iecLookupTableEntry checks if an table entry already exists that matches to the key.

FUNCTION F_iecLookupTableEntry: UDINT

```
VAR_INPUT
    key : ST_IEC870_5_101HashTableKey;
END_VAR
VAR_IN_OUT
    hTable  : T_HAODBTable;
ENd_VAR
```

key: Lookup key [▶ 325].

hTable: Application object data base handle [▶ 343] (hash table handle). The table handle has to be initialized once before using with the function F_iecCreateTableHnd [▶ 295].

Return parameter	Description
0	No error. A table entry that matches to the key exists.
<> 0	No table entry found. Error: IEC60870-5-10x error code

Example in ST:

It will be checked if three data points in the application data base exist. The searched data points have already been configured as hash table entries.
See description of: F_iecAddTableEntry [▶ 296].

eType	objAddr	asduAddr	group
M_SP_NA_1	100	11	IEC870_GRP_INRO1
M_DP_NA_1	200	11	IEC870_GRP_INROGEN
M_IT_NA_1	800	11	IEC870_GRP_REQCOGEN

```
PROGRAM P_LookupEntry
VAR_IN_OUT
    hTable : T_HAODBTable;
END_VAR
VAR
    key : ST_IEC870_5_101HashTableKey;
    bLookup : BOOL;
    nFound : BYTE;
    nError : UDINT;
END_VAR
```

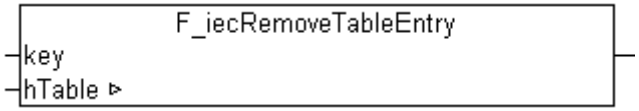
```
IF bLookup THEN
    bLookup := FALSE;

    key.eType       := M_SP_NA_1;
    key.asduAddr    := 11;
    key.objAddr     := 100;
    key.group       := IEC870_GRP_INRO1;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    nError      := F_iecLookupTableEntry( key := key, hTable := hTable );
    IF nError <> 0 THEN
        RETURN;
    ELSE
        nFound := nFound + 1;
    END_IF

    key.eType       := M_DP_NA_1;
    key.asduAddr    := 11;
    key.objAddr     := 200;
    key.group       := IEC870_GRP_INROGEN;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    nError      := F_iecLookupTableEntry( key := key, hTable := hTable );
    IF nError <> 0 THEN
        RETURN;
    ELSE
        nFound := nFound + 1;
    END_IF

    key.eType       := M_IT_NA_1;
    key.asduAddr    := 11;
    key.objAddr     := 800;
    key.group       := IEC870_GRP_REQCOGEN;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    nError      := F_iecLookupTableEntry( key := key, hTable := hTable );
    IF nError <> 0 THEN
        RETURN;
    ELSE
        nFound := nFound + 1;
    END_IF
END_IF
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included atomatically)

5.2.21 F_iecRemoveTableEntry

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.



The function F_iecRemoveTableEntry removes an hash table entry that matches to the key.

FUNCTION F_iecRemoveTableEntry: UDINT

```
VAR_INPUT
    key : ST_IEC870_5_101HashTableKey;
END_VAR
VAR_IN_OUT
    hTable  : T_HAODBTable;
ENd_VAR
```

key: Lookup key [▶ 325].

hTable: Application object data base handle [▶ 343] (hash table handle). The table handle has to be initialized once before using with the function F_iecCreateTableHnd [▶ 295].

Return parameter	Description
0	No error. The table entry has been removed successfully.
<> 0	Error: IEC60870-5-10x error code

Example in ST:

Three hash table entries will be romoved from the application data base. The searched data points have already been configured as hash table entries.
See description of: F_iecAddTableEntry [▶ 296].

eType	objAddr	asduAddr	group
M_SP_NA_1	100	11	IEC870_GRP_INRO1
M_DP_NA_1	200	11	IEC870_GRP_INROGEN
M_IT_NA_1	800	11	IEC870_GRP_REQCOGEN

```
PROGRAM P_RemoveEntry
VAR_IN_OUT
    hTable : T_HAODBTable;
END_VAR
VAR
    key : ST_IEC870_5_101HashTableKey;
    bRemove : BOOL;
    nError : UDINT;
END_VAR
```

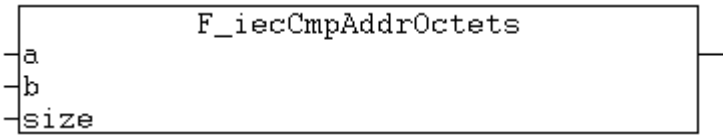
```
IF bRemove THEN
    bRemove         := FALSE;

    key.eType       := M_SP_NA_1;
    key.asduAddr    := 11;
    key.objAddr     := 100;
    key.group       := IEC870_GRP_INRO1;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    nError      := F_iecRemoveTableEntry( key := key, hTable := hTable );
    IF nError <> 0 THEN
        RETURN;
    END_IF

    key.eType       := M_DP_NA_1;
    key.asduAddr    := 11;
    key.objAddr     := 200;
    key.group       := IEC870_GRP_INROGEN;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    nError      := F_iecRemoveTableEntry( key := key, hTable := hTable );
    IF nError <> 0 THEN
        RETURN;
    END_IF

    key.eType       := M_IT_NA_1;
    key.asduAddr    := 11;
    key.objAddr     := 800;
    key.group       := IEC870_GRP_REQCOGEN;
    key.lookup      := IEC870_LOOKUP_KEY_ALL_ON;
    nError      := F_iecRemoveTableEntry( key := key, hTable := hTable );
    IF nError <> 0 THEN
        RETURN;
    END_IF
END_IF
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.22 F_iecCmpAddrOctets

From product version: TwinCAT PLC Library IEC60870-5-101/104 substation v3.0.2 / IEC60870-5-104 controll station v1.0.2 and higher.



This function compares two address values (e.g. link addresses, object addresses or common ASDU addresses).

FUNCTION F_iecCmpAddrOctets: BOOL

```
VAR_INPUT
    a    : DWORD; (* first address *)
    b    : DWORD; (* second address *)
    size : INT(0..4); (* address octet size (0..3, 4 = reserved) *)
END_VAR
```

a: First address.

b: Second address

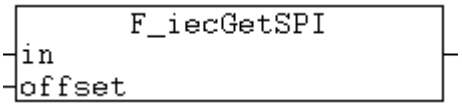
size: Address octet size.

Return value	Description
TRUE	(a and b are equal) or (b is broadcast address) or (size is 0).
FALSE	All other cases.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1307	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.2.23 F_iecGetSPI



This function decodes the single point information from byte variable. One byte variable yields up to 8 single points information. Every single point information occupies 1 bit.

FUNCTION F_iecGetSPI: E_IEC870_5_101SPI

E_IEC870_5_101SPI [▶ 342]

```
VAR_INPUT
    in     : BYTE; (* Byte variable from where the single point information have to be decoded *)
    offset : UDINT(0..7); (* Single point information offset *)
END_VAR
```

Example:

Decoding the information of 4 single points from byte variable.

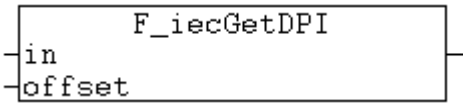
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eSPI           : E_IEC870_5_101SPI;
ENd_VAR
```

```
eSPI := F_iecGetSPI( memarea[0], 0 );
eSPI := F_iecGetSPI( memarea[0], 1 );
eSPI := F_iecGetSPI( memarea[0], 2 );
eSPI := F_iecGetSPI( memarea[0], 3 );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

5.2.24 F_iecGetDPI



This function decodes the double point information from byte variable. One byte variable yields up to 4 double points information. Every double point information occupies 2 bits.

FUNCTION F_iecGetDPI: E_IEC870_5_101DPI

E_IEC870_5_101DPI [▶ 342]

```
VAR_INPUT
    in     : BYTE; (* Byte variable from where the double point information have to be decoded *)
    offset : UDINT(0..3); (* Double point information offset *)
END_VAR
```

Example:

Decoding the information of 4 double points from one byte variable.

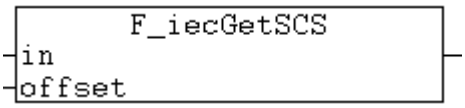
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eDPI : E_IEC870_5_101DPI;
ENd_VAR
```

```
eDPI := F_iecGetDPI( memarea[0], 0 );
eDPI := F_iecGetDPI( memarea[0], 1 );
eDPI := F_iecGetDPI( memarea[0], 2 );
eDPI := F_iecGetDPI( memarea[0], 3 );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

5.2.25 F_iecGetSCS



This function decodes the single command state from byte variable. One byte variable yields up to 8 single command states. Every single command state occupies 1 bit.

FUNCTION F_iecGetSCS: E_IEC870_5_101SCS

E_IEC870_5_101SCS [▶ 337]

```
VAR_INPUT
    in     : BYTE;(* Byte variable from where the single command state have to be decoded *)
    offset : UDINT(0..7); (* Single command state offset *)
END_VAR
```

Example:

Decoding the state of 4 single commands from one byte variable.

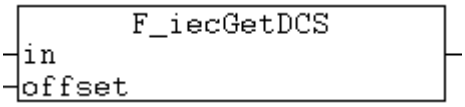
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eSCS : E_IEC870_5_101SCS;
ENd_VAR
```

```
eSCS := F_iecGetSCS( memarea[0], 0 );
eSCS := F_iecGetSCS( memarea[0], 1 );
eSCS := F_iecGetSCS( memarea[0], 2 );
eSCS := F_iecGetSCS( memarea[0], 3 );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

5.2.26 F_iecGetDCS



This function decodes the double command state from byte variable. One byte variable yields up to 4 double commands states. Every double command state occupies 2 bits.

FUNCTION F_iecGetDCS: E_IEC870_5_101DCS

E_IEC870_5_101DCS [▶ 338]

```
VAR_INPUT
    in     : BYTE; (* Byte variable from where the double command state have to be decoded *)
    offset : UDINT(0..3); (* Double command state offset *)
END_VAR
```

Example:

Decoding the state of 4 double commands from byte variable.

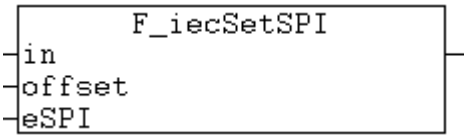
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eDCS : E_IEC870_5_101DCS;
ENd_VAR
```

```
eDCS := F_iecGetDCS( memarea[0], 0 );
eDCS := F_iecGetDCS( memarea[0], 1 );
eDCS := F_iecGetDCS( memarea[0], 2 );
eDCS := F_iecGetDCS( memarea[0], 3 );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

5.2.27 F_iecSetSPI



This function encodes the single point information in the byte variable. One byte variable yields up to 8 single point information. Every single point information occupies 1 bit.

FUNCTION F_iecSetSPI: BYTE

```
VAR_INPUT
    in     : BYTE; (* Byte variable where the new single point information have to be encoded. *)
    offset : UDINT(0..7);(* Single point information offset. *)
    eSPI   : E_IEC870_5_101SPI; (* The new value of single point information. *)
END_VAR
```

Example:

Setting the information of 4 single points to ON.

```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
END_VAR
```

```
memarea[0] := F_iecSetSPI( memarea[0], 0, eIEC870_SPI_ON );
memarea[0] := F_iecSetSPI( memarea[0], 1, eIEC870_SPI_ON );
memarea[0] := F_iecSetSPI( memarea[0], 2, eIEC870_SPI_ON );
memarea[0] := F_iecSetSPI( memarea[0], 3, eIEC870_SPI_ON );
```

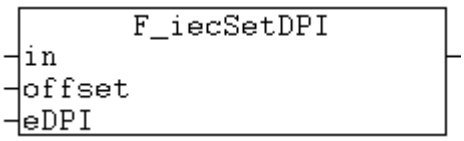
Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

Also see about this

- E_IEC870_5_101SPI [▶ 342]

5.2.28 F_iecSetDPI



This function encodes the double point information in the byte variable. One byte variable yields up to 4 double point information. Every double point information occupies 2 bits.

FUNCTION F_iecSetDPI: BYTE

```
VAR_INPUT
    in     : BYTE; (* Byte variable where the new double point information have to be encoded. *)
    offset : UDINT(0..3);(* Double point information offset. *)
    eDPI   : E_IEC870_5_101DPI; (* The new value of double point information. *)
END_VAR
```

Example:

Setting the information of 4 double points to ON.

```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
END_VAR
```

```
memarea[0] := F_iecSetDPI( memarea[0], 0, eIEC870_DPI_ON );
memarea[0] := F_iecSetDPI( memarea[0], 1, eIEC870_DPI_ON );
memarea[0] := F_iecSetDPI( memarea[0], 2, eIEC870_DPI_ON );
memarea[0] := F_iecSetDPI( memarea[0], 3, eIEC870_DPI_ON );
```

Requirements

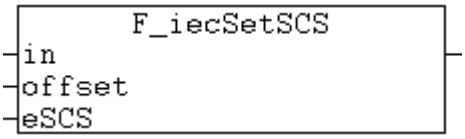
Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib

Development Environment	Target System	PLC libraries to include
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

Also see about this

E_IEC870_5_101DPI [▶ 342]

5.2.29 F_iecSetSCS



This function encodes the single command state in the byte variable. One byte variable yields up to 8 single command states. Every single command state occupies 1 bit.

FUNCTION F_iecSetSCS: BYTE

```
VAR_INPUT
    in     : BYTE; (* Byte variable where the new single command state have to be encoded. *)
    offset : UDINT(0..7);(* Single command state offset. *)
    eSCS   : E_IEC870_5_101SCS; (* The new value of single command state. *)
END_VAR
```

Example:

Setting the state of 4 single command to ON.

```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
END_VAR
```

```
memarea[0] := F_iecSetSCS( memarea[0], 0, eIEC870_SCS_ON );
memarea[0] := F_iecSetSCS( memarea[0], 1, eIEC870_SCS_ON );
memarea[0] := F_iecSetSCS( memarea[0], 2, eIEC870_SCS_ON );
memarea[0] := F_iecSetSCS( memarea[0], 3, eIEC870_SCS_ON );
```

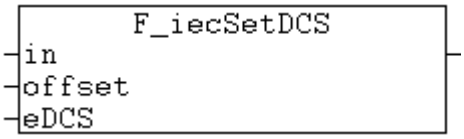
Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

Also see about this

E_IEC870_5_101SCS [▶ 337]

5.2.30 F_iecSetDCS



This function encodes the double command state in the byte variable. One byte variable yields up to 4 double command states. Every double command state occupies 2 bits.

FUNCTION F_iecSetDCS: BYTE

```
VAR_INPUT
    in     : BYTE; (* Byte variable where the new double command state have to be encoded. *)
    offset : UDINT(0..3);(* Double command state offset. *)
    eDCS   : E_IEC870_5_101DCS; (* The new value of double command state *)
END_VAR
```

Example:

Setting the state of 4 doble commands to ON.

```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
END_VAR
```

```
memarea[0] := F_iecSetDCS( memarea[0], 0, eIEC870_DCS_ON );
memarea[0] := F_iecSetDCS( memarea[0], 1, eIEC870_DCS_ON );
memarea[0] := F_iecSetDCS( memarea[0], 2, eIEC870_DCS_ON );
memarea[0] := F_iecSetDCS( memarea[0], 3, eIEC870_DCS_ON );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1324	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

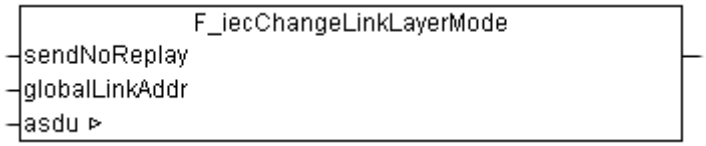
Also see about this

E_IEC870_5_101DCS [▶ 338]

5.2.31 F_iecChangeLinkLayerMode

Starting from the product version:

- **TwinCAT PLC Library IEC60870-5-101 Central Station v2.0.3;**
- **TwinCAT PLC Library IEC60870-5-101 Substation v4.0.3;**
- **TwinCAT PLC Library IEC60870-5-102 Central Station v2.0.3;**
- **TwinCAT PLC Library IEC60870-5-103 Central Station v2.0.3;**



With this function the default behaviour of each individual ASDU frame at the Link-Layer level in the primary station can be set or changed. The ASDU frame to be sent can be marked as a SEND/NO REPLY frame (function 4) or as a broadcast frame. The frames are marked before depositing the frame in the TX buffer I.e. before calling the FB_IEC870_5_101TBufferCtrl [▶ 272].TxAddObj action.

If you do not use the function, then all frames in the primary station are sent as SEND/CONFIRM frames (function 3). In this case the configured station address is used as the link address.

- In unbalanced mode the use of this function has no meaning in the substation. The substation never acts as a primary station in this mode.
- The function has no meaning when using the IEC 6087-5-104 protocol.
- This functionality is supported only when using the ‘low level’ interface.

Important notes:

- In unbalanced mode the use of this function has no meaning in the substation. The substation never acts as a primary station in this mode.
- The function has no meaning when using the IEC 6087-5-104 protocol.
- This functionality is supported only when using the ‘low level’ interface.

FUNCTION F_iecChangeLinkLayerMode: BOOL

```
VAR_INPUT
    sendNoReplay   : BOOL;(* TRUE => Use SEND / NO REPLAY link layer function, FALSE => Use SEND/
CONFIRM function (default) *)
    globalLinkAddr : BOOL;
(* TRUE => Use global (broadcast link address, 0xFF... ), FALSE => Use configured (station) link add
ress *)
END_VAR
VAR_IN_OUT
    asdu : ST_IEC870_5_101AOGen;
END_VAR
```

sendNoReplay : this parameter specifies whether the Link-Layer function SEND/NO REPLY (TRUE = function 4) or SEND/CONFIRM (FALSE = function 3) is to be used when sending the frame;

globalLinkAddr : this parameter specifies whether a global (broadcast) address is to be used instead of the configured station link address when sending the frame. If the value is TRUE, 16#FF or 16#FFFF (size one octet, link address size two octets) is used as the link address in the sent frame;

asdu : The ASDU frame [▶ 314] to be sent as a VAR_IN_OUT variable;

Return parameter	Description
FALSE	Function failed.
TRUE	No error.

Example 1 (excerpt):

The spontaneous data of a bitstring are to be sent to the central station with the aid of the SEND/NO REPLY function (balanced mode).

```
...

(* Send spontanous bitstring data *)
IF ( txQDS <> BITSTRING_QUALITY_100 ) OR ( txBSI <> BITSTRING_100 ) THEN

    txBSI   := BITSTRING_100;
    txQDS   := BITSTRING_QUALITY_100;(* Get quality *)
    txTT    := SYSTEMTIME_TO_CP56Time2a( fbRTC.CDT, TRUE );(* Get current time stamp *)
(* create asdu *)
    txAsdu.ident.eType      := M_BO_TB_1;  (* Bit string with time tag *)
    txAsdu.ident.bSQ        := FALSE;
    txAsdu.ident.nObj       := 1;
    txAsdu.ident.eCOT       := eIEC870_COT_SPONTAN;
    txAsdu.ident.nORG       := sysPara.nOrg;(* Set originator address *)
    txAsdu.ident.bPN        := FALSE;
    txAsdu.ident.bT         := FALSE;
    txAsdu.ident.eClass     := eIEC870_Class_1;(* Put to the high priority tx buffer *)
    txAsdu.ident.asduAddr   := sysPara.asduAddr;(* Set common asdu address *)
    txAsdu.info.objAddr     := 100;(* Set information object address *)
    F_iecResetStream( 0, txAsdu.info.stream ); (* clear previous data (this sets the stream length =
 0 *)
    F_iecCopyBufferToStream( ADR( txBSI ), SIZEOF( txBSI ), txAsdu.info.stream ); (* put BSI to stre
am *)
    F_iecCopyBufferToStream( ADR( txQDS ), SIZEOF( txQDS ), txAsdu.info.stream ); (* put QDS to stre
```

```
am *)
    F_iecCopyBufferToStream( ADR( txTT ), SIZEOF( txTT ), txAsdu.info.stream ); (* put time tag to s
tream *)F_iecChangeLinkLayerMode( TRUE, FALSE, txAsdu );

    fbBuffer.TxAddObj( putObj := txAsdu, buffer := buffer ); (* put asdu to the TX fifo *)F_iecChang
eLinkLayerMode( FALSE, FALSE, txAsdu );

    IF fbBuffer.bOk THEN
        fbLog( put := CONCAT( '<=', IEC101ASDU_TO_STRING(txAsdu) ) );
    ELSE(* Report send buffer overflow error *)
        fbLog( put := 'TX buffer overflow (spontanous bitstring data)!' );
    END_IF
END_IF

...
```

The txAsdu variable is used to send further data points. The default configuration of the txAsdu frame is produced by a further F_iecChangeLinkLayerMode (FALSE, FALSE,…) function call. Other ASDUs are not to be sent as SEND/NO REPLY telegrams.

Example 2 (excerpt):

A single command is to be sent to the substation with the aid of the SEND/NO REPLY function (unbalanced mode).

```
...

(* Send one single command *)
IF SND_SCS_2100 THEN
    SND_SCS_2100 := FALSE; (* Reset flag *)

    txAsdu.ident.eType      := C_SC_NA_1; (* Single command *)
    txAsdu.ident.bSQ        := FALSE;
    txAsdu.ident.nObj       := 1;
    txAsdu.ident.eCOT       := eIEC870_COT_ACT; (* Command activation *)
    txAsdu.ident.nORG       := sysPara.nOrg;(* Set originator address *)
    txAsdu.ident.bPN        := FALSE;
    txAsdu.ident.bT         := FALSE;
    txAsdu.ident.eClass     := eIEC870_Class_1;(* Put to the high priority tx buffer *)
    txAsdu.ident.asduAddr   := sysPara.asduAddr;(* Set common asdu address *)
    txAsdu.info.objAddr     := 2100;(* Set information object address *)
    tmpByte             := INT_TO_BYTE(SCS_2100);(* Set single command state *)
    tmpByte.7           := 0;(* Set select/execute bit *)

    F_iecResetStream( 0, txAsdu.info.stream ); (* Clear previous data (this sets the stream length =
 0 *)
    F_iecCopyBufferToStream( ADR( tmpByte ), SIZEOF( tmpByte ), txAsdu.info.stream ); (* put QCC to
stream *)     F_iecChangeLinkLayerMode( TRUE, FALSE, txAsdu );

    fbBuffer.TxAddObj( putObj := txAsdu, buffer := buffer ); (* put asdu to the TX fifo *)
     F_iecChangeLinkLayerMode( FALSE, FALSE, txAsdu );

    IF fbBuffer.bOk THEN
        timerCON( IN := FALSE );(* Reset timer *)
        timerTERM( IN := FALSE );(* Reset timer *)
        fbLog( put := CONCAT( '<=', IEC101ASDU_TO_STRING(txAsdu) ) );
        state := 80;(* Wait for command confirmation *)
    ELSE(* Report send buffer overflow error *)
        fbLog( put := 'TX buffer overflow (single command)!' );
        state := 1;
    END_IF

END_IF

...
```

Requirements

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.11.0 Build >= 1554	PC or CX (x86, ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib are included automatically)

5.2.32 F_GetVersionTcIEC870_5_101

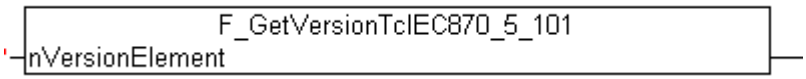


Fig. 6: F_GetVersionTcIEC870_5_101

This function reads version information from the plc library.

FUNCTION F_GetVersionTcIEC870_5_101: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

5.3 Data types

5.3.1 ST_IEC870_5_101TBuffer

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.6 / IEC60870-5-101 substation v2.0.2 and higher.

This data structure is used for data exchange (TX/RX data buffer) via the IEC60870-5-104/101 Transport Interface.

```
TYPE ST_IEC870_5_101TBuffer :
STRUCT
    eDbg        : E_IEC870_5_101FifoDbgFlags :=eIEC870_FIFO_DBG_OFF; (* enable/
disable log view hex output *)
    asduFmt     : ST_IEC870_5_101AsduFmtParams; (* ASDU frame format parameters *)
    asduSize    : BYTE := 0; (* max. length of ASDU data *)
    mode        : DWORD := 0;
    dataLink    : ST_IEC870_5_101DataLink; (* internal tx/rx buffer *)
    bOverwrite  : BOOL := FALSE;(* TRUE = Overwrite oldest entry, FALSE = don't overwrite *)
END_STRUCT
END_TYPE
```

eDbg: Debug [▶ 335] output parameter.

asduFmt: ASDU format [▶ 320] parameter.

asduSize: Maximum length of ASDU data.

mode: Reserved, not used. This value must be zero.

dataLink: The access to the elements of this data structure should only occur with an instance of the function block FB_IEC870_5_101TBufferCtrl [▶ 272], not directly.

The TX/RX data buffer uses internal two send Fifos and one receive Fifo:

1. Class 1 send Fifo with (high prior) data;
2. Class 2 send Fifo with (low prior) data;
3. Receive Fifo (for Class 1 and Class 2 data);

The lower transport functions of the library emtpy first the class 1 Fifo and afterwards the class 2 Fifo. The class 2 data are only sent, if the class 1 Fifo contains no data to be sent.

Each internal Fifo hast a fixed size of 10000 bytes. This is enough for the most applications, because the number of frames which can once be sent is limited by the iK and iW protocol parameters.
According to experience it is possible to store 200 ADSUs with an information element (object), or approx. 20 ADSUs with with a sequence of 100 information elements (objects) in each Fifo.

If a higher number of frames (to be sent or to be received) should be buffered (e.g. ~20000), they can be kept in external buffers/Fifos which are defined by the PLC programmer. The PLC application can refill the TwinCAT send Fifos with own Fifo entries, or empty the TwinCAT receive Fifo.

As an alternative it is possible to use two buffers and to fill/read them alternately and give them to the communication block .

bOverwrite: Implemented from IEC870-5-104 slave library **v3.0.14** and higher. Activates/deactivates the overwriting of the oldest messages in the send buffer if the max. permissible buffer size is exceeded. This parameter is active only in the offline mode, (i.e. if the connection has been interrupted) and if offline data storage has additionally been activated via the **bRetainBuffer** parameter in the protocol parameters (ST_IEC870_5_104ProtPara). In online mode, i.e. if data transfer has been started, no older messages are overwritten (otherwise some intermediate values would possibly be missing).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.2 ST_IEC870_5_101AODBEntry

```
TYPE ST_IEC870_5_101AODBEntry :
STRUCT
    aObj : ST_IEC870_5_101AOEntry;   (* application object *)
    ctrl : FB_IEC870_5_101AOCtrl;    (* application object control function *)
END_STRUCT
END_TYPE
```

An IEC application object database entry. The IEC application object database is declared as an array variable of type ST_IEC870_5_101AODBEntry. The member variables of this structured type are not manipulated (modified) directly, but only via the available functions or function blocks. F_iecIniAOEntry [▶ 280] is part of such a function, for example.

Example for a declaration of an application database with 2001 objects:

```
VAR_GLOBAL
    slave1AODB : ARRAY[0..2000] OF ST_IEC870_5_101AODBEntry;
ENd_VAR
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.3 ST_IEC870_5_101AOGen

From product version: TwinCAT PLC Library IEC60870-5-104 substation v2.0.5 and higher.

Variables of this type represent an ASDU object.

```
TYPE ST_IEC870_5_101AOGen:
STRUCT
    head : ST_IEC870_5_101FifoHead :=(      source := ( link := 0, addr := 0 ),
                    target := ( link := 0, addr := 0 ),
                    ctrl := 0 ); (* Header *)

    ident   : ST_IEC870_5_101DataUnit_Ident := ( eType     := ASDU_TYPEUNDEF,
                          nObj    := 0,
                          bSQ     := FALSE,
                          bT          := FALSE,
                          bPN     := FALSE,
                          nORG    := 0,
                          asduAddr    := 0,
                          eCOT    := eIEC870_COT_UNUSED,
                          eClass  := eIEC870_Class_None ); (* Data unit identifier *)

 info    : ST_IEC870_5_101AOInfoObj      := ( objAddr := 0, stream := ( length := 0 ) ); (* informat
ion object *)
END_STRUCT
END_TYPE
```

head: Header (reserved).

ident: Identification [▶ 315] field in the data unit (ASDU).

info: Information object [▶ 316] /Information element data field.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.4 ST_IEC870_5_101AOCfg

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

ASDU object configuration parameters. These parameters are set during configuration of the data points (F_iecInitAOEntry or F_iecAddTableEntry) and should not be written directly from the PLC application.

```
TYPE ST_IEC870_5_101AOCfg:
STRUCT
    group       : DWORD := IEC870_GRP_INROGEN;
    multiplier  : BYTE := 0;
    opt         : BYTE := 0;
    ioMapType   : E_IEC870_5_101IOMappingType := MAP_AREA_NONE;
    byteOffs    : DWORD := 0;
    bitOffs     : DWORD := 0;
    hUser       : DWORD := 0;
    ext         : DWORD := 0;
END_STRUCT
END_TYPE
```

group: Object group configuration flags. A description of all group flags can be found here [▶ 348]. The flags can be combined with an OR operation. Not all combinations are meaningful!

multiplier: Basic cycle time multiplier for cyclic/periodic data transfer. 0 = deactivated. The basic cycle time can be configured via the *tPerCyclicBase* parameter in the system parameters [▶ 317].

opt: Reserved.

ioMapType: TwinCAT PLC process data area. This parameter [▶ 331] defines the mapping for the TwinCAT PLC and IEC application object process data.

byteOffs: TwinCAT PLC process data byte offset.

bitOffs: TwinCAT PLC process data bit offset.

hUser: User handle. Freely definable 32-bit value.

ext: Reserved.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.5 ST_IEC870_5_101DataUnit_Ident

Identification field of data unit (ASDU).

```
TYPE ST_IEC870_5_101DataUnit_Ident:
STRUCT
    eType    : E_IEC870_5_101TcTypeID; (* TwinCAT ASDU type identifier *)
    bSQ      : BOOL; (* Single/Sequence *)
    nObj     : BYTE; (* Number of information objects or elements *)
    bT       : BOOL; (* Test bit *)
    bPN      : BOOL; (* Positive/Negative confirmation *)
    eCOT     : E_IEC870_5_101COTType; (* Cause of transmission *)
    nORG     : BYTE; (* Originator address (optional) *)
    asduAddr: DWORD; (* Common address of ASDU *)
    eClass   : E_IEC870_5_101ClassType; (* Object class *)
END_STRUCT
END_TYPE
```

eType: Type [▶ 327].

bSQ: Sequence flag.

nObj: Number of information objects or information elements.

bT: Test flag

bPN: Positive/negative confirmation.

eCOT: Cause of transmission [▶ 333].

nORG: Source address.

asduAddr: Common ASDU address

eClass: Priority class [▶ 333].

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.6 ST_IEC870_5_101AOInfoObj

Information object description.

```
TYPE ST_IEC870_5_101AOInfoObj :
STRUCT
    objAddr : DWORD; (* Information object address *)
    stream  : ST_IEC870_5_101Stream; (* Information element of max. length *)
END_STRUCT
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Also see about this

- ST_IEC870_5_101Stream [▶ 316]

5.3.7 ST_IEC870_5_101Stream

Variables of this type are used for data buffer (Stream). In the stream "raw" databytes are buffered in that order they have to be sent /received later. During sending or receiving the data byte zero is sent/received first.

```
TYPE ST_IEC870_5_101Stream :
STRUCT
    length  : DWORD := 0; (* current stream length *)
    data    : ARRAY[0..IEC870_MAX_ASDU_DATA_BYTE] OF BYTE; (* stream data *)
END_STRUCT
END_TYPE
```

length: Current number of data bytes in the stream;

data: Stream data buffer;

Memory description of a stream variable with an intermediate stored DWORD variable with the value: **16#BECF1234**. Please regard the interchanged data bytes in the Intel format!

length	data											
4	16#34	16#12	16#CF	16#BE	...	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Please use the following functions to change the memory content of a stream variable:

Function	Description
F_iecResetStream [▶ 294]	Stream Initialisation /Reset.
F_iecCopyBufferToStream [▶ 288]	Copies data bytes from an external buffer variable to the stream. The memory content of the stream variable increases.
F_iecCopyStreamToBuffer [▶ 289]	Copies data bytes from the stream to an external buffer variable. The memory content of the stream variable remains unchanged.
F_iecCopyStreamToStream [▶ 290]	Copies data bytes from one stream to another stream. The memory content of the source variable remains unchanged. The memory content of the target variable increases.
F_iecMoveStreamToBuffer [▶ 292]	Copies data bytes from the stream to an external buffer variable and deletes afterwards the copied data bytes in the stream. The memory content of the stream variable decreases.
F_iecMoveStreamToStream [▶ 294]	Copies data bytes from the source stream to the target stream an deletes afterwards the copied data bytes in the source stream. The memory content of the source variable decreases and the memory content of the target variable increases.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.8 ST_IEC870_5_101SystemParams

```
TYPE ST_IEC870_5_101SystemParams :
STRUCT
    asduAddr           : DWORD := 11;
    asduAddrRev        : DWORD := 22;
    asduFmt            : ST_IEC870_5_101AsduFmtParams := ( eCOTSize    := eIEC870_COT_TwoOctets,
                             eAsduAddrSize   := eIEC870_AsduAddr_TwoOctets,
                             eObjAddrSize    := eIEC870_ObjAddr_ThreeOctets );
    bEndOfInit         : BOOL := TRUE;

    bSyncTime          : BOOL := TRUE;
    bSyncPCTime        : BOOL := FALSE;

    bUsePCTime         : BOOL := TRUE;
    tSyncToPCTime      : TIME := T#0s;
    sPCTimeNetID       : T_AmsNetID := '';

    bTimeCOT3          : BOOL := FALSE;
    tSyncTimeut        : TIME := T#0m;

    bACTCONRes         : BOOL := TRUE;
    bACTTERMRes        : BOOL := TRUE;

    bPerCyclic         : BOOL := FALSE;
    tPerCyclicBase     : TIME := T#60s;

    bBackScan          : BOOL := FALSE;
    tBackScanCycle     : TIME := T#60s;

    bPerFRZ            : BOOL := FALSE;
    tPerFRZCycle       : TIME := T#60s;

    tSelExecTimeout    : TIME := T#10s;
    tActConTimeout     : TIME := T#5s;
    tActTermTimeout    : TIME := T#30s;
    tDeactConTimeout   : TIME := T#5s;
    tReadResTimeout    : TIME := T#5s;

    dbgMode            : DWORD := 0; (* 0 => disabled,
```

```
                Bit 0 set => IEC870_DEBUGMODE_ASDU => debug asdu's,
                Bit 1 set => IEC870_DEBUGMODE_DEVSTATE => debug device state changes
                Bit 2 set => IEC870_DEBUGMODE_LINKLAYER => link layer frame data *)
    orgAddr           : BYTE := 1; (* Oiginator address, reserved, not used *)

    bOverwrite      : BOOL := FALSE;(* TRUE = Overwrite oldest entries, FALSE = don't overwrite *)
END_STRUCT
END_TYPEEND_TYPE
```

asduAddr: Common ASDU address in standard direction. Default: 11.

asduAddrRev: Common ASDU address in reverse direction. Default: 22.

asduFmt: ASDU format [▶ 320] parameter (e.g. Octet length of the transfer cause, ASDU address and the Object address);

bEndOfInit: If TRUE, send M_EI_NA_1 (End of init) after station initialisation is completed. Station initialisation is carried out during system start-up or after a process reset. Default: TRUE.

bSyncTime: If TRUE, enable IEC system time synchronization after reception of C_CS_NA_1 command. Default: TRUE.

bSyncPCTime: If TRUE, synchronise the IEC system time and the system time of the TwinCAT PC (the Windows system time in the taskbar). Default: FALSE.

bUsePCTime: If TRUE, synchronise the IEC system time with the system time of the PC. After the initialisation of the substation, the IEC system time is first synchronised with the system time of the TwinCAT PC. Default: TRUE.

tSyncToPCTime: Controls the cyclic synchronisation of the IEC system itme with the system time of the TwinCAT PC. Implemented in IEC870-5-104 slave v3.0.3 and higher. The cyclic synchronisation is deactivated if tSyncToPCTime = T#0s . Default: T#0s.

sPCTimeNetID: TwinCAT network address of the PC whose system time is to be used for the synchronisation. For the local PC an empty string may be specified. Default: empty stirng = local PC.

bTimeCOT3: If TRUE, send the system time to the central station with the cause of transmission <*3*> *spontaneous* on the hour. Default: FALSE.

tSyncTimeout: Clock time synchronsation intervall timeout monitoring. Implemented in IEC870-5-101/104 slave libraries v2.0.0 and higher. Earlier versions don´t use this parameter. The time stamps have a IV-Quality-Flag (invalid). For value <> T#0m (eg. T#60m) the IV-Quality-Flag is set invalid for all following time stamps if during one hour the Clock time synchronsation has not been executed. The IV-Quality-Flag of the time stamp can be used by the master as indicator for the quality of the time stamp. If tSyncTimeout = T#0m the monitoring is not active . Default: T#0s.

bACTCONRes: If TRUE, send ACTCON response.

bACTTERMRes: If TRUE, send ACTTERM response.

bPerCyclic: If TRUE, enable cyclic/ periodic data transfer

tPerCyclicBase: Base time of cyclic/ periodic data transfer

bBackScan: If TRUE, enable back scan.

tBackScanCycle: Cycle time for back scan.

bPerFRZ: If TRUE, activate local re-storing/resetting of the counter values.

tPerFRZCycle: Cycle time for local re-storing/resetting.

tSelExecTimeout: Max. select/execute time (control station only).

tActConTimeout: Max. activation confirmation time (control station only).

tActTermTimeout: Max. activation termination time (control station only).

tDeactConTimeout: Max. deactivation confirmation time (control station only).

tReadResTimeout: Max. Read response time (control station only).

dbgMode:Debug flags.

Value	Description
IEC870_DEBUGMODE_DISABLED	Log disabled
IEC870_DEBUGMODE_ASDU	The ASDUs are logged as hexadecimal output in the application log.
IEC870_DEBUGMODE_DEVSTATE	Changes in station status are logged in the application log.
IEC870_DEBUGMODE_LINKLAYER	Link layer frames are logged in the application log IEC870-5-101 slave library: all versions IEC870-5-104 slave library: v2.0.0 and higher
IEC870_DEBUGMODE_LINKERROR	Link layer errors are logged in the application log

The flags can be ORed in the desired combination.

orgAddr: Source address (not used).

bOverwrite: Implemented from IEC870-5-104 slave library **v3.0.14** and higher. Activates/deactivates the overwriting of the oldest messages in the send buffer if the max. permissible buffer size is exceeded. This parameter is active only in the offline mode, (i.e. if the connection has been interrupted) and if offline data storage has additionally been activated via the **bRetainBuffer** parameter in the protocol parameters (ST_IEC870_5_104ProtPara). In online mode, i.e. if data transfer has been started, no older messages are overwritten (otherwise some intermediate values would possibly be missing).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.9 ST_IEC870_5_101DeviceInterface

```
TYPE ST_IEC870_5_101DeviceInterface :
STRUCT
    asduAddr    : DWORD;
    asduAddrRev : DWORD;
    dataLink    : ST_IEC870_5_101DataLink;
    clock       : ST_IEC870_5_101SystemRTC;
    comp        : ST_IEC870_5_101DeviceCompatibility;
    errors      : FB_IEC870_5_101ErrorFifo;
    status      : DWORD;
END_STRUCT
END_TYPEEND_TYPE
```

asduAddr: Common ASDU address in standard direction;

asduAddrRev: Common ASDU address in reverse direction;

dataLink: Internal ASDU Send/Receive Fifos;

clock: IEC system time object.

comp: Compatibility list with mapping between ASDU types and transfer causes.

errors: Device error fifo [▶ 271];

status: Global device status. 1=OK, 0=Not Ok;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.10 ST_IEC870_5_101AsduFmtParams

```
TYPE ST_IEC870_5_101AsduFmtParams :
STRUCT
    eCOTSize        : E_IEC870_5_101COTSize        := eIEC870_COT_TwoOctets;
    eAsduAddrSize   : E_IEC870_5_101AsduAddrSize   := eIEC870_AsduAddr_TwoOctets;
    eObjAddrSize    : E_IEC870_5_101ObjAddrSize    := eIEC870_ObjAddr_ThreeOctets;
END_STRUCT
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

Also see about this

- E_IEC870_5_101COTSize [▶ 331]
- E_IEC870_5_101AsduAddrSize [▶ 331]
- E_IEC870_5_101ObjAddrSize [▶ 332]

5.3.11 ST_IEC870_5_101ErrorFifoEntry

IEC60870-5-10x device error

```
TYPE ST_IEC870_5_101ErrorFifoEntry :
STRUCT
    nErrId : UDINT;
    eSrcId : E_IEC870_5_101ErrorSourceID;
END_STRUCT
END_TYPEEND_TYPE
```

nErrID: error code.

eSrcID: error source [▶ 332].The error source gives detailed information about error cause, component or service which reports the error.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.12 ST_IEC870_5_101AcquisitionParams

From product version:

- **TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher;**
- **TwinCAT PLC Library IEC60870-5-101 controlling station v1.0.1 and higher.**

Configuration parameters for cyclic data acquisition. The initialisation sequence commands configured in the *arrInitSeq* array are executed once after receipt of M_EI_NA_1 (end of initialisation) or after the connection is established. The cyclic commands configured via the structure parameters *testCmd*, *clockSync*, *arrGenro, arrCoro* or *genCmd* are then executed.

```
TYPE ST_IEC870_5_101AcquisitionParams :
STRUCT
    (* Initialization sequence steps *)
    arrInitSeq : ARRAY[IEC870_MIN_ISTEP..IEC870_MAX_ISTEP] OF E_IEC870_5_101InitSeqStep :=
    eIEC870_ISTEP_TEST, (* Send test command *)
    eIEC870_ISTEP_CLOCK, (* Send clock synchronization command *)
    eIEC870_ISTEP_GENRO, (* Send general interrogation command *)
    eIEC870_ISTEP_CORO, (* Send counter interrogation command *)
    eIEC870_ISTEP_UNUSED; (* Reserved *)(* Test command polling settings *)
    testCmd : ST_IEC870_5_101TestPollParams := (
    asduAddr := IEC870_ASDUADDR_SYSPARA,
    tPollCycle := T#60s,
    bEnable := TRUE );

    (* Clock synchronisation polling settings *)
    clockSync : ST_IEC870_5_101ClockPollParams := (
    asduAddr := IEC870_ASDUADDR_SYSPARA,
    tPollCycle := T#60s,
    bEnable := TRUE );

    (* Station interrogation polling settings *)
    arrGenro : ARRAY[0..16] OF ST_IEC870_5_101GenroPollParams := (
    asduAddr := IEC870_ASDUADDR_SYSPARA,
    tPollCycle := T#60s,
    eQOI := eIEC870_QOI_INROGEN,
    bEnable := TRUE );

    (* Counter interrogation polling settings *)
    arrCoro : ARRAY[0..4] OF ST_IEC870_5_101CoroPollParams := (
    asduAddr := IEC870_ASDUADDR_SYSPARA,
    tPollCycle := T#60s,
    eRQT := eIEC870_RQT_REQCOGEN,
    eFRZ := eIEC870_FRZ_READ,
    bEnable := TRUE );

  (* General command acquisition settings *)
 genCmd : ST_IEC870_5_101GenCmdPollParams := (
    tPollCycle := T#1h,
    bEnable := FALSE );

  (* Delay command acquisition settings *)
 delayCmd : ST_IEC870_5_101DelayPollParams := (
    tDelay := T#5s );


    eAODBType : E_IEC870_5_101AODBType := eIEC870_AODB_STATIC;
END_STRUCT
END_TYPE
```

arrInitSeq: Initialisation sequence [▶ 336]. The initialisation sequence is always executed once after receipt of M_EI_NA_1 (end of initialisation) or after the connection is established.

testCmd: Parameter for cyclic test commands [▶ 322]. Default: One test command every 60 seconds.

clockSync: Parameter for cyclic time synchronization commands [▶ 323]. Default: Time synchronisation every 60 s.

arrGenro: Parameter for cyclic station query commands [▶ 323]. Station queries for up to 17 datapoint groups may be configured. Default: One station query from the 'General' group every 60 s.

arrCoro: Parameter for cyclic counter query commands [▶ 324]. Counter queries for up to 5 counter groups may be configured. Default: One counter query from the 'General' group every 60 s.

genCmd: Parameter for cyclic transmission of data points in controlling direction [▶ 324] (single-commands, double-commands, setpoint-commands and so forth). Default: Execute all commands every 60 minutes.

delayCmd: Parameter for delay commands [▶ 325]. Execution delay time of the next initialization step command. Default: Delay time = 5 s.

eAODBType: Application database type [▶ 336]. This parameter defines how the data points are stored in the application database.

Example in ST:

In the following program section cyclic data acquisition is configured as follows: All initialisation steps are deactivated. The cyclic test command and time synchronisation command are also deactivated. In addition to the standard station query a further station query from group 1 is configured (every 100 s). In addition to the standard counter query, a counter query (freezing) from counter group 1 is configured (every 200 s).

```
PROGRAM P_AcquisitionConfig
VAR_IN_OUT
    acqPara : ST_IEC870_5_101AcquisitionParams;
END_VAR
```

```
acqPara.arrInitSeq[0]        := eIEC870_ISTEP_UNUSED;
acqPara.arrInitSeq[1]       := eIEC870_ISTEP_UNUSED;
acqPara.arrInitSeq[2]       := eIEC870_ISTEP_UNUSED;
acqPara.arrInitSeq[3]       := eIEC870_ISTEP_UNUSED;
acqPara.arrInitSeq[4]       := eIEC870_ISTEP_UNUSED;

acqPara.testCmd.bEnable         := FALSE;

acqPara.clockSync.bEnable           := FALSE;

acqPara.arrGenro[1].asduAddr    := IEC870_ASDUADDR_SYSPARA;
acqPara.arrGenro[1].eQOI        := eIEC870_QOI_INRO1;
acqPara.arrGenro[1].tPollCycle  := T#100s;
acqPara.arrGenro[1].bEnable     := TRUE;

acqPara.arrCoro[1].asduAddr     := IEC870_ASDUADDR_SYSPARA;
acqPara.arrCoro[1].eFRZ         := eIEC870_FRZ_FREEZE;
acqPara.arrCoro[1].eRQT         := eIEC870_RQT_REQCO1;
acqpara.arrCoro[1].tPollCycle   := T#200s;
acqPara.arrCoro[1].bEnable      := TRUE;
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.13 ST_IEC870_5_101TestPollParams

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

Configuration parameter for the cyclic test command.

```
TYPE ST_IEC870_5_101TestPollParams:
STRUCT
    asduAddr    : DWORD := IEC870_ASDUADDR_SYSPARA;
    tPollCycle  : TIME := T#60s;
    bEnable     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

asduAddr: Target address;

tPollCycle: Cycle time of the test command.

bEnable: Enables (TRUE) or disables (FALSE) the cyclic test commands.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.14 ST_IEC870_5_101ClockPollParams

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

Configuration parameter for the cyclic clock time synchronization command. The station that sends clock time synchronization commands. Has an own internal software clock. This clock is synchronized with the local windows system time (the clock time in the windows task bar) during the station initialization. During operation the target stations are synchronized via the clock time synchronization commands with the tie of the internal software clock.

```
TYPE ST_IEC870_5_101ClockPollParams:
STRUCT
    asduAddr    : DWORD := IEC870_ASDUADDR_SYSPARA;
    tPollCycle  : TIME := T#60s;
    bEnable     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

asduAddr: Target address

tPollCycle: Cycle time of clock time synchronization command.

bEnable: Enables/disables the cyclic clock time synchronization commands.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.15 ST_IEC870_5_101GenroPollParams

From product version: TwinCAT PLC Library: IEC60870-5-104 controlling station v1.0.0 and higher.

Configuration parameter for the cyclic station poll command.

```
TYPE ST_IEC870_5_101GenroPollParams:
STRUCT
    asduAddr    : DWORD := IEC870_ASDUADDR_SYSPARA;
    tPollCycle  : TIME := T#60s;
    eQOI        : E_IEC870_5_101QOI := eIEC870_QOI_INROGEN;
    bEnable     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

asduAddr: Target address

tPollCycle: station poll cycle time.

eQOI: Polling parameter [▶ 338]/characteristic value

bEnable: Enables(TRUE) or disables (FALSE) the station poll commands.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.16 ST_IEC870_5_101CoroPollParams

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

Configuration parameters for the cyclic counter query command.

```
TYPE ST_IEC870_5_101CoroPollParams:
STRUCT
    asduAddr    : DWORD := IEC870_ASDUADDR_SYSPARA;
    tPollCycle  : TIME := T#60s;
    eRQT        : E_IEC870_5_101RQT := eIEC870_RQT_REQCOGEN;
    eFRZ        : E_IEC870_5_101FRZ := eIEC870_FRZ_READ;
    bEnable     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

asduAddr: Target address.

tPollCycle: Cycle time of the counter query command.

eRQT: Counter query ID [▶ 340].

eFRZ: FREEZE/RESET ID [▶ 339].

bEnable : Activates/deactivates the cyclic counter query commands.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.17 ST_IEC870_5_101GenCmdPollParams

General command acquisition settings.

```
TYPE ST_IEC870_5_101GenCmdPollParams:
STRUCT
    tPollCycle : TIME := T#1h; (* General command cycle time *)
    bEnable    : BOOL := FALSE; (* TRUE = Enable cyclic general command, FALSE = Disable *)
    options    : DWORD := 0; (* Additional general command options *)
END_STRUCT
END_TYPE
```

tPollCycle: General command cycle time. Default: 60min.

bEnable: TRUE = Enable cyclic general command, FALSE = Disable.

options: Additional general command options.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.18 ST_IEC870_5_101DelayPollParams

Configuration parameters for the delay of the next initialization step.

```
TYPE ST_IEC870_5_101DelayPollParams:
STRUCT
    tDelay : TIME := T#5s; (* Delay time *)
END_STRUCT
END_TYPE
```

tDelay: Delay time.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.19 ST_IEC870_5_101HashTableKey

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

Application object database lookup key. The key can be used for locating and modifying the hash table entries.

```
TYPE ST_IEC870_5_101HashTableKey :
STRUCT
    eType       : E_IEC870_5_101TcTypeID := ASDU_TYPEUNDEF;
    asduAddr    : DWORD := 0;
    objAddr     : DWORD := 0;
    group       : DWORD := 0;
    lookup      : DWORD := IEC870_LOOKUP_KEY_ALL_ON;
END_STRUCT
END_TYPE
```

eType: Application object type, ASDU identifier [▶ 327] (e.g.: M_SP_NA_1 for single-point or M_DP_NA_1 for double-point etc.).

asduAddr: Common ASDU address.

objAddr : Object address, freely selectable.

group: Object group configuration flags. A description of all group flags can be found here [▶ 348]. The flags can be combined with an OR operation. If this parameter is zero the group parameters are ignored.

lookup: Additional lookup key parameters. The available parameters are declared as constants (see table below). They can be combined with an OR operation.

Constant	Value	Description
IEC870_LOOKUP_KEY_ALL_ON	0	All parameters are considered in the search (*eType, asduAddr, objAddr, group*).
IEC870_LOOKUP_KEY_TYPE_OFF	1	The *eType* parameter is ignored in the search.
IEC870_LOOKUP_KEY_ASDUADDR_OFF	2	The *asduAddr* parameter is ignored in the search.
IEC870_LOOKUP_KEY_GROUP_OFF	4	The *group* parameter is ignored in the search.
IEC870_LOOKUP_KEY_OBJADDR_OFF	8	The *objAddr* parameter is ignored in the search. This parameter is not recommended because all data points must be identifiable via a unique object address.

Example in ST:

See function description: F_iecLookupTableEntry [▶ 300], F_iecRemoveTableEntry [▶ 301], F_iecGetPosOfTableEntry [▶ 298].

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.20 ST_IEC870_5_101FBufferCfg

From product version: TwinCAT PLC Library IEC60870-5-101/104 substation v3.0.2 / IEC60870-5-104 controlling station v1.0.2 and higher.

Configuration settings for offline ASDU data buffer. This structure is used by the functionblock FB_IEC870_5_101FBufferCtrl [▶ 274] .

```
TYPE ST_IEC870_5_101FBufferCfg :
STRUCT
    sNetID      : T_AmsNetID := ''; (* TwinCAT System network address *)
 sPathName       : T_MaxString := 'c:
\Temp\data.dat'; (* File buffer path name (max. length = 255 characters) *)
 ePath       : E_OpenPath := PATH_GENERIC; (* Default: Open generic file *)
 cbBuffer    : UDINT := 16#100000; (* Max. size of file: 16#100000 = 1MB *)
 bOverwrite      : BOOL := TRUE; (* TRUE = overwrite oldest entry, FALSE = don't overwrite *)
 bFilter         : BOOL := FALSE; (* Enable/disable frame filter (reserved)*)
 cotFilter       : T_IEC870_5_101COTBits := 8(0); (* COT (cause of transfer) filter, reserved for fu
ture use *)
 tTimeout    : TIME := DEFAULT_ADS_TIMEOUT; (* ADS (file access) timeout *)
ND_STRUCT
END_TYPE
```

sNetID: It is possible here to provide thenetwork address of the TwinCAT computer on which the buffer file is to be created The string can also be empty for the local computer.

sPathName: Contains the path and filename for the file to be opened. The path can only point to the local computer's file system. This means that network paths cannot be used here.

ePath: This input can be used to select a TwinCAT system path on the target device for opening the file.

cbBuffer: Maximum byte size of buffer file. If bOverwrite = FALSE is set and the maximum size was exceeded an error is returned.

bOverwrite: If variable is set TRUE: the oldest entries are overwritten by reaching the maximum size.

bFilter: Not yet implemented. Enables/disables a COT filter (Cause of transfer). Only ASDUs with specific cause of transfer are buffered in the file.

cotFilter: Not yet implemented. Via this variable COTS [▶ 345] (cause of transfer) can be configured, that have tio be buffered in the file.

tTimeout: States the length of the timeout that may not be exceeded by file access.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.21 ST_IEC870_5_101FBufferStatus

From product version: TwinCAT PLC Library IEC60870-5-101/104 substation v3.0.2 / IEC60870-5-104 controllingstation v1.0.2 and higher.

Status information to offline ASDU data buffer. This structure is used by the functionblock FB_IEC870_5_101FBufferCtrl [▶ 274] .

```
TYPE ST_IEC870_5_101FBufferStatus:
STRUCT
    eState      : E_IEC870_5_101FBufferState := eIEC870_FBUFFER_IDLE; (* File buffer data direction
status (storing, loading, idle, error) *)
    nErrID      : UDINT; (* File access error code *)
    bCorrupted  : BOOL; (* TRUE => Existing file was corrupted and with new (empty) file replaced.
    FALSE => Existing file was not corrupted or new (empty) file created. *)
    nCount      : UDINT; (* Number of buffered entries in file *)
END_STRUCT
END_TYPE
```

eState: Supplies the file buffer data direction status [▶ 337]. (File is written, loaded, closed or an error occurs while writing/reading).

nErrID: Supplies the file access ADS error code..

bCorrupted: TRUE: the exisiting file was corrupted and is replaced with a new empty one. A corrupted file occurs if the maximum data buffer size was changed, or the file was not closed accurately.

nCount: Number of current buffered entries in the data buffer. The data buffer must be opened to detect the number of entries in the existing data buffer. That is the connection has to be in the offline mode for a short moment.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.22 E_IEC870_5_101TcTypeID

Data point type identifier.

```
TYPE E_IEC870_5_101TcTypeID :
(
    ASDU_TYPEUNDEF := 0, (* (000, 0x00) not allowed *)
(* reserved standard asdu types *)
    M_SP_NA_1, (* (001, 0x01) Single-point information *)
    M_SP_TA_1, (* (002, 0x02) Single-point information with time tag *)
    M_DP_NA_1, (* (003, 0x03) Double-point information *)
    M_DP_TA_1, (* (004, 0x04) Double-point information with time tag *)
    M_ST_NA_1, (* (005, 0x05) Step position information *)
    M_ST_TA_1, (* (006, 0x06) Step position information with time tag *)
    M_BO_NA_1, (* (007, 0x07) Bitstring of 32 bit *)
    M_BO_TA_1, (* (008, 0x08) Bitstring of 32 bit with time tag *)
    M_ME_NA_1, (* (009, 0x09) Measured value, normalised value *)
    M_ME_TA_1, (* (010, 0x0A) Measured value, normalized value with time tag *)
    M_ME_NB_1, (* (011, 0x0B) Measured value, scaled value *)
    M_ME_TB_1, (* (012, 0x0C) Measured value, scaled value wit time tag *)
    M_ME_NC_1, (* (013, 0x0D) Measured value, short floating point number *)
    M_ME_TC_1, (* (014, 0x0E) Measured value, short floating point number with time tag *)
    M_IT_NA_1, (* (015, 0x0F) Integrated totals *)
    M_IT_TA_1, (* (016, 0x10) Integrated totals with time tag *)
    M_EP_TA_1, (* (017, 0x11) Event of protection equipment with time tag *)
    M_EP_TB_1, (* (018, 0x12) Packed start events of protection equipment with time tag *)
    M_EP_TC_1, (* (019, 0x13) Packed output circuit information of protection equipment with time ta
g *)
    M_PS_NA_1, (* (020, 0x14) Packed single point information with status change detection *)
    M_ME_ND_1, (* (021, 0x15) Measured value, normalized value without quality descriptor *)
    ASDU_TYPE_22,
    ASDU_TYPE_23,
```

```
    ASDU_TYPE_24,
    ASDU_TYPE_25,
    ASDU_TYPE_26,
    ASDU_TYPE_27,
    ASDU_TYPE_28,
    ASDU_TYPE_29,
    M_SP_TB_1, (* (030, 0x1E) Single-point information with time tag CP56Time2a *)
    M_DP_TB_1, (* (031, 0x1F) Double-point information with time tag CP56Time2a *)
    M_ST_TB_1, (* (032, 0x20) Step position information with time tag CP56Time2a *)
    M_BO_TB_1, (* (033, 0x21) Bitstring of 32 bit with time tag CP56Time2a *)
    M_ME_TD_1, (* (034, 0x22) Measured value, normalised value with time tag CP56Time2a *)
    M_ME_TE_1, (* (035, 0x23) Measured value, scaled value with time tag CP56Time2a *)
    M_ME_TF_1, (* (036, 0x24) Measured value, short floating point number with time tag CP56Time2a *
)
    M_IT_TB_1, (* (037, 0x25) Integrated totals with time tag CP56Time2a *)
    M_EP_TD_1, (* (038, 0x26) Event of protection equipment with time tag CP56Time2a *)
    M_EP_TE_1, (* (039, 0x27) Packed start events of protection equipment with time tag CP56Time2a *
)
    M_EP_TF_1, (* (040, 0x28) Packed output circuit information of protection equipment with time ta
g CP56Time2a *)
    ASDU_TYPE_41,
    ASDU_TYPE_42,
    ASDU_TYPE_43,
    ASDU_TYPE_44,
    C_SC_NA_1, (* (045, 0x2D) Single command *)
    C_DC_NA_1, (* (046, 0x2E) Double command *)
    C_RC_NA_1, (* (047, 0x2F) Regulating step command *)
    C_SE_NA_1, (* (048, 0x30) Set-point Command, normalised value *)
    C_SE_NB_1, (* (049, 0x31) Set-point Command, scaled value *)
    C_SE_NC_1, (* (050, 0x32) Set-point Command, short floating point number *)
    C_BO_NA_1, (* (051, 0x33) Bitstring 32 bit command *)
    ASDU_TYPE_52,
    ASDU_TYPE_53,
    ASDU_TYPE_54,
    ASDU_TYPE_55,
    ASDU_TYPE_56,
    ASDU_TYPE_57,
    C_SC_TA_1, (* (058, 0x3A) Single command with time tag CP56Time2a *)
    C_DC_TA_1, (* (059, 0x3B) Double command with time tag CP56Time2a *)
    C_RC_TA_1, (* (060, 0x3C) Regulating step command with time tag CP56Time2a *)
    C_SE_TA_1, (* (061, 0x3D) Measured value, normalised value command with time tag CP56Time2a *)
    C_SE_TB_1, (* (062, 0x3E) Measured value, scaled value command with time tag CP56Time2a *)
    C_SE_TC_1, (* (063, 0x3F) Measured value, short floating point number command with time tag CP56
Time2a *)
    C_BO_TA_1, (* (064, 0x40) Bitstring of 32 bit command with time tag CP56Time2a *)
    ASDU_TYPE_65,
    ASDU_TYPE_66,
    ASDU_TYPE_67,
    ASDU_TYPE_68,
    ASDU_TYPE_69,
    M_EI_NA_1, (* (070, 0x46) End of Initialisation *)
    ASDU_TYPE_71,
    ASDU_TYPE_72,
    ASDU_TYPE_73,
    ASDU_TYPE_74,
    ASDU_TYPE_75,
    ASDU_TYPE_76,
    ASDU_TYPE_77,
    ASDU_TYPE_78,
    ASDU_TYPE_79,
    ASDU_TYPE_80,
    ASDU_TYPE_81,
    ASDU_TYPE_82,
    ASDU_TYPE_83,
    ASDU_TYPE_84,
    ASDU_TYPE_85,
    ASDU_TYPE_86,
    ASDU_TYPE_87,
    ASDU_TYPE_88,
    ASDU_TYPE_89,
    ASDU_TYPE_90,
    ASDU_TYPE_91,
    ASDU_TYPE_92,
    ASDU_TYPE_93,
    ASDU_TYPE_94,
    ASDU_TYPE_95,
    ASDU_TYPE_96,
    ASDU_TYPE_97,
    ASDU_TYPE_98,
    ASDU_TYPE_99,
```

```
    C_IC_NA_1, (* (100, 0x64) Interrogation command *)
    C_CI_NA_1, (* (101, 0x65) Counter interrogation command *)
    C_RD_NA_1, (* (102, 0x66) Read Command*)
    C_CS_NA_1, (* (103, 0x67) Clock synchronization command *)
    C_TS_NA_1, (* (104, 0x68) Test command *)
    C_RP_NA_1, (* (105, 0x69) Reset process command *)
    C_CD_NA_1, (* (106, 0x6A) C_CD_NA_1 Delay acquisition command *)
    C_TS_TA_1, (* (107, 0x6B) Test command with time tag CP56Time2a *)
ASDU_TYPE_108,
    ASDU_TYPE_109,
    P_ME_NA_1, (* (110, 0x6E) Parameter of measured values, normalized value *)
P_ME_NB_1, (* (111, 0x6F) Parameter of measured values, scaled value *)
    P_ME_NC_1, (* (112, 0x70) Parameter of measured values, short floating point number *)
    P_AC_NA_1, (* (113, 0x71) Parameter activation *)
ASDU_TYPE_114,
    ASDU_TYPE_115,
    ASDU_TYPE_116,
    ASDU_TYPE_117,
    ASDU_TYPE_118,
    ASDU_TYPE_119,
    F_FR_NA_1, (* (120, 0x78) File ready *)
    F_SR_NA_1, (* (121, 0x79) Section ready *)
    F_SC_NA_1, (* (122, 0x7A) Call directory, select file, call file, call section *)
    F_LS_NA_1, (* (123, 0x7B) Last section, last segment *)
    F_FA_NA_1, (* (124, 0x7C) ACK file, ACK section *)
    F_SG_NA_1, (* (125, 0x7D) Segment *)
    F_DR_TA_1, (* (126, 0x7E) Directory *)
    ASDU_TYPE_127,
    (* reserved user asdu types *)
    ASDU_TYPE_128,
    ASDU_TYPE_129,
    ASDU_TYPE_130,
    ASDU_TYPE_131,
    ASDU_TYPE_132,
    ASDU_TYPE_133,
    ASDU_TYPE_134,
    ASDU_TYPE_135,
    ASDU_TYPE_136,
    ASDU_TYPE_137,
    ASDU_TYPE_138,
    ASDU_TYPE_139,
    ASDU_TYPE_140,
    ASDU_TYPE_141,
    ASDU_TYPE_142,
    ASDU_TYPE_143,
    ASDU_TYPE_144,
    ASDU_TYPE_145,
    ASDU_TYPE_146,
    ASDU_TYPE_147,
    ASDU_TYPE_148,
    ASDU_TYPE_149,
    ASDU_TYPE_150,
    ASDU_TYPE_151,
    ASDU_TYPE_152,
    ASDU_TYPE_153,
    ASDU_TYPE_154,
    ASDU_TYPE_155,
    ASDU_TYPE_156,
    ASDU_TYPE_157,
    ASDU_TYPE_158,
    ASDU_TYPE_159,
    ASDU_TYPE_160,
    ASDU_TYPE_161,
    ASDU_TYPE_162,
    ASDU_TYPE_163,
    ASDU_TYPE_164,
    ASDU_TYPE_165,
    ASDU_TYPE_166,
    ASDU_TYPE_167,
    ASDU_TYPE_168,
    ASDU_TYPE_169,
    ASDU_TYPE_170,
    ASDU_TYPE_171,
    ASDU_TYPE_172,
    ASDU_TYPE_173,
    ASDU_TYPE_174,
    ASDU_TYPE_175,
    ASDU_TYPE_176,
    ASDU_TYPE_177,
    ASDU_TYPE_178,
```

```
    ASDU_TYPE_179,
    ASDU_TYPE_180,
    ASDU_TYPE_181,
    ASDU_TYPE_182,
    ASDU_TYPE_183,
    ASDU_TYPE_184,
    ASDU_TYPE_185,
    ASDU_TYPE_186,
    ASDU_TYPE_187,
    ASDU_TYPE_188,
    ASDU_TYPE_189,
    ASDU_TYPE_190,
    ASDU_TYPE_191,
    ASDU_TYPE_192,
    ASDU_TYPE_193,
    ASDU_TYPE_194,
    ASDU_TYPE_195,
    ASDU_TYPE_196,
    ASDU_TYPE_197,
    ASDU_TYPE_198,
    ASDU_TYPE_199,
    ASDU_TYPE_200,
    ASDU_TYPE_201,
    ASDU_TYPE_202,
    ASDU_TYPE_203,
    ASDU_TYPE_204,
    ASDU_TYPE_205,
    ASDU_TYPE_206,
    ASDU_TYPE_207,
    ASDU_TYPE_208,
    ASDU_TYPE_209,
    ASDU_TYPE_210,
    ASDU_TYPE_211,
    ASDU_TYPE_212,
    ASDU_TYPE_213,
    ASDU_TYPE_214,
    ASDU_TYPE_215,
    ASDU_TYPE_216,
    ASDU_TYPE_217,
    ASDU_TYPE_218,
    ASDU_TYPE_219,
    ASDU_TYPE_220,
    ASDU_TYPE_221,
    ASDU_TYPE_222,
    ASDU_TYPE_223,
    ASDU_TYPE_224,
    ASDU_TYPE_225,
    ASDU_TYPE_226,
    ASDU_TYPE_227,
    ASDU_TYPE_228,
    ASDU_TYPE_229,
    ASDU_TYPE_230,
    ASDU_TYPE_231,
    ASDU_TYPE_232,
    ASDU_TYPE_233,
    ASDU_TYPE_234,
    ASDU_TYPE_235,
    ASDU_TYPE_236,
    ASDU_TYPE_237,
    ASDU_TYPE_238,
    ASDU_TYPE_239,
    ASDU_TYPE_240,
    ASDU_TYPE_241,
    ASDU_TYPE_242,
    ASDU_TYPE_243,
    ASDU_TYPE_244,
    ASDU_TYPE_245,
    ASDU_TYPE_246,
    ASDU_TYPE_247,
    ASDU_TYPE_248,
    ASDU_TYPE_249,
    ASDU_TYPE_250,
    ASDU_TYPE_251,
    ASDU_TYPE_252,
    ASDU_TYPE_253,
    ASDU_TYPE_254,
    ASDU_TYPE_255,
    ASDU_TYPEMAX (* not used *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.23 E_IEC870_5_101IOMappingType

```
TYPE E_IEC870_5_101IOMappingType :
(
    MAP_AREA_NONE   := 0,
    MAP_AREA_MEMORY     := 1,
    MAP_AREA_INPUT  := 2,
    MAP_AREA_OUTPUT     := 4,
    MAP_AREA_DATA   := 8
);
END_TYPE
```

TwinCAT PLC process data range (inputs, outputs, memory, data) into resp. from the IEC process data are to be mapped (copied).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.24 E_IEC870_5_101AsduAddrSize

```
TYPE E_IEC870_5_101AsduAddrSize :
(
    eIEC870_AsduAddr_OneOctet           := 1, (* One octet *)
    eIEC870_AsduAddr_TwoOctets      := 2 (* Two octets *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.25 E_IEC870_5_101COTSize

```
TYPE E_IEC870_5_101COTSize :
(
    eIEC870_COT_OneOctet    := 1, (* One octet *)
    eIEC870_COT_TwoOctets   := 2 (* Two octets *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.26 E_IEC870_5_101LinkAddrSize

```
TYPE E_IEC870_5_101LinkAddrSize :
(
    eIEC870_LinkAddr_None       := 0, (* None *)
    eIEC870_LinkAddr_OneOctet   := 1, (* One octet *)
    eIEC870_LinkAddr_TwoOctets  := 2 (* Two octets *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.27 E_IEC870_5_101ObjAddrSize

```
TYPE E_IEC870_5_101ObjAddrSize :
(
    eIEC870_ObjAddr_OneOctet        := 1, (* One octet *)
    eIEC870_ObjAddr_TwoOctets       := 2, (* Two octets *)
    eIEC870_ObjAddr_ThreeOctets     := 3 (* Three octets *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.28 E_IEC870_5_101ErrorSourceID

Error source.

```
TYPE E_IEC870_5_101ErrorSourceID :
(
    eIEC870_ESRC_NONE,              (* Error source is unknown *)
    eIEC870_ESRC_COUNTER_INTERROGATION,     (* Error during counter interrogation command execution
*)
    eIEC870_ESRC_SYNC_CLOCK_CTRL,   (* Error during execution of clock synchronisation command *)
    eIEC870_ESRC_CLOCK_EVENT,           (* Error during spontanous transmission of clock time *)
    eIEC870_ESRC_GETPCTIME,     (* Error during synchronisation of device time to PC time *)
    eIEC870_ESRC_SETPCTIME,     (* Error during synchronisation of PC time to device time *)
    eIEC870_ESRC_STATION_INTERROGATION,     (* Error during execution of general interrogation *)
    eIEC870_ESRC_READ_DATA_CMD,         (* Error during execution of read command *)
    eIEC870_ESRC_RESET_PROCESS,         (* Error during execution of process reset command *)
```

```
    eIEC870_ESRC_TEST_CMD,      (* Error during execution of test command *)
    eIEC870_ESRC_ENDOFINIT,     (* Error during transmission of M_EI_NA_1 (end of initialisation) *)
    eIEC870_ESRC_BACKGROUND_SCAN,   (* Error during execution of background scan *)
    eIEC870_ESRC_COMMAND_CTRL,          (* Error during initialisation of command (single-
command, double-command, setpoint-command and so forth ) *)
    eIEC870_ESRC_COMMAND_EXEC,          (* Error during command execution *)
    eIEC870_ESRC_LOCAL_FREEZE_RESET,    (* Error during counter freeze/reset *)
    eIEC870_ESRC_PERIODIC_CYCLIC,   (* Error during cyclic/periodic data transmission *)
    eIEC870_ESRC_USERAPP_OBJECT,    (* Error in application object (invalid configuration/value) *)
    eIEC870_ESRC_USERAPP_SETQUALITY,    (* Error in quality flag execution *)
    eIEC870_ESRC_IEC60870_5_104LINK,    (* Error in IEC60870-5-104 link layer *)
    eIEC870_ESRC_IEC60870_5_101LINK,    (* Error in IEC60870-5-101 link layer *)
    eIEC870_ESRC_COMLIB,        (* Error in the basic serial communication interface *)
    eIEC870_ESRC_POLLING_SEVICE     (* Error in the acquisition state machine (execution of cyclic p
oll commands) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.29 E_IEC870_5_101ClassType

```
TYPE E_IEC870_5_101ClassType :
(
    eIEC870_Class_None,
    eIEC870_Class_1,
    eIEC870_Class_2
);
END_TYPE
```

Priority class of ASDU. High prior data are assigned to class 1, low prior data are assigned to class 2.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.30 E_IEC870_5_101COTType

Cause of transfer

```
TYPE E_IEC870_5_101COTType:
(
    eIEC870_COT_UNUSED := 0,
    eIEC870_COT_CYCLIC := 1,
    eIEC870_COT_BACKGROUND := 2,
    eIEC870_COT_SPONTAN := 3,
    eIEC870_COT_INIT := 4,
    eIEC870_COT_REQ := 5,
    eIEC870_COT_ACT := 6,
    eIEC870_COT_ACT_CON := 7,
    eIEC870_COT_DEACT := 8,
    eIEC870_COT_DEACT_CON := 9,
    eIEC870_COT_ACT_TERM := 10,
    eIEC870_COT_RETREM := 11,
    eIEC870_COT_RETLOC := 12,
    eIEC870_COT_FILE := 13,
    eIEC870_COT_14 := 14,
    eIEC870_COT_15 := 15,
```

```
    eIEC870_COT_16 := 16,
    eIEC870_COT_17 := 17,
    eIEC870_COT_18 := 18,
    eIEC870_COT_19 := 19,
    eIEC870_COT_INROGEN := 20,
    eIEC870_COT_INRO1 := 21,
    eIEC870_COT_INRO2 := 22,
    eIEC870_COT_INRO3 := 23,
    eIEC870_COT_INRO4 := 24,
    eIEC870_COT_INRO5 := 25,
    eIEC870_COT_INRO6 := 26,
    eIEC870_COT_INRO7 := 27,
    eIEC870_COT_INRO8 := 28,
    eIEC870_COT_INRO9 := 29,
    eIEC870_COT_INRO10 := 30,
    eIEC870_COT_INRO11 := 31,
    eIEC870_COT_INRO12 := 32,
    eIEC870_COT_INRO13 := 33,
    eIEC870_COT_INRO14 := 34,
    eIEC870_COT_INRO15 := 35,
    eIEC870_COT_INRO16 := 36,
    eIEC870_COT_REQCOGEN := 37,
    eIEC870_COT_REQCO1 := 38,
    eIEC870_COT_REQCO2 := 39,
    eIEC870_COT_REQCO3 := 40,
    eIEC870_COT_REQCO4 := 41,
    eIEC870_COT_42 := 42,
    eIEC870_COT_43 := 43,
    eIEC870_COT_UNKNOWN_TYPE := 44,
    eIEC870_COT_UNKNOWN_CAUSE := 45,
    eIEC870_COT_UNKNOWN_ASDU_ADDRESS := 46,
    eIEC870_COT_UNKNOWN_OBJECT_ADDRESS := 47,
    eIEC870_COT_48 := 48,
    eIEC870_COT_49 := 49,
    eIEC870_COT_50 := 50,
    eIEC870_COT_51 := 51,
    eIEC870_COT_52 := 52,
    eIEC870_COT_53 := 53,
    eIEC870_COT_54 := 54,
    eIEC870_COT_55 := 55,
    eIEC870_COT_56 := 56,
    eIEC870_COT_57 := 57,
    eIEC870_COT_58 := 58,
    eIEC870_COT_59 := 59,
    eIEC870_COT_60 := 60,
    eIEC870_COT_61 := 61,
    eIEC870_COT_62 := 62,
    eIEC870_COT_63 := 63
);
END_TYPE
```

Value	Description
eIEC870_COT_UNUSED	Not used
eIEC870_COT_CYCLIC	Cyclic data
eIEC870_COT_BACKGROUND	Background request
eIEC870_COT_SPONTAN	Spontaneous data
eIEC870_COT_INIT	End of initialisation
eIEC870_COT_REQ	Read-Request
eIEC870_COT_ACT	Command activation
eIEC870_COT_ACT_CON	Acknowledgement of command activation
eIEC870_COT_DEACT	Command abort
eIEC870_COT_DEACT_CON	Acknowledgement of command abort
eIEC870_COT_ACT_TERM	Termination of command activation
eIEC870_COT_RETREM	Return because of remote command
eIEC870_COT_RETLOC	Return because local command
eIEC870_COT_FILE	File access
eIEC870_COT_INROGEN	Station interrogation (general)
eIEC870_COT_INRO1	Station interrogation of group 1
eIEC870_COT_INRO2	Station interrogation of group 2

Value	Description
eIEC870_COT_INRO3	Station interrogation of group 3
eIEC870_COT_INRO4	Station interrogation of group 4
eIEC870_COT_INRO5	Station interrogation of group 5
eIEC870_COT_INRO6	Station interrogation of group 6
eIEC870_COT_INRO7	Station interrogation of group 7
eIEC870_COT_INRO8	Station interrogation of group 8
eIEC870_COT_INRO9	Station interrogation of group 9
eIEC870_COT_INRO10	Station interrogation of group 10
eIEC870_COT_INRO11	Station interrogation of group 11
eIEC870_COT_INRO12	Station interrogation of group 12
eIEC870_COT_INRO13	Station interrogation of group 13
eIEC870_COT_INRO14	Station interrogation of group 14
eIEC870_COT_INRO15	Station interrogation of group 15
eIEC870_COT_INRO16	Station interrogation of group 16
eIEC870_COT_REQCOGEN	Counter request (general)
eIEC870_COT_REQCO1	Counter request of group 1
eIEC870_COT_REQCO2	Counter request of group 2
eIEC870_COT_REQCO3	Counter request of group 3
eIEC870_COT_REQCO4	Counter request of group 4
eIEC870_COT_UNKNOWN_TYPE	Unknown type
eIEC870_COT_UNKNOWN_CAUSE	Unknown transmission cause
eIEC870_COT_UNKNOWN_ASDU_ADDRESS	Unknown collective ASDU address
eIEC870_COT_UNKNOWN_OBJECT_ADDRESS	Unknown object address
eIEC870_COT_14..eIEC870_COT_19 eIEC870_COT_42..eIEC870_COT_43 eIEC870_COT_48..eIEC870_COT_63	Reserved/not used area

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.31 E_IEC870_5_101FifoDbgFlags

Debug output flags. If the debug output is activated, the ASDU hexbytes are logged in the TwinCAT System Manager logger window.

```
TYPE E_IEC870_5_101FifoDbgFlags:
(
    eIEC870_FIFO_DBG_OFF := 0,
    eIEC870_FIFO_DBG_PUT := 1,
    eIEC870_FIFO_DBG_GET := 2,
    eIEC870_FIFO_DBG_ALL := 3
);
END_TYPE
```

Please regard that the debug output creates a higher system load.

Value	Description
eIEC870_FIFO_DBG_OFF	No debug output
eIEC870_FIFO_DBG_PUT	Debug output when adding the Fifo elements
eIEC870_FIFO_DBG_GET	Debug otuput when removing the Fifo elements
eIEC870_FIFO_DBG_ALL	Debug output when adding und removing the Fifo elements

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.32 E_IEC870_5_101AODBType

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

Type of application data base. At the moment only the type eIEC870_AODB_STATIC is supported. That means all application objects (data points) have to be configured by the PLC application.

```
TYPE E_IEC870_5_101AODBType :
(
    eIEC870_AODB_STATIC := 0,
    eIEC870_AODB_DYNAMIC := 1
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.33 E_IEC870_5_101InitSeqStep

Commands to be executed once (initialisation sequence) after receipt of M_EI_NA_1 (end of initialisation) or after the communication establishment with the slave device.

```
TYPE E_IEC870_5_101InitSeqStep :
(
    eIEC870_ISTEP_UNUSED := 0,      (* Do nothing *)
    eIEC870_ISTEP_CLOCK,    (* Send clock synchronization command *)
    eIEC870_ISTEP_TEST,     (* Send test command *)
    eIEC870_ISTEP_GENRO,    (* Send general interrogation command *)
    eIEC870_ISTEP_CORO,     (* Send counter interrogation command *)
    eIEC870_ISTEP_COMMAND,  (* Send general command *)
    eIEC870_ISTEP_DELAY     (* Delay timer *)
);
END_TYPE
```

Initialisation step	Description
eIEC870_ISTEP_UNUSED	The initialisation step is not used.
eIEC870_ISTEP_CLOCK	The server is to send a clock synchronization command.

Initialisation step	Description
eIEC870_ISTEP_TEST	The server is to send a test command.
eIEC870_ISTEP_GENRO	The server is to send a general interrogation command.
eIEC870_ISTEP_CORO	The server ist to send a general conunter interrogation command.
eIEC870_ISTEP_COMMAND	The server is to send commands once (all data points in controlling direction: single-points, double-points, set-point commands and so forth).
eIEC870_ISTEP_DELAY	The server is to wait the specified number of time prior execution of the next initialization step command

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.34 E_IEC870_5_101FBufferState

From product version: TwinCAT PLC Library IEC60870-5-101/104 substation v3.0.2 / IEC60870-5-104 controlling station v1.0.2 and higher.

Current offline ASDU file buffer state.

```
TYPE E_IEC870_5_101QU :
(
    eIEC870_FBUFFER_IDLE := 0,
    eIEC870_FBUFFER_SAVING := 1,
    eIEC870_FBUFFER_LOADING := 2,
    eIEC870_FBUFFER_ERROR := 3
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.35 E_IEC870_5_101SCS

Single command state.

```
TYPE E_IEC870_5_101SCS :
(
    eIEC870_SCS_OFF     := 0,
    eIEC870_SCS_ON  := 1
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.36 E_IEC870_5_101DCS

Double command state.

```
TYPE E_IEC870_5_101DCS :
(
    eIEC870_DCS_INDETERMINATE0  := 0,
    eIEC870_DCS_OFF             := 1,
    eIEC870_DCS_ON              := 2,
    eIEC870_DCS_INDETERMINATE3  := 3
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.37 E_IEC870_5_101COI

Cause of initialization.

```
TYPE E_IEC870_5_101COI :
(
    eIEC870_COI_LOCAL_POWER_ON      := 0, (* Local power ON *)
    eIEC870_COI_LOCAL_MANUAL_RESET  := 1, (* Local manual reset *)
    eIEC870_COI_REMOTE_RESET        := 2 (* Remote reset *)
(* <3..31> := reserved for future norm definitions
    <32..127> := reserved for user definitions (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.38 E_IEC870_5_101QOI

Qualifier of interrogation command.

```
TYPE E_IEC870_5_101QOI :
(
    eIEC870_QOI_UNUSED      := 0, (* not used*)
(* <1..19> := reserved for standard definitions of this companion standard *)
    eIEC870_QOI_INROGEN     := 20, (* global station interrogation *)
    eIEC870_QOI_INRO1       := 21, (* group 1 station interrogation *)
    eIEC870_QOI_INRO2       := 22, (* group 2 station interrogation...*)
    eIEC870_QOI_INRO3       := 23,
    eIEC870_QOI_INRO4       := 24,
```

```
    eIEC870_QOI_INRO5       := 25,
    eIEC870_QOI_INRO6       := 26,
    eIEC870_QOI_INRO7       := 27,
    eIEC870_QOI_INRO8       := 28,
    eIEC870_QOI_INRO9       := 29,
    eIEC870_QOI_INRO10      := 30,
    eIEC870_QOI_INRO11      := 31,
    eIEC870_QOI_INRO12      := 32,
    eIEC870_QOI_INRO13      := 33,
    eIEC870_QOI_INRO14      := 34,
    eIEC870_QOI_INRO15      := 35,
    eIEC870_QOI_INRO16      := 36 (* group 16 station interrogation*)
(* <37..63> := reserved for future norm definitions of this companion standard (compatible range) *)
    (* <64..255> := reserved for user definitions (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.39 E_IEC870_5_101QL

Qualifier of set-point command.

```
TYPE E_IEC870_5_101QL :
(
    eIEC870_QL_DEFAULT := 0
  (* <1..63> := reserved for standard definitions (compatible range)
    <64..127> := reserved for special use (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.40 E_IEC870_5_101FRZ

Freeze/reset qualifier of counter interrogation command.

```
TYPE E_IEC870_5_101FRZ :
(
    eIEC870_FRZ_READ        := 0, (* Read only (no freeze or reset) *)
    eIEC870_FRZ_FREEZE      := 1, (* Counter freeze without reset (value frozen represents integrate
d total) *)
    eIEC870_FRZ_FREEZE_AND_RESET := 2, (* Counter freeze with reset (value frozen represents increme
ntal information) *)
    eIEC870_FRZ_RESET       := 3 (* Counter reset *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.41 E_IEC870_5_101RQT

Request qualifier of counter interrogation command.

```
TYPE E_IEC870_5_101RQT :
(
    eIEC870_RQT_NONE        := 0, (* No counter read *)
 eIEC870_RQT_REQCO1      := 1, (* Group 1 counter interrogation *)
    eIEC870_RQT_REQCO2      := 2, (* Group 2 counter interrogation *)
    eIEC870_RQT_REQCO3      := 3, (* Group 3 counter interrogation *)
    eIEC870_RQT_REQCO4      := 4, (* Group 4 counter interrogation *)
    eIEC870_RQT_REQCOGEN    := 5 (* General counter interrogation *)  (* <6..31> reserved for future
 norm definitions
    <32..63> reserved for user definitions (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.42 E_IEC870_5_101QRP

Qualifier of reset process command.

```
TYPE E_IEC870_5_101QRP :
(
    eIEC870_QRP_UNUSED      := 0, (* not used *)
    eIEC870_QRP_GENERAL     := 1, (* general process reset *)
    eIEC870_QRP_TTEVENTS    := 2 (* reset pending events with time tag *)  (* <3..127> := reserved f
or future norm definitions
    <128..255> := reserved for user definitions (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.43 E_IEC870_5_101QU

Qualifier of command.

```
TYPE E_IEC870_5_101QU :
(
    eIEC870_QU_UNSPECIFIED  := 0,
    eIEC870_QU_SHORTPULSE   := 1,
    eIEC870_QU_LONGPULSE    := 2,
    eIEC870_QU_PERSISTENT   := 3
  (* <4..8> := reserved for standrad definitions of companion standard (compatible range)
    <9..15> := reserved for the selection of other predefined functions
    <16..31> := reserved for special use (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.44 E_IEC870_5_101ES

Event state (single event of protection equipment).

```
TYPE E_IEC870_5_101ES:
(
    eIEC870_ES_INDETERMINATE0       := 0,
    eIEC870_ES_OFF      := 1,
    eIEC870_ES_ON       := 2,
    eIEC870_ES_INDETERMINATE3       := 3
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.45 E_IEC870_5_101KPA

Kind of parameter.

```
TYPE E_IEC870_5_101KPA:
(
    eIEC870_KPA_UNUSED := 0,
    eIEC870_KPA_THRESH := 1,
    eIEC870_KPA_FILTER := 2,
    eIEC870_KPA_LOLIMIT := 3,
    eIEC870_KPA_HILIMIT := 4
  (* <5..31> := reserved for standard definitions (compatible range)
    <32..63> := reserved for special use (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.46 E_IEC870_5_101QPA

Qualifier of parameter activation.

```
TYPE E_IEC870_5_101QPA:
(
    eIEC870_QPA_UNUSED := 0,
    eIEC870_QPA_GENERAL := 1,
    eIEC870_QPA_OBJECT := 2,
    eIEC870_QPA_TRANSMISSION := 3
  (* <4..127> := reserved for standard definitions (compatible range)
    <128..255> := reserved for special use (private range) *)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.47 E_IEC870_5_101RCS

Regulating step command state.

```
TYPE E_IEC870_5_101RCS :
(
    eIEC870_RCS_NOTALLOWED0     := 0,
    eIEC870_RCS_DECREMENT   := 1,
    eIEC870_RCS_INCREMENT   := 2,
    eIEC870_RCS_NOTALLOWED3     := 3
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.48 E_IEC870_5_101SPI

Single-point information.

```
TYPE E_IEC870_5_101SPI :
(
    eIEC870_SPI_OFF     := 0,
    eIEC870_SPI_ON  := 1
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.49 E_IEC870_5_101DPI

Double-point information.

```
TYPE E_IEC870_5_101DPI :
(
    eIEC870_DPI_INDETERMINATE0  := 0,
    eIEC870_DPI_OFF             := 1,
    eIEC870_DPI_ON              := 2,
    eIEC870_DPI_INDETERMINATE3  := 3
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.50 T_HAODBTable

From product version: TwinCAT PLC Library IEC60870-5-104 controlling station v1.0.0 and higher.

Application object data base handle (hash table handle). The table handle has to be initialized once before using with the function F_iecCreateTableHnd.

```
TYPE T_HAODBTable :
STRUCT
(*...*)
END_STRUCT
END_TYPE
```

The access to structured variables is not direct, but indirect with the aid of the available functions and function blocks. The following functions can be used:

Function	Description
F_iecCreateTableHnd [▶ 295]	Initializes the hash table handle
F_iecAddTableEntry [▶ 296]	Configures and adds a new hash table entry
F_iecRemoveTableEntry [▶ 301]	Removes a hash table entry
F_iecLookupTableEntry [▶ 300]	Checks if a special hash table entry exists
F_iecGetPosOfTableEntry [▶ 298]	Determines the linear position of a hash table entry

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.51 T_CP16Time2a

Two octets binary time format.

```
TYPE T_CP16Time2a :
STRUCT
    Milliseconds    : WORD; (* 0..59.999ms = 60sec = 1min *)
END_STRUCT
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.52 T_CP24Time2a

Three octets binary time format.

```
TYPE T_CP24Time2a :
STRUCT
    Milliseconds    : WORD; (* 0..59.999ms = 60sec = 1min *)
    IVResMinute     : BYTE; (* Bit 7 = IV (invalid time), Bit 6 = Res (spare bit), Bit 0..5 = Minute
s (0..59min) *)
END_STRUCT
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.53 T_CP32Time2a

Four octets binary time format.

```
TYPE T_CP32Time2a :
STRUCT
    Milliseconds    : WORD; (* 0..59.999ms = 60sec = 1min *)
    IVResMinute     : BYTE; (* Bit 7 = IV (invalid time), Bit 6 = Res (spare bit), Bit 0..5 = Minute
s (0..59min) *)
    SURes2Hour      : BYTE; (* Bit 7 = SU (1=summer time, 0=normal time), Bits 5..6 = Res2, Bits 0..
4 = Hours (0..23) *)
END_STRUCT
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1307	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.54 T_CP56Time2a

Seven octets binary time format.

```
TYPE T_CP56Time2a :
STRUCT
    Milliseconds: WORD; (* 0..59.999ms = 60sec = 1min *)
    IVResMinute : BYTE; (* Bit 7 = IV (invalid time), Bit 6 = Res (spare bit), Bit 0..5 = Minutes (0
..59min) *)
    SURes2Hour  : BYTE; (* Bit 7 = SU (1=summer time, 0=normal time), Bits 5..6 = Res2, Bits 0..4 =
Hours (0..23) *)
    DOWDay      : BYTE; (* Bits 5..7 = Day of week (1..7, not used 0 !!!), Bits 0..4 = Day of month
(1..31)*)
    Res3Month   : BYTE; (* Bits 4..7 = Res3 (spare bits), Bits 0..3 = Month (1..12) *)
    Res4Year    : BYTE; (* Bit 7 = Res4, Bits 0..6 = Year (0..99) *)
END_STRUCTEND_TYPE
```

Day of week (DOW): 1 = Monday, 7 = Sunday, 0 = not used;

SU: 1 = Summer time, 0 = normal time;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.55 T_CP24IOA

From product version: TwinCAT PLC Library IEC60870-5-104 substation v3.0.0 / IEC60870-5-104 controlling station v1.0.0 and higher.

Structured TwinCAT object address.

```
TYPE T_CP24IOA:
STRUCT
    ioMapType    : E_IEC870_5_101IOMappingType := MAP_AREA_NONE;
    byteOffs     : WORD := 0;
    bitOffs      : BYTE(0..7) := 0;
END_STRUCT
END_TYPE
```

ioMapType: TwinCAT PLC process data area. This parameter [▶ 331] defines the mapping for the TwinCAT PLC and IEC application object process data.

byteOffs: TwinCAT PLC process data byte offset. Value range: 0..65535.

bitOffs: TwinCAT PLC process data bit offset. Value range: 0..7.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)

5.3.56 T_IEC870_5_101COTBits

```
TYPE T_IEC870_5_101COTBits :  ARRAY[0..7] OF BYTE; (* Cause of transfer bit field (e.g. used in comp
atiblity list mask).*)
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.57 Information elements

5.3.57.1 NVA

Normalized value.

5.3.57.2 S/E

Select/execute state.

- <0> = Execute;
- <1> = Select;

5.3.57.3 BSI

Bitstring of 32 bits.

5.3.57.4 SVA

Scaled value.

5.3.57.5 R32

Short floating point value.

5.3.57.6 TSC

Test command counter.

5.3.57.7 LPC

Local parameter change flag.

- <0> = No change;
- <1> = Changed;

5.3.57.8 BCR

Binary counter reading.

5.3.57.9 VTI

Value with transient state indication (8 bits).

Transient state (bit 7):

- <0> = equipment is not in transient state;
- <1> = equipment is in transient state;

Value (bits 0..6) = <-64..63>;

5.3.57.10 Other information elements

Quality descriptor

Quality descriptor

QDS

QDS

OV

OV

Overflow quality flag:

- <0> = no overflow;

- <1> = overflow;

BL

BL

Blocked quality flag:

- <0> = not blocked;
- <1> = blocked;

SB

SB

Substituted quality flag:

- <0> = not substituted;
- <1> = substituted;

NT

NT

Topical quality flag:

- <0> = topical;
- <1> = not topical;

IV

IV

Invalid quality flag:

- <0> = valid;
- <1> = invalid;

CY

CY

Carry flag:

- <0> = no carry;
- <1> = carry;

CA

CA

Adjusted flag:

- <0> = Counter was not adjusted;
- <1> = Counter was adjusted;

EI

EI

Elapsed flag:

- <0> = Elapsed time valid;
- <1> = Elapsed time not valid;

5.4 Constants

5.4.1 Group Configuration flags

The Application objects are assigned to groups with the Group-Flags. The Group-Flags for Station interrogation and Count transfer are defined as follows:

- Station interrogation: All Application objects, which are assigned to the group 1 to 16, belog automatically to the global group: IEC870_GRP_INROGEN;
- Count transfer: All Application objects, which are assigned to the group 1 to 4, belog automatically to the global group: IEC870_GRP_REQCOGEN;

You can prevent the automatical assignment of the application objects to the global group if you mask the according bits during the configuration

Examples:

IEC870_GRP_INRO3 AND NOT IEC870_GRP_INROGEN

or

IEC870_GRP_REQCO1 AND NOT IEC870_GRP_REQCOGEN

Constant	Value	Description
Station interrogation		
IEC870_GRP_INROGEN	16#00000001	Interrogated by general interrogation (station or global)
IEC870_GRP_INRO1	16#00000003	Interrogated by group 1 interrogation
IEC870_GRP_INRO2	16#00000005	Interrogated by group 2 interrogation
IEC870_GRP_INRO3	16#00000009	Interrogated by group 3 interrogation
IEC870_GRP_INRO4	16#00000011	Interrogated by group 4 interrogation
IEC870_GRP_INRO5	16#00000021	Interrogated by group 5 interrogation
IEC870_GRP_INRO6	16#00000041	Interrogated by group 6 interrogation
IEC870_GRP_INRO7	16#00000081	Interrogated by group 7 interrogation
IEC870_GRP_INRO8	16#00000101	Interrogated by group 8 interrogation
IEC870_GRP_INRO9	16#00000201	Interrogated by group 9 interrogation
IEC870_GRP_INRO10	16#00000401	Interrogated by group 10 interrogation
IEC870_GRP_INRO11	16#00000801	Interrogated by group 11 interrogation
IEC870_GRP_INRO12	16#00001001	Interrogated by group 12 interrogation
IEC870_GRP_INRO13	16#00002001	Interrogated by group 13 interrogation
IEC870_GRP_INRO14	16#00004001	Interrogated by group 14 interrogation
IEC870_GRP_INRO15	16#00008001	Interrogated by group 15 interrogation
IEC870_GRP_INRO16	16#00010001	Interrogated by group 16 interrogation
Counter interrogation		
IEC870_GRP_REQCOGEN	16#00020000	Interrogated by general counter request
IEC870_GRP_REQCO1	16#00060000	Interrogated by group 1 counter request
IEC870_GRP_REQCO2	16#000A0000	Interrogated by group 2 counter request
IEC870_GRP_REQCO3	16#00120000	Interrogated by group 3 counter request
IEC870_GRP_REQCO4	16#00220000	Interrogated by group 4 counter request
IEC870_GRP_LOCFREEZE	16#00400000	Enable cyclic local freeze of counter value
IEC870_GRP_LOCRESET	16#00800000	Enable cyclic local reset of counter value
Others		
IEC870_GRP_IV_OFF	16#01000000	Don't set invalid quality bit

Constant	Value	Description
IEC870_GRP_REVERSE	16#02000000	Object used in reverse direction (reserved)
IEC870_GRP_SELECTCMD	16#04000000	Force select/execute for this point (reserved)
not defined	16#08000000	not used
IEC870_GRP_USERCMD	16#10000000	User command (reserved)
IEC870_GRP_BACKGROUND	16#20000000	Enable background scan for this point
IEC870_GRP_PERCYC	16#40000000	Enable periodic/cyclic data for this point
IEC870_GRP_SPONTOFF	16#80000000	Disable event (spontanous) scanning of this point

Requirements

Development Enviroment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.4.2 Quality-Flags

Constant	Value	Description
IECQ_BL_ON	16#0001	Blocked
IECQ_BL_OFF	16#0002	Not blocked
IECQ_SB_ON	16#0004	Substituted
IECQ_SB_OFF	16#0008	Not substituted
IECQ_NT_ON	16#0010	Not topical
IECQ_NT_OFF	16#0020	Topical
IECQ_IV_ON	16#0040	Invalid
IECQ_IV_OFF	16#0080	Valid
IECQ_OV_ON	16#0100	Overflow
IECQ_OV_OFF	16#0200	Not overflow
IECQ_EI_ON	16#0400	Elapsed time invalid
IECQ_EI_OFF	16#0800	Elapsed time valid
IECQ_CY_ON	16#1000	Carry
IECQ_CY_OFF	16#2000	Not carry
IECQ_CA_ON	16#4000	Counter was adjusted
IECQ_CA_OFF	16#8000	Counter was not adjusted

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

6 TcIEC870_5_101Master: IEC 60870-5-101 Control Station (master)

The control stations (masters) according to IEC60870-5-101 can be realized in the TwinCAT PLC with the PLC functions and function blocks.

The PLC library has two software interfaces. The end application is imposed on one of these interfaces. The choice of interface depends on the requirements for the end application. The characteristics of both interfaces are described briefly below.

‘High level’ interface: IEC 60870-5-101 Control Station

This interface is a so-called ‘single-block solution’. All functions are encapsulated in one PLC block. The block implements the most important services and functions. This implementation is sufficient for over 90% of applications.

Pro: Very little PLC programming work is required in order to create an application; all services, such as general query, counter query, time synchronization, command execution, spontaneous data transmission etc. are already implemented in the block and are executed automatically; the mapping of the IEC<->PLC process data and that of the data points is configured via function calls; the PLC programmer does not need to be very well acquainted with the protocol standard;

Contra: The PLC application has only a small influence on the execution of the protocol; no influence on the execution of the services – these are automatically implemented internally; time stamps are automatically generated by the block and cannot be changed (handed over by externals); only the direct command execution, for example, is possible; poorer performance if there are many data points.

This interface is recommended if you:

- are not familiar with the protocol standard;
- are implementing a simple application with few data points (<1000);
- are not placing any great performance demands on the application;
- are not sending any special command execution such as Select/Execute or data + time stamp from external devices;
- do not require any functions that are not supported according to the compatibility list;

‘Low level’ interface: IEC 60870-5-101 Serial Link Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronization, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

This interface is recommended if you:

- are familiar with the protocol standard;
- are implementing a protocol converter application;
- are implementing virtually all available standard functions in the application;

- are using special functions, such as the relaying of the time stamp from a Modbus device or the gaining of control over the command execution;
- require functions that are not supported according to the compatibility list;
- have many data points (>1000) and need high performance;

Interoperability check list

for TwinCAT PLC Library: IEC 60870-5-101 control station (relating to the "high level" interface). Here you can https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11763692683/.zip

System requirements

Development environment:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT installation level:TwinCAT PLC or higher;
- TwinCAT System version 2.10.0 Build >= 1328 or higher;

Target system:

- Industrial PC or Embedded PC/CX (x86, ARM);
- Operating system:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86, ARM) (image v2.16 or higher);
- TwinCAT PLC runtime system version 2.10.0 or higher;
- Serial COM port or KL6xxx or EL6xxx bus terminal (**EL6xxx support implemented in product version 1.0.3 and higher**)

Product components

- **TcIEC870_5_101Master.Lib** (implements the Beckhoff IEC60870-5-101 controlling station). This library have to be included in the PLC project. All other libraries are included automatically.
- TcIEC870_5_101Link.Lib (base library, implements the transfer procedures for the transport of ASDU's via the serial interfaces of PC and Beckhoff KL6xxx- or EL6xxx bus terminal);
- TcIEC870_5_101.Lib (implements the connection functions and common data types);
- COMLibV2.Lib (implements the functions for the serial COM- or KL6xxx- or EL6xxx communication);

Installation on Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

The PLC libraries are copied into the folder: ..\TwinCAT\Plc\Lib.

Installation on Windows CE

You don't have to install any additional components on the CE platform.

This document is not full product manual. Please install the full version of the Beckhoff Information System.

You will find it

- on the Beckhoff Product DVDs
- on our Web-Server: http://www.beckhoff.com under download.

Examples

Example projects are located in the documentation of TwinCAT PLC library (Beckhoff Information System required).

Link to 'high level' example overview page: IEC 60870-5-101 Control Station [▶ 369];

Link to 'low level' example overview page: IEC 60870-5-101 Serial Link Interface [▶ 415];

Further Documentation

- Documentation of TwinCAT PLC Library ('low level' interface): IEC 60870-5-101 Serial Link Interface [▶ 397];
- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
- Documentation of TwinCAT PLC Library: Serial communication;
- IEC 60870-5-1 Transmission frame formats;
- IEC 60870-5-2 Link transmission procedures;
- IEC 60870-5-3 General structure of application data;
- IEC 60870-5-4 Definition and coding of application information elements;
- IEC 60870-5-5 Basic application functions;
- IEC 60870-5-101 Companion Standard for basic telecontrol tasks;
- IEC 60870-5-101:1995/A1:2000 Companion Standard for basic telecontrol tasks. Amendment 1;
- IEC 60870-5-101:1995/A2:2001 Companion Standard for basic telecontrol tasks. Amendment 2;

6.1 Introduction (tutorial)

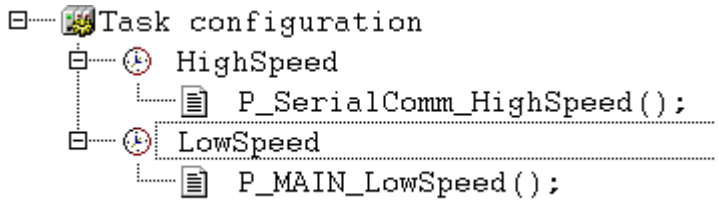
This introduction explains how to implement and configure an IEC60870-5-101 control station (master) within the TwinCAT PLC.

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

6.1.1 Creating a PLC project and integrating PLC libraries

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

1. Start TwinCAT PLC Control.
2. Create a new PLC project via File -> New. Select PC or CX (x86 or ARM) as the target system.
3. A new MAIN program block will now be created automatically. Select ST (Structured Text) as the language for the block. Confirm.
   Rename the new program block to *P_MAIN_LowSpeed*.
4. Add second program block and rename it to *P_SerialComm_HighSpeed*.
5. Go to the task configuration and configure one fast task (T#1ms) and one slow task (T#10ms). Assign the *P_SerialComm_HighSpeed* to the fast task and the *P_MAIN_LowSpeed* to the slow task (see picture below).



6. From the menu select Window -> Library Manager and then Insert -> Additional Library...

7. Select **TcIEC870_5_101Master.Lib** from the list of TwinCAT libraries and confirm.

6.1.2 The fast PLC task

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

Add the following PLC code to the declaration part of *P_SerialComm_HighSpeed*:

```
PROGRAM P_SerialComm_HighSpeed
VAR
     fbSerialLineCtrl    : FB_IEC870_SerialLineCtrl := (
                Mode         := SERIALLINEMODE_PC_COM_PORT, (*SERIALLINEMODE_KL6_5B_STANDARD *)
             Baudrate         := 19200,
                NoDatabits       := 8,
                Parity       := PARITY_EVEN,
                Stopbits         := 1,
                Handshake        := HANDSHAKE_NONE,
                ContinousMode    := FALSE );

    serial_in           AT%IB4000       : PcComInData;
    serial_out      AT%QB4000       : PcComOutData;

    KL6_in      AT%IB4100       : KL6inData5B;
    KL6_out     AT%QB4100       : KL6outData5B;

    hSerial     : T_HSERIALCTRL;
END_VAR
```

and the instance is called in the program part:

```
fbSerialLineCtrl(       pComIn      := SEL( fbSerialLineCtrl.Mode = SERIALLINEMODE_KL6_5B_STANDARD,
ADR( serial_in ), ADR( KL6_in ) ),
                    pComOut     := SEL( fbSerialLineCtrl.Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR(
 serial_out ), ADR( KL6_out ) ),
                    SizeComIn           := SEL( fbSerialLineCtrl.Mode = SERIALLINEMODE_KL6_5B_STANDA
RD, SIZEOF( serial_in ), SIZEOF( KL6_in ) ),
                    hSerial     := hSerial );
```

You can find a Twincat System Manager configuration fitting to the PLC on the example overview page.

Communication via standard PC COMx interface

- In this case the mode parameter is set to the value: **SERIALLINEMODE_PC_COM_PORT**.
- At the TwinCAT System Manager the *serial_in* and *serial_out* variables are bound to the corresponding IO variables of the serial interface.
- The interface will and has to be configured within the TwinCAT System Manager (Baudrate, Parity etc.). Other communication parameters at the FB_IEC870_SerialLineCtrl function block are not relevant at this mode.

Communication via serial Beckhoff Bus Terminals KL6xxx

- In this case the mode parameter is set to the value: **SERIALLINEMODE_KL6_5B_STANDARD**.
- At the TwinCAT System Manager the *KL6_in* and *KL6_out* variables are bound to the corresponding IO variables of the serial terminal KL6xxx.
- The interface is configured within the TwinCAT PLC by the instance of the FB_IEC870_SerialLineCtrlfunction block. Communication parameters like baud rate, parity etc. must be set via this function block.

Communication via serial Beckhoff Bus Terminals EL6xxx

```
PROGRAM P_SerialComm_HighSpeed
VAR
    fbSerialLineCtrl : FB_IEC870_SerialLineCtrl := ( Mode := SERIALLINEMODE_EL6_22B );

    EL6_in AT%IB4100 : EL6inData22B;
    EL6_out AT%QB4100 : EL6outData22B;
    hSerial : T_HSERIALCTRL;
END_VAR
```

```
fbSerialLineCtrl( pComIn := ADR( EL6_in ),
        pComOut := ADR( EL6_out ),
        SizeComIn := SIZEOF( EL6_in ),
        hSerial := hSerial );
```

- In this case the mode parameter is set to the value: **SERIALLINEMODE_EL6_22B**.
- At the TwinCAT System Manager the *EL6_in* and *EL6_out* variables are bound to the corresponding IO variables of the serial terminal EL6xxx.
- The interface will and has to be configured within the TwinCAT System Manager (Baudrate, Parity etc.). Other communication parameters at the FB_IEC870_SerialLineCtrl function block are not relevant at this mode.

6.1.3 Defining and configuring an application object database of controling station

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

Application objects = single points, double points, measured values, short floating point values, etc.

In this example the process data in monitoring direction (commands) are placed in the same memory area but on another byte/bit offset than the data of information in control direction. But you can also place the commands to the same byte/bit offset like information in control direction.

Example:

C_SC_NA_1 with IOA = 10 on the same byte/bit offset like M_SP_NA_1 with IOA = 100 (both byte ofset = 100 and bit offset = 0). In this case a change of M_SP_NA_1 value will cause a activation of the C_SC_NA_1 command.

As an example, we will configure the following application objects as part of the introductory project:

ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
M_SP_NA_1	100	General interrogation group global	0	Memory	100	0	1 bit
M_SP_TA_1	101	General interrogation group global	0	Memory	100	1	1 Bit
M_SP_TB_1	102	General interrogation group global	0	Memory	100	2	1 Bit
M_DP_NA_1	200	General interrogation group global	0	Memory	200	0	2 Bits
M_DP_TA_1	201	General interrogation group global	0	Memory	200	2	2 Bits
M_DP_TB_1	202	General interrogation group global	0	Memory	200	4	2 Bits
M_ST_NA_1	300	General interrogation group global	0	Memory	300	0	1 Byte
M_ST_TA_1	301	General interrogation group global	0	Memory	301	0	1 Byte

ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
M_ST_TB_1	302	General interrogation group global	0	Memory	302	0	1 Byte
M_BO_NA_1	400	General interrogation group global	0	Memory	400	0	4 Byte
M_BO_TA_1	401	General interrogation group global	0	Memory	404	0	4 Byte
M_BO_TB_1	402	General interrogation group global	0	Memory	408	0	4 Byte
M_ME_NA_1	500	General interrogation group global	0	Memory	500	0	2 Byte
M_ME_TA_1	501	General interrogation group global	0	Memory	502	0	2 Byte
M_ME_TD_1	502	General interrogation group global	0	Memory	504	0	2 Byte
M_ME_NB_1	600	General interrogation group global	0	Memory	600	0	2 Byte
M_ME_TB_1	601	General interrogation group global	0	Memory	602	0	2 Byte
M_ME_TE_1	602	General interrogation group global	0	Memory	604	0	2 Byte
M_ME_NC_1	700	General interrogation group global	0	Memory	700	0	4 Byte
M_ME_TC_1	701	General interrogation group global	0	Memory	704	0	4 Byte
M_ME_TF_1	702	General interrogation group global	0	Memory	708	0	4 Byte
M_IT_NA_1	800	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	800	0	4 Byte
M_IT_TA_1	801	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	804	0	4 Byte

ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
M_IT_TB_1	802	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	808	0	4 Byte
Commands							
C_SC_NA_1	10	-	0	Memory	2100	0	1 Bit
C_SC_NA_1	11	-	0	Memory	2100	1	1 Bit
C_SC_TA_1	12	-	0	Memory	2100	2	1 Bit
C_DC_NA_1	20	-	0	Memory	2200	0	2 Bit
C_DC_NA_1	21	-	0	Memory	2200	2	2 Bit
C_DC_TA_1	22	-	0	Memory	2200	4	2 Bit
C_BO_NA_1	40	-	0	Memory	2400	0	4 Byte
C_BO_NA_1	41	-	0	Memory	2404	0	4 Byte
C_BO_TA_1	42	-	0	Memory	2408	0	4 Byte
C_SE_NA_1	50	-	0	Memory	2500	0	2 Byte
C_SE_NA_1	51	-	0	Memory	2502	0	2 Byte
C_SE_TA_1	52	-	0	Memory	2504	0	2 Byte
C_SE_NB_1	60	-	0	Memory	2600	0	2 Byte
C_SE_NB_1	61	-	0	Memory	2602	0	2 Byte
C_SE_TB_1	62	-	0	Memory	2604	0	2 Byte
C_SE_NC_1	70	-	0	Memory	2700	0	4 Byte
C_SE_NC_1	71	-	0	Memory	2704	0	4 Byte
C_SE_TC_1	72	-	0	Memory	2708	0	4 Byte

Declaring a database variable

The application object database is an array variable of type ST_IEC870_5_101AODBEntry [▶ 313]. Each array element corresponds to an application object.

The array elements are not manipulated directly but through specially provided functions and a database handle (table handle). The database handle must be initialized via a F_iecCreateTableHnd [▶ 295] function call before it can be used. During this process the array elements are linked as a hash table. With a larger number of data points the hash table enables faster access to individual data points.

The maximum number of application objects is freely selectable and is only limited by the available memory. During PLC programming you have to specify a constant maximum number. The maximum number of application objects cannot be changed at runtime. In our example 50 application objects are declared. This number is sufficient for most applications. Please note that many application objects require adequate memory and runtime resources.

Define the following variable in P_MAIN_LowSpeed:

```
PROGRAM P_MAIN_LowSpeed
VAR
    AODB    : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable  : T_HAODBTable;
END_VAR
```

Configuring application objects

The object type (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 etc.), the object address and further object parameters are specified during configuration of the individual application objects.

After initialization of the database handle is the application object database (database array) is empty and must be filled with the required data (data points). The data point configuration for the central station must match the data point configuration for the substation, i.e. in the central station data points of the same type and with the same common ASDU address and the same information object address must be configured as in the substation. Other parameters such as the mapping range and byte/bit offset can be configured as required.

The following functions are available for manipulating the application database:

Function	Description
F_iecCreateTableHnd [▶ 295]	Initializes the hash table handle
F_iecAddTableEntry [▶ 296]	Configures and adds a new hash table entry
F_iecRemoveTableEntry [▶ 301]	Removes a hash table entry
F_iecLookupTableEntry [▶ 300]	Checks if a special hash table entry exists
F_iecGetPosOfTableEntry [▶ 298]	Determines the linear position of a hash table entry

The required application objects are configured during program runtime. The data base handle is transferred to the function via VAR_IN_OUT. Configuration is usually carried out once during PLC program start-up via an Init routine.

To configure the application objects during program start-up, add the following PLC code in P_MAIN_LowSpeed:

```
PROGRAM P_MAIN_LowSpeed
VAR
    AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable      : T_HAODBTable;

     init        : BOOL := TRUE;
    initError    : UDINT;
    asduAddr     : UDINT := 7;
END_VAR
```

```
IF init THEN
    init := FALSE;

    initError := F_iecCreateTableHnd( ADR( AODB ), SIZEOF( AODB ), hTable );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
            'F_iecCreateTableHnd() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF


  (* Monitored Single Points *)
 initError := F_iecAddTableEntry( M_SP_NA_1, asduAddr, 100, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
100, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_SP_NA_1, asduAddr, 101, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 100, 1, 0, hTable );
    initError := F_iecAddTableEntry( M_SP_TB_1, asduAddr, 102, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 100, 2, 0, hTable );
  (* Double Points*)
 initError := F_iecAddTableEntry( M_DP_NA_1, asduAddr, 200, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
200, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_DP_NA_1, asduAddr, 201, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 200, 2, 0, hTable );
    initError := F_iecAddTableEntry( M_DP_TB_1, asduAddr, 202, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 200, 4, 0, hTable );
  (* Regulating step value *)
 initError := F_iecAddTableEntry( M_ST_NA_1, asduAddr, 300, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
300, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ST_NA_1, asduAddr, 301, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 301, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ST_TB_1, asduAddr, 302, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 302, 0, 0, hTable );
  (* 32 bit string *)
 initError := F_iecAddTableEntry( M_BO_NA_1, asduAddr, 400, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
400, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_BO_NA_1, asduAddr, 401, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 404, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_BO_TB_1, asduAddr, 402, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 408, 0, 0, hTable );
```

```
  (* Measured value, normalized value *)
 initError := F_iecAddTableEntry( M_ME_NA_1, asduAddr, 500, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
500, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_NA_1, asduAddr, 501, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 502, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_TD_1, asduAddr, 502, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 504, 0, 0, hTable );
  (* Mesured value, scaled value *)
 initError := F_iecAddTableEntry( M_ME_NB_1, asduAddr, 600, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
600, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_NB_1, asduAddr, 601, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 602, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_TE_1, asduAddr, 602, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 604, 0, 0, hTable );
  (* Measured value , short floating point value *)
 initError := F_iecAddTableEntry( M_ME_NC_1, asduAddr, 700, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
700, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_NC_1, asduAddr, 701, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 704, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_TF_1, asduAddr, 702, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 708, 0, 0, hTable );
  (* Integrated totals *)
 initError := F_iecAddTableEntry( M_IT_NA_1, asduAddr, 800, IEC870_GRP_REQCOGEN, 0, MAP_AREA_MEMORY,
 800, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_IT_NA_1, asduAddr, 801, IEC870_GRP_REQCOGEN, 0, MAP_AREA_MEMO
RY, 804, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_IT_TB_1, asduAddr, 802, IEC870_GRP_REQCOGEN, 0, MAP_AREA_MEMO
RY, 808, 0, 0, hTable );

  (* Single commands *)
 initError := F_iecAddTableEntry( C_SC_NA_1, asduAddr, 10, 0, 0, MAP_AREA_MEMORY, 2100, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SC_NA_1, asduAddr, 11, 0, 0, MAP_AREA_MEMORY, 2100, 1, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_SC_TA_1, asduAddr, 12, 0, 0, MAP_AREA_MEMORY, 2100, 2, 0, hTa
ble );
  (* Double commands *)
 initError := F_iecAddTableEntry( C_DC_NA_1, asduAddr, 20, 0, 0, MAP_AREA_MEMORY, 2200, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_DC_NA_1, asduAddr, 21, 0, 0, MAP_AREA_MEMORY, 2200, 2, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_DC_TA_1, asduAddr, 22, 0, 0, MAP_AREA_MEMORY, 2200, 4, 0, hTa
ble );
  (* 32 bit string commands *)
 initError := F_iecAddTableEntry( C_BO_NA_1, asduAddr, 40, 0, 0, MAP_AREA_MEMORY, 2400, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_BO_NA_1, asduAddr, 41, 0, 0, MAP_AREA_MEMORY, 2404, 0, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_BO_TA_1, asduAddr, 42, 0, 0, MAP_AREA_MEMORY, 2408, 0, 0, hTa
ble );
  (* Set point, normalized values*)
 initError := F_iecAddTableEntry( C_SE_NA_1, asduAddr, 50, 0, 0, MAP_AREA_MEMORY, 2500, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SE_NA_1, asduAddr, 51, 0, 0, MAP_AREA_MEMORY, 2502, 0, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_SE_TA_1, asduAddr, 52, 0, 0, MAP_AREA_MEMORY, 2504, 0, 0, hTa
ble );
  (* Set point, scaled valuess *)
 initError := F_iecAddTableEntry( C_SE_NB_1, asduAddr, 60, 0, 0, MAP_AREA_MEMORY, 2600, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SE_NB_1, asduAddr, 61, 0, 0, MAP_AREA_MEMORY, 2602, 0, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_SE_TB_1, asduAddr, 62, 0, 0, MAP_AREA_MEMORY, 2604, 0, 0, hTa
ble );
  (* Set point, short floating point values *)
 initError := F_iecAddTableEntry( C_SE_NC_1, asduAddr, 70, 0, 0, MAP_AREA_MEMORY, 2700, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SE_NC_1, asduAddr, 71, 0, 0, MAP_AREA_MEMORY, 2704, 0, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_SE_TC_1, asduAddr, 72, 0, 0, MAP_AREA_MEMORY, 2708, 0, 0, hTa
ble );

END_IF
```

6.1.4 Mapping of PLC and IEC process data

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

The TwinCAT PLC process data are cyclically mapped (copied) into the IEC process data (application objects) and vice versa at program runtime. For mapping of the IEC<->PLC process data up to 4 process data areas (IO inputs, IO outputs, memory area, data area) can be declared as buffer variables in the PLC program. The byte size of the buffers is freely selectable and may be different for each area. Its not necessary to declare unused buffer areas.

In our introductory example we declare 4 PLC process data areas with 3000 bytes each:

```
PROGRAM P_MAIN_LowSpeed
VAR
    AODB    : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable  : T_HAODBTable;

    init    : BOOL := TRUE;
    initError       : UDINT;
    asduAddr    : UDINT := 7;


     inputs AT%IB0   : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0   : ARRAY[0..2999] OF BYTE;
    memory AT%MB0    : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;

END_VAR
```

How the process data are to be mapped at runtime is specified during configuration of the application objects via the F_iecAddTableEntry [▶ 296] function.
See also in: Declaration and configuration of application objects [▶ 354].

The buffer variables were now declared as byte arrays. To improve access to the required data we define the individual variables a second time and allocate them to the corresponding byte/bit offset addresses. In the event of a change in the byte array the corresponding variable is changed at the same time and vice versa, although this is not compulsory. The bytes/bits of the byte array buffer variables can be accessed directly.

```
VAR_GLOBAL
(* Memory offset 0..99 unused *)
  (* Single points *)
 msgSingle_0     AT%MX100.0 : BOOL;
    msgSingle_1     AT%MX100.1 : BOOL;
    msgSingle_2     AT%MX100.2 : BOOL;

  (* Double points *)
    (*      Bit 0..1 = first double point,
        Bit 2..3 = second double point,
        Bit 4..5 = third double point,
        Bit 6..7 = fourth double point *)
    msgDouble_0     AT%MB200    : BYTE;

  (* Regulating step values *)
    msgStep_0           AT%MB300    : BYTE;
    msgStep_1           AT%MB301    : BYTE;
    msgStep_2           AT%MB302    : BYTE;

  (* 32 bit strings *)
 msgBitStr_0     AT%MD400    : DWORD;
    msgBitStr_1     AT%MD404    : DWORD;
    msgBitStr_2     AT%MD408    : DWORD;

  (* Measured values, normalized values *)
 msgNormalized_0     AT%MW500    : WORD;
    msgNormalized_1     AT%MW502    : WORD;
    msgNormalized_2     AT%MW504    : WORD;

  (* Mesured values, scaled values *)
 msgScaled_0     AT%MW600    : INT;
    msgScaled_1     AT%MW602    : INT;
    msgScaled_2     AT%MW604    : INT;

  (* Measured values, short floating point values *)
 msgFloating_0   AT%MD700    : REAL;
    msgFloating_1   AT%MD704    : REAL;
    msgFloating_2   AT%MD708    : REAL;

  (* Integrated totals *)
 msgTotal_0      AT%MD800    : UDINT;
    msgTotal_1      AT%MD804    : UDINT;
```

```
    msgTotal_2      AT%MD808    : UDINT;

  (* Single commands *)
 cmdSingle_0     AT%MX2100.0 : BOOL;
    cmdSingle_1     AT%MX2100.1 : BOOL;
    cmdSingle_2     AT%MX2100.2 : BOOL;

  (* Double commands *)
    (*      Bit 0..1 = first double command,
        Bit 2..3 = second double command,
        Bit 4..5 = third double command,
        Bit 6..7 = fourth double command *)
 cmdDouble_0     AT%MB2200       : BYTE;


  (* 32 bit string commands *)
 cmdBitStr_0     AT%MD2400       : DWORD;
    cmdBitStr_1     AT%MD2404       : DWORD;
    cmdBitStr_2     AT%MD2408       : DWORD;

  (* Set point, normalized values *)
 cmdNormalized_0     AT%MW2500       : WORD;
    cmdNormalized_1     AT%MW2502       : WORD;
    cmdNormalized_2     AT%MW2504       : WORD;

  (* Set point, scaled values *)
 cmdScaled_0     AT%MW2600       : INT;
    cmdScaled_1     AT%MW2602       : INT;
    cmdScaled_2     AT%MW2604       : INT;

  (* Set point, short floating point values *)
 cmdFloating_0   AT%MD2700       : REAL;
    cmdFloating_1   AT%MD2704       : REAL;
    cmdFloating_2   AT%MD2708       : REAL;
END_VAR
```

Mapping of the IEC<->PLC process data in the controlling station

Process data in monitoring direction (slave->master information)

Example 1

Single point information (M_SP_NA_1) with the IOA = 100, PLC memory area, byte offset = 100, bit offset = 0.

Controlled station -> ... -> Controlling station FB -> memory[100].0 == msgSingle_0

Example 2

Measured value, short floating point value (M_ME_NC_1) with the IOA = 700, PLC memory area, byte offset = 700, bit offset = 0 (irrelevant).

Controlled station -> ... -> Controlling station FB -> memory[700..703] == msgFloating_0

Process data in control direction (master->slave commands)

Example 1

Single command state (C_SC_NA_1) with the IOA = 10, PLC memory area, byte offset = 2100, bit offset = 0.

cmdSingle_0 == memory[2100].0 -> Controlling station FB -> ... -> Controlled station

Example 2

Set point, short floating point value (C_SE_NC_1) with the IOA = 70, PLC memory area, byte offset = 2700, bit offset = 0 (irrelevant).

cmdFloating_0 == memory[2700..2703] -> Controlling station FB -> ... -> Controlled station

6.1.5 Declaring and calling an instance of the IEC60870-5-101 controlling station

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

The complete functionality of a controlling station is encapsulated in the FB_IEC870_5_101Master function block. An instance can be used to establish a connection to the controlled station. For establishing a further connection a further instance of the function block can be declared.

Add the following PLC code to the declaration part of P_MAIN_LowSpeed:

```
PROGRAM P_MAIN_LowSpeed
VAR
    AODB        : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable      : T_HAODBTable;

    init        : BOOL := TRUE;
    initError       : UDINT;
    asduAddr        : UDINT := 7;

    inputs AT%IB0   : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..2999] OF BYTE;
    memory AT%MB0   : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;


     bEnable         : BOOL := TRUE;
    client      : FB_IEC870_5_101Master;

ND_VAR
```

and the instance is called in the program part:

```
IF init THEN
    init := FALSE;
...
ELSE
...

    client(
        pAOEntries := ADR( AODB ),
        cbAOEntries := SIZEOF( AODB ),
        pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
        cbOutputs := SIZEOF( outputs ),
        pMemory := ADR( memory ),
        cbMemory := SIZEOF( memory ),
        pData := ADR( data ),
        cbData := SIZEOF( data ),
        bEnable := bEnable,
        hSerial := P_SerialComm_HighSpeed.hSerial,
        hTable := hTable );

...
END_IF
```

6.1.6 IEC60870-5-101 protocol parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

The behaviour of the controlling station can be adapted to the requirements of the slave via the IEC60870-5-101 protocol parameters. Most parameters have preallocated default values that do not have to be changed.

In our example we change the link address, link address octet size and the cycle time of polling for class 1 and class 2 data.

```
IF init THEN
    init := FALSE;
...
  (*Configure protocol parameter *)
```

```
 client.protPara.linkAddr := 220;(* link address of remote slave *)
 client.protPara.eLinkAddrSize := eIEC870_LinkAddr_TwoOctets; (* link addres octet size *)
 client.protPara.tClass1Poll := T#0ms; (* poll class 1 data with max. speed *)
 client.protPara.tClass2Poll := T#0ms; (* poll class 2 data with max. speed *)
...

ELSE
    client( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all transfer protocol parameters can be found here: ST_IEC870_5_101PotocolParams [▶ 407].

6.1.7 System parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

The common ASDU address and the user functions of the controlling station are configured via the system parameters.

In our introduction we configure the following system parameters:

- The common ASDU address is set to 7. (*asduAddr*);
- The octet size of cause of transmission is set to two octets (one octet COT + one octet originator address) (*eCOTSize*);
- The octet size of common asdu address is set to two octets (*eAsduAddrSize*);
- The octet size of information object address is set to three octets (*eObjAddrSize*);
- Logging of debugging messages in the application log is activated (*dbgMode*). Logging of device state changes and link layer errors is enabled;

Add the following PLC code to your PLC project:

```
IF init THEN
    init := FALSE;
...


    client.sysPara.asduAddr := 7;
    client.sysPara.asduFmt.eCOTSize := eIEC870_COT_TwoOctets;(* cause of transfer octet size *)
 client.sysPara.asduFmt.eAsduAddrSize := eIEC870_AsduAddr_TwoOctets;
(* common ASDU address octet size *)
 client.sysPara.asduFmt.eObjAddrSize := eIEC870_ObjAddr_ThreeOctets;
(* information object address octet size *)
 client.sysPara.dbgMode := IEC870_DEBUGMODE_DEVSTATE OR IEC870_DEBUGMODE_LINKERROR;
(* IEC870_DEBUGMODE_ASDU OR IEC870_DEBUGMODE_LINKLAYER *)

...

ELSE
    client( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all system parameters can be found here: ST_IEC870_5_101SystemParams [▶ 317].

6.1.8 Initialisation sequence

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

```
client.acqPara.arrInitSeq[0] := eIEC870_ISTEP_TEST; (* Send test command *)
client.acqPara.arrInitSeq[1] := eIEC870_ISTEP_CLOCK; (* Send clock synchronization command *)
client.acqPara.arrInitSeq[2] := eIEC870_ISTEP_GENRO; (* Send general interrogation command *)
client.acqPara.arrInitSeq[3] := eIEC870_ISTEP_CORO; (* Send counter interrogation command *)
client.acqPara.arrInitSeq[5] := eIEC870_ISTEP_UNUSED; (* not used *)
```

6.1.9 Station interrogation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

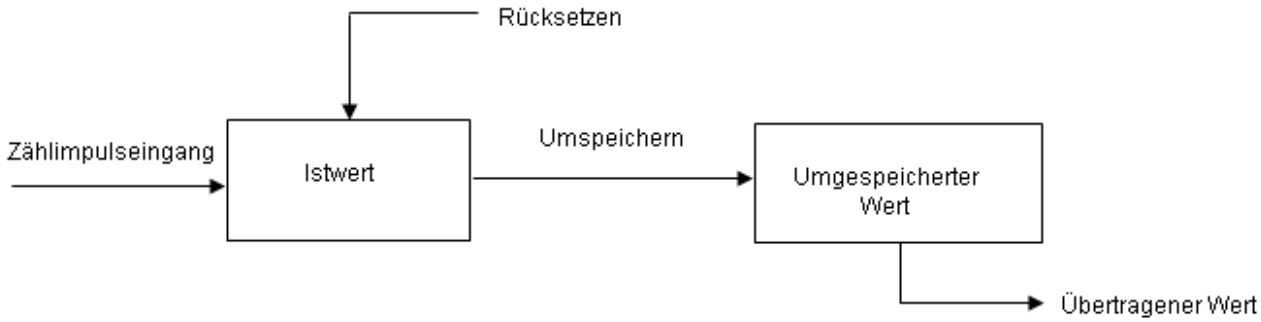
The station interrogation command is initiated by the central station. The ID field of the command also contains the group (1 to 16 or general). The substation transfers the application objects associated with this group to the central station with cause of transmission *<20>* to *<36>*. Application objects with time tags are transferred without time tags.

```
client.acqPara.arrGenro[0].tPollCycle := T#60s;
client.acqPara.arrGenro[0].eQOI := eIEC870_QOI_INROGEN;
client.acqPara.arrGenro[0].bEnable := TRUE;
```

6.1.10 Transfer of integrated totals (counter interrogation)

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

General count transfer model:



The actual values are added by counters. Via a re-store command that is either received by the central station or generated locally (in the substation), the actual values can be re-stored (copied) periodically into re-stored values. After freezing, the recorded value is either reset to zero (logging of incremental values), or the counter continues adding up (logging of counter readings).

Application objects with counts are assigned to groups. The groups are frozen individually, reset, or transferred. The central station sends count query commands to the substation. The task to be carried out (FRZ) and the group (RQT) are specified in an ID field of the command (QCC).

The allocation of the application objects to the individual groups (1 to 4 or general) is specified via the group flag parameter during configuration. There are four operating modes for recording counter readings and incremental values. Each mode includes notes about the configuration of the system parameters or the application objects.

Mode A: local freeze with spontaneous transfer

The substation internally initiates freeze or freeze with reset. The frozen counts are transferred spontaneously, once the freeze or freeze with reset function was executed. In this mode the central station does not issue any count query commands.

Configuration of the system parameters:

Configuration of the application objects:

Mode B: Local freeze with counter interrogation

The substation internally initiates freeze or freeze with reset. The central station queries the frozen counts via count query commands. In this case the central station must not use freeze or freeze with reset in the command ID field (FRZ=0). The counts are queried generally or in groups 1 to 4.

Configuration of the system parameters:

Configuration of the application objects:

Mode C: The central station initiates freeze, freeze with reset, or reset

The central station periodically issues a count query command to the substation for controlling the freeze or (and) reset process. This command does not result in a count transfer. The central station sends a subsequent count query command for collecting the frozen counts. This is like operating mode B.

Configuration of the system parameters:

Configuration of the application objects:

Mode D: The central station initiates freeze and (or) reset, and the frozen values are transferred spontaneously

This mode is a combination of the count command from the central station as in mode C and spontaneous transfer of the counts as in mode A.

Configuration of the system parameters:

Configuration of the application objects:

6.1.11 Clock (time) synchronisation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

Under construction...

6.1.12 Testing the communication

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

You can simulate the command transmission. Setting the *bExecuteCmd* variable to TRUE enables and to FALSE disables the simulation. In our example single command (C_SC_NA_1, IOA = 10) is send cyclicaly every 10 seconds to the controlled station.

```
PROGRAM MAIN
VAR
    ...

    bExecuteCmd : BOOL;
    timer : TON;

    ...
END_VAR
```

```
...

(* Simple command simulation *)
timer( IN := bExecuteCmd, PT := T#10s ); (* Send cyclic command *)
IF timer.Q THEN
    timer( IN := FALSE );
    cmdSingle_0 := NOT cmdSingle_0; (* toggle single command ON<->OFF *)

(*      cmdDouble_0 := SEL( cmdDouble_0 = 1, 1, 2 );

    cmdBitStr_0 := cmdBitStr_0 + 1;

    cmdNormalized_0 := cmdNormalized_0 + 2;
```

```
    cmdScaled_0 := cmdScaled_0 + 4;

    cmdFloating_0 := cmdFloating_0 + 1.2; *)
END_IF

...
```

6.1.13 Protocol and data transmission errors

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip

The station error messages are placed into a FIFO. Up to 10 error messages can be stored. At fatal communication errors (e.g. error of the linking layer, the check sum of the frame doesn't fit) the connection is cut and has to be build on new. Errors within the application layer (e.g. the ADSU send buffer is overflowed by to many frames) are only logged and doesn't cut the connection.

For this errors it is also possible to cut the connection out of the application. In addition to the error code also the error source is notified in the error message. This simplifies the localization of an error.

Example

The occurring error messages of an IEC 60870-5-101 control station can be read out via the request:

```
PROGRAM MAIN
VAR
...

    client : FB_IEC870_5_101Master;

...
END_VAR
```

```
....


REPEAT
    client.system.device.errors.RemoveError( );
    IF client.system.device.errors.bOk THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'IEC60870-5-101 master error: 0x%s',
            DWORD_TO_HEXSTR( client.system.device.errors.getError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT client.system.device.errors.bOk
END_REPEAT
...
```

6.2 Quick start

Simple projects with complete sources can be found here: IEC60870-5-101 control station [▶ 369].

Interoperability check list can be found here: Interoperability check list

Communication- and/or device error code overview can be found here: Error codes

Detailed introduction to implementation of control station in TwinCAT PLC can be found here: TUTORIAL [▶ 352]

Short quide

Application object database

The application object database of the central station must be configured as hash table with the function F_iecCreateTableHnd [▶ 295]. The individual array elements are linked with each other in the form of a hash table. This enables faster access to the individual data points, but also has certain disadvantages that must be taken into account:

- The size of the application database (array size) must not modify at runtime (e.g. through online change). The central station stops execution immediately and reports an error. The reason: The hash table links no longer match. When the program is modified it is best to load the complete project into the runtime system.
- The individual array elements must not be accessed via the index but via the special functions (e.g. F_iecAddTableEntry [▶ 296] etc.).
- With indexed table element access the internal configuration parameters must not be overwritten or modified. If the type, the ASDU address or the object address is changed the data point can no longer be found. To reconfigure a data point it should first be removed from the table via function call F_iecRemoveTableEntry [▶ 301]. The new data point can then be added.

An implementation in the form of linear table would mean that for each received ASDU (data unit) the central station would have to search the complete array for a suitable element. With many data points this would lead to long execution times.

Protocol parameters

Most protocol parameters are preconfigured with default values and do not have to be set explicitly.

System parameters

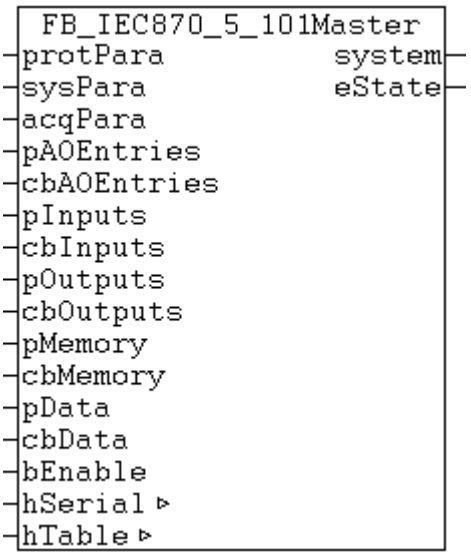
The system parameters are also preconfigured with default values. During commissioning it is useful to activate debugging output (*dbgMode*) to be able to locate any errors.

Parameters for cyclic data acquisition

The following parameters are preconfigured with default values:

- Initialisation sequence (consisting of a test command, time synchronisation, station query and counter query);
- Cyclic commands:
  - Test command every 60 s;
  - Time synchronisation every 60 s;
  - Group station query: generally, every 60 s;
  - Group counter query: generally, every 60 s;

6.2.1 FB_IEC870_5_101Master



An instance of the FB_IEC870_5_101Master function block can be used to implement an IEC60870-5-101 central station (master) in the TwinCAT PLC. A connection to the slave is established for each instance of the function block. Normally the data exchange is started automatically once the connection is established. This is the default configuration of the function block.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial  : T_HSERIALCTRL;
    hTable   : T_HAODBTable;
END_VAR
```

hSerial [▶ 415]:

hTable: Application object database handle [▶ 343] (hash table handle). The table handle must be initialised once with the function F_iecCreateTableHnd [▶ 295] before it can be used.

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_101ProtocolParams := ( eType := eIEC870_101_MASTER );
    sysPara     : ST_IEC870_5_101SystemParams := ( bEndOfInit := FALSE );
    acqPara     : ST_IEC870_5_101AcquisitionParams;
    pAOEntries  : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF ST_IEC870_5_101AODBEntry := 0;
    cbAOEntries : UDINT := 0;
    pInputs     : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbInputs    : UDINT := 0;
    pOutputs    : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbOutputs   : UDINT := 0;
    pMemory     : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbMemory    : UDINT := 0;
    pData       : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbData      : UDINT := 0;
    bEnable     : BOOL := TRUE;
END_VAR
```

protPara: IEC60870-5-101 protocol parameter [▶ 407].

sysPara: System parameter [▶ 317].

acqPara: Parameter [▶ 320] for cyclic data acquisition.

pAOEntries: Address of the application object database variables.

cbAOEntries: Byte size of the application object database variables [▶ 313].

pInputs: Address of the PLC process data area for the inputs.

cbInputs: Byte size of the PLC process data area for the inputs.

pOutputs: Address of the PLC process data area for the outputs.

cbOutputs: Byte size of the PLC process data area for the outputs.

pMamory: Address of the PLC process data area for the flags.

cbMamory: Byte size of the PLC process data area for the flags.

pData: Address of the PLC data area.

cbData: Byte size of the PLC data area.

bEnable : Activates/deactivates the function block (communication and connections).

The addresses can be determined with the ADR operator and the byte sizes with the SIZEOF operator.

VAR_OUTPUT

```
VAR_OUTPUT
    system  : ST_IEC870_5_101ExSystemInterface;
    eState  : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED;
END_VAR
```

system: system interface [▶ 368]. This variable is used by other IEC application functions as a communication interface for the IEC device (here: central station).

- Member variable *system.device* is expected by the F_iecSetAOQuality [▶ 282] function as VAR_IN_OUT parameter, for example.
- Member variable *system.device.errors* is a device error FIFO. The PLC application can read and analyse registered errors.

eState: Status [▶ 413] of the connection with the slave..

Example in ST: IEC60870-5-101 central station [▶ 369]

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1328	PC or CX (X86, ARM)	TcIEC870_5_101Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; COMLibV2.Lib; TcIEC870_5_101Link.Lib; TcIEC870_5_101.Lib; are included automatically)

6.2.2 F_GetVersionTcIEC870_5_101Master

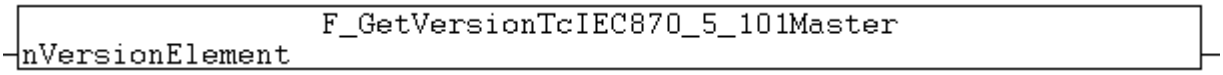


Fig. 7: F_GetVersionTcIEC870_5_101Master

This function reads version information from the plc library.

FUNCTION F_GetVersionTcIEC870_5_101Master: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1328	PC or CX (x86, ARM)	TcIEC870_5_101Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; COMLibV2.Lib; TcIEC870_5_101Link.Lib; TcIEC870_5_101.Lib are included automatically)

6.2.3 ST_IEC870_5_101ExSystemInterface

```
TYPE ST_IEC870_5_101ExSystemInterface :
STRUCT
    device   : ST_IEC870_5_101DeviceInterface;
    service  : ST_IEC870_5_101SystemServices;
    hSOTable : T_HAODBTABLE;
END_STRUCT
END_TYPE
```

device: Communication interface [▶ 319] of IEC device;

service: IEC device service;

hSOTable : System object database handle [▶ 343];

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1328	PC or CX (x86, ARM)	TcIEC870_5_101Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; COMLibv2.Lib; TcIEC870_5_101Link.Lib; TcIEC870_5_101.Lib are included automatically)

6.3 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.).
2. Compare the interoperability check list of control station and controlled station.
3. Check the IO configuration and mapping of PLC variables in TwinCAT System Manager (configuration of serial interface, baudrate, parity, stoppbits settings). Compare them to the parameters set in the controlled station.
4. Check whether the function block issues an error code/source [▶ 365]. The documentation for the error codes can be found here: Overview of error codes.
5. Check the protocol parameters [▶ 407] that are transferred to the function block (link address, link address octet size, FRAMELength etc.). Compare them to the parameters set in the controlled station.
6. Check the system parameter [▶ 317] that are transferred to the function block (ASDU address, ASDU address octet size, information object address octet size, cause of transfer octet size, etc.). Compare them to the the parameters set in the controlled station.
7. Check the acquisition parameters [▶ 320] that are transferred to the function block (initialization sequence, cyclic general interrogation, cyclic counter interrogation, cyclic test commands, etc.).
8. Check the configured data points (type, information object address etc.).
9. Check if the controlled station issues an error code.
10. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings.

6.4 Examples

Used controlling station configuration parameters:

- Standard PC interface: **COM1**
- **19200 Baud**
- Link address: **220**
- Link address size: **2 octets**
- Cause of transfer size: **2 octets**
- Originator address: **1**
- Common ASDU address: **7**
- Common ASDU address size: **2 octets**
- Information object address size: **3 octets**

Requirements

PLC project	TwinCAT System Manager Configuration	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697520139/.zip	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697520139/.zip	Simple IEC60870-5-101 controlling station. Very **small** PLC project with one single point in monitoring (IOA = 100) and one single command in control direction (IOA = 10).
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11697518731/.zip	Simple **IEC60870-5-101 controlling station (master)**

7 TcIEC870_5_101Slave: IEC 60870-5-101 Controlled Station (slave)

The substations (slaves) according to IEC60870-5-101 can be realised in the TwinCAT PLC with the PLC functions and function blocks.

The PLC library has two software interfaces. The end application is imposed on one of these interfaces. The choice of interface depends on the requirements for the end application. The characteristics of both interfaces are described briefly below.

'High level' interface: IEC 60870-5-101 Controlled Station

This interface is a so-called ‘single-block solution’. All functions are encapsulated in one PLC block. The block implements the most important services and functions. This implementation is sufficient for over 90% of applications.

Pro: Very little PLC programming work is required in order to create an application; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. are already implemented in the block and are executed automatically; the mapping of the IEC<->PLC process data and that of the data points is configured via function calls; the PLC programmer does not need to be very well acquainted with the protocol standard;

Contra: The PLC application has only a small influence on the execution of the protocol; no influence on the execution of the services – these are automatically implemented internally; time stamps are automatically generated by the block and cannot be changed (handed over by externals); only the direct command execution, for example, is possible; poorer performance if there are many data points.

This interface is recommended if you:

- are not familiar with the protocol standard;
- are implementing a simple application with few data points (<1000);
- are not placing any great performance demands on the application;
- are not sending any special command execution such as Select/Execute or data + time stamp from external devices;
- do not require any functions that are not supported according to the compatibility list;

'Low level' interface: IEC 60870-5-101 Serial Link Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

This interface is recommended if you:

- are familiar with the protocol standard;
- are implementing a protocol converter application;
- are implementing virtually all available standard functions in the application;

- are using special functions, such as the relaying of the time stamp from a Modbus device or the gaining of control over the command execution;
- require functions that are not supported according to the compatibility list;
- have many data points (>1000) and need high performance;

Interoperability check list

for TwinCAT PLC Library: IEC 60870-5-101 controlled station (relating to the "high level" interface). Here you can https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11763793547/.zip

System requirements

Development environment:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT installation level:TwinCAT PLC or higher;
- TwinCAT System version 2.9.0 Build >= 1030 (CX (ARM) Build >=1301) or higher;

Target system:

- Industrial PC or Embedded PC/CX (x86, ARM);
- Operating system:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86) (image v1.75 or higher);
  - Windows CE (ARM) (image v2.13 or higher);
- TwinCAT PLC runtime system version 2.9.0 or higher;
- Serial COM port or KL6xxx or EL6xxx bus terminal (**EL6xxx support implemented in product version 3.0.9 and higher**);

Product components

- **TcIEC870_5_101Slave.Lib** (implements the Beckhoff IEC60870-5-101 Substation). This library have to be included in the PLC project. All other libraries are included automatically.
- TcIEC870_5_101.Lib (implements the connection functions and common data types);
- TcIEC870_5_101Link.Lib (implements the transfer procedures for the transport of ASDU's via the serial interfaces of PC and Beckhoff KL6xxx or EL6xxx bus terminal);
- COMLibV2.Lib (implements the functions for the serial COM- or KL6xxx or EL6xxx communication);

Installation on Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

The PLC libraries are copied into the folder: ..\TwinCAT\Plc\Lib.

Installation on Windows CE

You don't have to install any additional components on the CE platform.

This document is not full product manual. Please install the full version of the Beckhoff Information System.

You will find it

- on the Beckhoff Product DVDs

- on our Web-Server: http://www.beckhoff.com under download.

Examples

Example projects are in the Beckhoff Information System documentation of TwinCAT PLC library.

Link to 'high level' example overview page: IEC 60870-5-101 controlled station [▶ 396];

Link to 'low level' example overview page: IEC 60870-5-101 Serial Link Interface [▶ 415];

Further Documentation

- Documentation of TwinCAT PLC Library ('low level' interface): IEC 60870-5-101 Serial Link interface [▶ 397];
- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
- Documentation of TwinCAT PLC Library: Serial communication;
- IEC 60870-5-1 Transmission frame formats;
- IEC 60870-5-2 Link transmission procedures;
- IEC 60870-5-3 General structure of application data;
- IEC 60870-5-4 Definition and coding of application information elements;
- IEC 60870-5-5 Basic application functions;
- IEC 60870-5-101 Companion Standard for basic telecontrol tasks;
- IEC 60870-5-101:1995/A1:2000 Companion Standard for basic telecontrol tasks. Amendment 1;
- IEC 60870-5-101:1995/A2:2001 Companion Standard for basic telecontrol tasks. Amendment 2;

7.1 Introduction (tutorial)

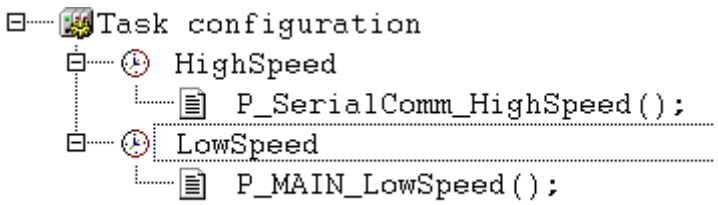
This introduction explains how to implement and configure an IEC60870-5-101 substation (slave) within the TwinCAT PLC.

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

7.1.1 Creating a PLC project and integrating PLC libraries

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

1. Start TwinCAT PLC Control.
2. Create a new PLC project via File -> New. Select PC or CX (x86 or ARM) as the target system.
3. A new MAIN program block will now be created automatically. Select ST (Structured Text) as the language for the block. Confirm. Rename the new program block to *P_MAIN_LowSpeed*.
4. Add second program block and rename it to *P_SerialComm_HighSpeed*.
5. Go to the task configuration and configure one fast task (T#1ms) and one slow task (T#10ms). Assign the *P_SerialComm_HighSpeed* to the fast task and the *P_MAIN_LowSpeed* to the slow task (see picture below).



1. From the menu select Window -> Library Manager and then Insert -> Additional Library...
2. Select **TcIEC870_5_101Slave.Lib** from the list of TwinCAT libraries and confirm.

7.1.2 The fast PLC task

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

Add the following PLC code to the declaration part of *P_SerialComm_HighSpeed*:

```
PROGRAM P_SerialComm_HighSpeed
VAR
     fbSerialLineCtrl    : FB_IEC870_SerialLineCtrl := (
                Mode         := SERIALLINEMODE_PC_COM_PORT, (*SERIALLINEMODE_KL6_5B_STANDARD *)
             Baudrate         := 19200,
                NoDatabits       := 8,
                Parity       := PARITY_EVEN,
                Stopbits         := 1,
                Handshake        := HANDSHAKE_NONE,
                ContinousMode    := FALSE );

    serial_in           AT%IB4000       : PcComInData;
    serial_out      AT%QB4000       : PcComOutData;

    KL6_in      AT%IB4100       : KL6inData5B;
    KL6_out     AT%QB4100       : KL6outData5B;

    hSerial     : T_HSERIALCTRL;
END_VAR
```

and the instance is called in the program part:

```
fbSerialLineCtrl(       pComIn      := SEL( fbSerialLineCtrl.Mode = SERIALLINEMODE_KL6_5B_STANDARD,
ADR( serial_in ), ADR( KL6_in ) ),
                    pComOut     := SEL( fbSerialLineCtrl.Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR(
 serial_out ), ADR( KL6_out ) ),
                    SizeComIn           := SEL( fbSerialLineCtrl.Mode = SERIALLINEMODE_KL6_5B_STANDA
RD, SIZEOF( serial_in ), SIZEOF( KL6_in ) ),
                    hSerial     := hSerial );
```

You can find a Twincat System Manager configuration fitting to the PLC on the example overview page.

Communication via standard PC COMx interface

- In this case the mode parameter is set to the value: **SERIALLINEMODE_PC_COM_PORT**.
- At the TwinCAT System Manager the *serial_in* and *serial_out* variables are bound to the corresponding IO variables of the serial interface.
- The interface will and has to be configured within the TwinCAT System Manager (Baudrate, Parity etc.). Other communication parameters at the FB_IEC870_SerialLineCtrl function block are not relevant at this mode.

Communication via serial Beckhoff Bus Terminals KL6xxx

- In this case the mode parameter is set to the value: **SERIALLINEMODE_KL6_5B_STANDARD**.
- At the TwinCAT System Manager the *KL6_in* and *KL6_out* variables are bound to the corresponding IO variables of the serial terminal KL6xxx.

- The interface is configured within the TwinCAT PLC by the instance of the FB_IEC870_SerialLineCtrlfunction block. Communication parameters like baud rate, parity etc. must be set via this function block.

Communication via serial Beckhoff Bus Terminals EL6xxx

```
PROGRAM P_SerialComm_HighSpeed
VAR
    fbSerialLineCtrl : FB_IEC870_SerialLineCtrl := ( Mode := SERIALLINEMODE_EL6_22B );

    EL6_in AT%IB4100 : EL6inData22B;
    EL6_out AT%QB4100 : EL6outData22B;
    hSerial : T_HSERIALCTRL;
END_VAR
```

```
fbSerialLineCtrl( pComIn := ADR( EL6_in ),
        pComOut := ADR( EL6_out ),
        SizeComIn := SIZEOF( EL6_in ),
        hSerial := hSerial );
```

- In this case the mode parameter is set to the value: **SERIALLINEMODE_EL6_22B**.
- At the TwinCAT System Manager the *EL6_in* and *EL6_out* variables are bound to the corresponding IO variables of the serial terminal EL6xxx.
- The interface will and has to be configured within the TwinCAT System Manager (Baudrate, Parity etc.). Other communication parameters at the FB_IEC870_SerialLineCtrl function block are not relevant at this mode.

7.1.3 Defining and configuring an application object database of controlled station

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

Application objects = single points, double points, measured values, short floating point values, etc.

In this example the commands have been configured in a way that the process data are placed in the same memory area but on another byte/bit offset than the data of information in control direction. But you can also place the commands to the same byte/bit offset like information in control direction.

Example:

C_SC_NA_1 with IOA = 10 on the same byte/bit offset like M_SP_NA_1 with IOA = 100 (both byte ofset = 100 and bit offset = 0).
In this case a change of value via a command from the control station will cause a transfer of the M_SP_NA_1 with the object address 100 and transfer cause <11> (returned by remote command).

As an example we will configure the following application objects as part of the introductory project:

Requirements

Array element	ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
0	M_SP_NA_1	100	General interrogation group global	0	Memory	100	0	1 bit
1	M_SP_NA_1	101	General interrogation group global	0	Memory	100	1	1 Bit

Array element	ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
2	M_SP_TB_1	102	General interrogation group global	0	Memory	100	2	1 Bit
3	M_DP_NA_1	200	General interrogation group global	0	Memory	200	0	2 Bits
4	M_DP_NA_1	201	General interrogation group global	0	Memory	200	2	2 Bits
5	M_DP_TB_1	202	General interrogation group global	0	Memory	200	4	2 Bits
6	M_ST_NA_1	300	General interrogation group global	0	Memory	300	0	1 Byte
7	M_ST_NA_1	301	General interrogation group global	0	Memory	301	0	1 Byte
8	M_ST_TB_1	302	General interrogation group global	0	Memory	302	0	1 Byte
9	M_BO_NA_1	400	General interrogation group global	0	Memory	400	0	4 Byte
10	M_BO_NA_1	401	General interrogation group global	0	Memory	404	0	4 Byte
11	M_BO_TB_1	402	General interrogation group global	0	Memory	408	0	4 Byte
12	M_ME_NA_1	500	General interrogation group global	0	Memory	500	0	2 Byte
13	M_ME_NA_1	501	General interrogation group global	0	Memory	502	0	2 Byte
14	M_ME_TD_1	502	General interrogation group global	0	Memory	504	0	2 Byte
15	M_ME_NB_1	600	General interrogation group global	0	Memory	600	0	2 Byte
16	M_ME_NB_1	601	General interrogation group global	0	Memory	602	0	2 Byte
17	M_ME_TE_1	602	General interrogation group global	0	Memory	604	0	2 Byte
18	M_ME_NC_1	700	General interrogation group global	0	Memory	700	0	4 Byte

Array element	ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
19	M_ME_NC_1	701	General interrogation group global	0	Memory	704	0	4 Byte
20	M_ME_TF_1	702	General interrogation group global	0	Memory	708	0	4 Byte
21	M_IT_NA_1	800	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	800	0	4 Byte
22	M_IT_NA_1	801	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	804	0	4 Byte
23	M_IT_TB_1	802	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	808	0	4 Byte
Commands								
24	C_SC_NA_1	10	-	0	Memory	2100	0	1 Bit
25	C_SC_NA_1	11	-	0	Memory	2100	1	1 Bit
26	C_SC_TA_1	12	-	0	Memory	2100	2	1 Bit
27	C_DC_NA_1	20	-	0	Memory	2200	0	2 Bit
28	C_DC_NA_1	21	-	0	Memory	2200	2	2 Bit
29	C_DC_TA_1	22	-	0	Memory	2200	4	2 Bit
30	C_RC_NA_1	30	-	0	Memory	2300	0	1 Byte
31	C_RC_NA_1	31	-	0	Memory	2301	0	1 Byte
32	C_RC_TA_1	32	-	0	Memory	2302	0	1 Byte
33	C_BO_NA_1	40	-	0	Memory	2400	0	4 Byte
34	C_BO_NA_1	41	-	0	Memory	2404	0	4 Byte
35	C_BO_TA_1	42	-	0	Memory	2408	0	4 Byte
36	C_SE_NA_1	50	-	0	Memory	2500	0	2 Byte
37	C_SE_NA_1	51	-	0	Memory	2502	0	2 Byte
38	C_SE_TA_1	52	-	0	Memory	2504	0	2 Byte
39	C_SE_NB_1	60	-	0	Memory	2600	0	2 Byte
40	C_SE_NB_1	61	-	0	Memory	2602	0	2 Byte
41	C_SE_TB_1	62	-	0	Memory	2604	0	2 Byte
42	C_SE_NC_1	70	-	0	Memory	2700	0	4 Byte
43	C_SE_NC_1	71	-	0	Memory	2704	0	4 Byte
44	C_SE_TC_1	72	-	0	Memory	2708	0	4 Byte

Declaring a database variable

The application object database is an array variable of type ST_IEC870_5_101AODBEntry [▶ 313]. Each array element corresponds to an application object. The maximum number of application objects is freely selectable and is only limited by the available memory. During PLC programming you must specify a constant maximum number. The maximum number of application objects cannot be changed at runtime.

In our example 50 application objects are declared. This number is sufficient for most applications. Please note that many application objects require adequate memory and runtime resources.

Define the following variable in MAIN:

```
PROGRAM MAIN
VAR
     AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
END_VAR
```

Configuring application objects

The object type (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 etc.), the object address and further object parameters are specified during configuration of the individual application objects.

The required application objects are configured during program runtime. Each application object (database array element) is configured by calling the F_iecInitAOEntry [▶ 280] function once. The array element to be configured is transferred to the function via VAR_IN_OUT. Configuration is usually carried out once during PLC program start-up via an Init routine. The F_iecInitAOEntry function expects the following function parameters (from left to right):

```
FUNCTION F_iecInitAOEntry : UDINT
VAR_INPUT
    eType           : E_IEC870_5_101TcTypeID;
    objAddr         : DWORD := 0;
    group           : DWORD := 0;
    multiplier      : BYTE := 0;
    ioMapType       : E_IEC870_5_101IOMappingType;
    byteOffs        : UDINT := 0;
    bitOffs         : UDINT := 0;
END_VAR
VAR_IN_OUT
    dbEntry         : ST_IEC870_5_101AODBEntry;
END_VAR
```

eType: application object type (ASDU identifier [▶ 327], e.g.: M_SP_NA_1 for single point or M_DP_NA_1 for double point). Please note that only the ASDU types listed in the compatibility list can be used. Invalid types are ignored.

objAddr : object address, e.g. 100. Each application object should be configured with a unique address.

group: group configuration parameters. The available group parameters are defined as constants and can be combined with an OR operator. E.g.: IEC870_GRP_INROGEN OR IEC870_GRP_PERCYC.

A description of all group configuration parameters [▶ 348] can be found here.

mutiplier: base time multiplier for cyclic/periodic data transfer (0=deactivated). The base time is configured via the system parameters. If the base time was set to T#10 s, and the multiplier to 2, for example, the periodic/cyclic data of the application object are sent every 20 seconds.

ioMapType: This parameter [▶ 331] defines from or in which process data area of the TwinCAT PLC the IEC process data are to be mapped at runtime (inputs, outputs, memory, data).

byteOffs: process data area byte offset;

bitOffs: process data area bit offset;

dbEntry: application object to be configured (a database variable array element that is transferred to the function via VAR_IN_OUT).

To configure the application objects during program start-up, add the following PLC code in MAIN:

```
PROGRAM MAIN
VAR
    AODB      : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
```

```
    init     : BOOL := TRUE;
    initError : UDINT;
END_VAR
```

```
IF init THEN
    init := FALSE;

  (* Monitored Single Points *)
 initError := F_iecInitAOEntry( M_SP_NA_1, 100, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 100, 0, AODB
[0] );
    initError := F_iecInitAOEntry( M_SP_NA_1, 101, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 100, 1, A
ODB[1] );
    initError := F_iecInitAOEntry( M_SP_TB_1, 102, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 100, 2, A
ODB[2] );
  (* Double Points*)
 initError := F_iecInitAOEntry( M_DP_NA_1, 200, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 200, 0, AODB
[3] );
    initError := F_iecInitAOEntry( M_DP_NA_1, 201, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 200, 2, A
ODB[4] );
    initError := F_iecInitAOEntry( M_DP_TB_1, 202, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 200, 4, A
ODB[5] );
  (* Regulating step value *)
 initError := F_iecInitAOEntry( M_ST_NA_1, 300, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 300, 0, AODB
[6] );
    initError := F_iecInitAOEntry( M_ST_NA_1, 301, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 301, 0, A
ODB[7] );
    initError := F_iecInitAOEntry( M_ST_TB_1, 302, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 302, 0, A
ODB[8] );
  (* 32 bit string*)
 initError := F_iecInitAOEntry( M_BO_NA_1, 400, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 400, 0, AODB
[9 );
    initError := F_iecInitAOEntry( M_BO_NA_1, 401, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 404, 0, A
ODB[10] );
    initError := F_iecInitAOEntry( M_BO_TB_1, 402, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 408, 0, A
ODB[11] );
  (* Measured value, normalized value *)
 initError := F_iecInitAOEntry( M_ME_NA_1, 500, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 500, 0, AODB
[12] );
    initError := F_iecInitAOEntry( M_ME_NA_1, 501, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 502, 0, A
ODB[13] );
    initError := F_iecInitAOEntry( M_ME_TD_1, 502, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 504, 0, A
ODB[14] );
  (* Mesured value, scaled value *)
 initError := F_iecInitAOEntry( M_ME_NB_1, 600, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 600, 0, AODB
[15] );
    initError := F_iecInitAOEntry( M_ME_NB_1, 601, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 602, 0, A
ODB[16] );
    initError := F_iecInitAOEntry( M_ME_TE_1, 602, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 604, 0, A
ODB[17] );
  (* Measured value , short floating point value *)
 initError := F_iecInitAOEntry( M_ME_NC_1, 700, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 700, 0, AODB
[18] );
    initError := F_iecInitAOEntry( M_ME_NC_1, 701, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 704, 0, A
ODB[19] );
    initError := F_iecInitAOEntry( M_ME_TF_1, 702, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 708, 0, A
ODB[20] );
  (* Integrated totals *)
 initError := F_iecInitAOEntry( M_IT_NA_1, 800, IEC870_GRP_REQCOGEN OR IEC870_GRP_LOCFREEZE, 0, MAP_
AREA_MEMORY, 800, 0, AODB[21] );
    initError := F_iecInitAOEntry( M_IT_NA_1, 801, IEC870_GRP_REQCOGEN OR IEC870_GRP_LOCFREEZE, 0, M
AP_AREA_MEMORY, 804, 0, AODB[22] );
    initError := F_iecInitAOEntry( M_IT_TB_1, 802, IEC870_GRP_REQCOGEN OR IEC870_GRP_LOCFREEZE, 0, M
AP_AREA_MEMORY, 808, 0, AODB[23] );

  (* Single commands *)
 initError := F_iecInitAOEntry( C_SC_NA_1, 10, 0, 0, MAP_AREA_MEMORY, 2100, 0, AODB[24] );
    initError := F_iecInitAOEntry( C_SC_NA_1, 11, 0, 0, MAP_AREA_MEMORY, 2100, 1, AODB[25] );
    initError := F_iecInitAOEntry( C_SC_TA_1, 12, 0, 0, MAP_AREA_MEMORY, 2100, 2, AODB[26] );
  (* Double commands *)
 initError := F_iecInitAOEntry( C_DC_NA_1, 20, 0, 0, MAP_AREA_MEMORY, 2200, 0, AODB[27] );
    initError := F_iecInitAOEntry( C_DC_NA_1, 21, 0, 0, MAP_AREA_MEMORY, 2200, 2, AODB[28] );
    initError := F_iecInitAOEntry( C_DC_TA_1, 22, 0, 0, MAP_AREA_MEMORY, 2200, 4, AODB[29] );
  (* Regulating step commands *)
 initError := F_iecInitAOEntry( C_RC_NA_1, 30, 0, 0, MAP_AREA_MEMORY, 2300, 0, AODB[30] );
    initError := F_iecInitAOEntry( C_RC_NA_1, 31, 0, 0, MAP_AREA_MEMORY, 2301, 0, AODB[31] );
    initError := F_iecInitAOEntry( C_RC_TA_1, 32, 0, 0, MAP_AREA_MEMORY, 2302, 0, AODB[32] );
  (* 32 bit string commands *)
 initError := F_iecInitAOEntry( C_BO_NA_1, 40, 0, 0, MAP_AREA_MEMORY, 2400, 0, AODB[33] );
    initError := F_iecInitAOEntry( C_BO_NA_1, 41, 0, 0, MAP_AREA_MEMORY, 2404, 0, AODB[34] );
```

```
    initError := F_iecInitAOEntry( C_BO_TA_1, 42, 0, 0, MAP_AREA_MEMORY, 2408, 0, AODB[35] );
  (* Set point, normalized values*)
 initError := F_iecInitAOEntry( C_SE_NA_1, 50, 0, 0, MAP_AREA_MEMORY, 2500, 0, AODB[36] );
    initError := F_iecInitAOEntry( C_SE_NA_1, 51, 0, 0, MAP_AREA_MEMORY, 2502, 0, AODB[37] );
    initError := F_iecInitAOEntry( C_SE_TA_1, 52, 0, 0, MAP_AREA_MEMORY, 2504, 0, AODB[38] );
  (* Set point, scaled valuess *)
 initError := F_iecInitAOEntry( C_SE_NB_1, 60, 0, 0, MAP_AREA_MEMORY, 2600, 0, AODB[39] );
    initError := F_iecInitAOEntry( C_SE_NB_1, 61, 0, 0, MAP_AREA_MEMORY, 2602, 0, AODB[40] );
    initError := F_iecInitAOEntry( C_SE_TB_1, 62, 0, 0, MAP_AREA_MEMORY, 2604, 0, AODB[41] );
  (* Set point, short floating point values *)
 initError := F_iecInitAOEntry( C_SE_NC_1, 70, 0, 0, MAP_AREA_MEMORY, 2700, 0, AODB[42] );
    initError := F_iecInitAOEntry( C_SE_NC_1, 71, 0, 0, MAP_AREA_MEMORY, 2704, 0, AODB[43] );
    initError := F_iecInitAOEntry( C_SE_TC_1, 72, 0, 0, MAP_AREA_MEMORY, 2708, 0, AODB[44] );

END_IF
```

7.1.4 Mapping of PLC and IEC process data

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The TwinCAT PLC process data are cyclically mapped (copied) into the IEC process data (application objects) and vice versa at program runtime. For mapping of the IEC<->PLC process data up to 4 process data areas (IO inputs, IO outputs, memory area, data area) can be declared as buffer variables in the PLC program. The byte size of the buffers is freely selectable and may be different for each area. Its not necessary to declare unused buffer areas.

In our introductory example we declare 4 PLC process data areas with 3000 bytes each:

```
PROGRAM MAIN
VAR
    AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;

    init        : BOOL := TRUE;
    initError   : UDINT;

     inputs AT%IB0  : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..2999] OF BYTE;
    memory AT%MB0   : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;

END_VAR
```

How the process data are to be mapped at runtime is specified during configuration of the application objects via the F_iecInitAOEntry [▶ 280] function.
See also in: Declaration and configuration of application objects [▶ 375].

Now the buffer variables are declared as byte arrays. To have better access to the desired data the variables are declared a second time and assigned to the corresponding byte/bit offset addresses.
Now a change within the byte array causes a change of the corresponding variable and vice versa. This is not imperative necessary. You can also have access directly to the bits/bytes of the byte array buffer variables.

```
VAR_GLOBAL
(* Memory offset 0..99 unused *)
  (* Single points *)
 msgSingle_0     AT%MX100.0 : BOOL;
    msgSingle_1     AT%MX100.1 : BOOL;
    msgSingle_2     AT%MX100.2 : BOOL;

  (* Double points *)
    (*      Bit 0..1 = first double point,
        Bit 2..3 = second double point,
        Bit 4..5 = third double point,
        Bit 6..7 = fourth double point *)
    msgDouble_0     AT%MB200    : BYTE;

  (* Regulating step values *)
    msgStep_0           AT%MB300    : BYTE;
    msgStep_1           AT%MB301    : BYTE;
    msgStep_2           AT%MB302    : BYTE;

  (* 32 bit strings *)
 msgBitStr_0     AT%MD400    : DWORD := 2#10001000_10001000_10001000_10001000;
```

```
    msgBitStr_1     AT%MD404    : DWORD := 2#10001000_10001000_10001000_10001000;
    msgBitStr_2     AT%MD408    : DWORD := 2#10001000_10001000_10001000_10001000;

  (* Measured values, normalized values *)
 msgNormalized_0     AT%MW500    : WORD;
    msgNormalized_1     AT%MW502    : WORD;
    msgNormalized_2     AT%MW504    : WORD;

  (* Mesured values, scaled values *)
 msgScaled_0     AT%MW600    : INT;
    msgScaled_1     AT%MW602    : INT;
    msgScaled_2     AT%MW604    : INT;

  (* Measured values, short floating point values *)
 msgFloating_0   AT%MD700    : REAL;
    msgFloating_1   AT%MD704    : REAL;
    msgFloating_2   AT%MD708    : REAL;

  (* Integrated totals *)
 msgTotal_0      AT%MD800    : UDINT;
    msgTotal_1      AT%MD804    : UDINT;
    msgTotal_2      AT%MD808    : UDINT;

  (*################################################################*)
    (* Single commands *)
 cmdSingle_0     AT%MX2100.0 : BOOL;
    cmdSingle_1     AT%MX2100.1 : BOOL;
    cmdSingle_2     AT%MX2100.2 : BOOL;

  (* Double commands *)
    (*      Bit 0..1 = first double command,
        Bit 2..3 = second double command,
        Bit 4..5 = third double command,
        Bit 6..7 = fourth double command *)
 cmdDouble_0     AT%MB2200       : BYTE;

  (* Regulating step commands *)
 cmdStep_0           AT%MB2300 : BYTE;
    cmdStep_1           AT%MB2301 : BYTE;
    cmdStep_2           AT%MB2302 : BYTE;

  (* 32 bit string commands *)
 cmdBitStr_0     AT%MD2400       : DWORD;
    cmdBitStr_1     AT%MD2404       : DWORD;
    cmdBitStr_2     AT%MD2408       : DWORD;

  (* Set point, normalized values *)
 cmdNormalized_0     AT%MW2500       : WORD;
    cmdNormalized_1     AT%MW2502       : WORD;
    cmdNormalized_2     AT%MW2504       : WORD;

  (* Set point, scaled values *)
 cmdScaled_0     AT%MW2600       : INT;
    cmdScaled_1     AT%MW2602       : INT;
    cmdScaled_2     AT%MW2604       : INT;

  (* Set point, short floating point values *)
 cmdFloating_0   AT%MD2700       : REAL;
    cmdFloating_1   AT%MD2704       : REAL;
    cmdFloating_2   AT%MD2708       : REAL;
END_VAR
```

Mapping of the IEC<->PLC process data in controlled station

Process data in monitoring direction (slave->master information)

Example 1

Single point information (M_SP_NA_1) with the IOA = 100, PLC memory area, byte offset = 100, bit offset = 0.

msgSingle_0 == memory[100].0 -> Controlled station FB -> ... -> Controlling station

Example 2

Measured value, short floating point value (M_ME_NC_1) with the IOA = 700, PLC memory area, byte offset = 700, bit offset = 0 (irrelevant).

msgFloating_0 == memory[700..703] -> Controlled station FB -> ... -> Controlling station

Process data in control direction (master->slave commands)

Example 1

Single command state (C_SC_NA_1) with the IOA = 10, PLC memory area, byte offset = 2100, bit offset = 0.

Controlling station -> ... -> Controlled station FB -> memory[2100].0 == cmdSingle_0

Example 2

Set point, short floating point value (C_SE_NC_1) with the IOA = 70, PLC memory area, byte offset = 2700, bit offset = 0 (irrelevant).

Controlling station -> ... -> Controlled station FB -> memory[2700..2703] == cmdFloating_0

7.1.5 Declaring and calling an instance of the IEC60870-5-101 substation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The complete functionality of a substation is encapsulated in the FB_IEC870_5_101Slave function block. An instance can be used to establish a connection to the master. Assign the connection handle *hSerial* from the fast task to the substation function block instance.

Add the following PLC code to the declaration part of *P_MAIN_LowSpeed* program:

```
PROGRAM P_MAIN_LowSpeed
VAR
    AODB        : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;

    init        : BOOL := TRUE;
    error       : UDINT;

    inputs AT%IB0   : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..2999] OF BYTE;
    memory AT%MB0   : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;


     bEnable         : BOOL := TRUE;
    server      : FB_IEC870_5_101Slave;
END_VAR
```

and the instance is called in the program part:

```
IF init THEN
    init := FALSE;
...

ELSE

...
 server(
     pInputs := ADR( inputs ),
     cbInputs := SIZEOF( inputs ),
     pOutputs := ADR( outputs ),
     cbOutputs := SIZEOF( outputs ),
     pMemory := ADR( memory ),
     cbMemory := SIZEOF( memory ),
     pData := ADR( data ),
     cbData := SIZEOF( data ),
     pAOEntries := ADR( AODB ),
     cbAOEntries := SIZEOF( AODB ),
     hSerial := P_SerialComm_HighSpeed.hSerial, (* serial link interface connection handle from fast
task *)
```

```
    bEnable := bEnable );
...

END_IF
```

7.1.6 IEC60870-5-101 protocol parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The behaviour of the substation can be adapted to the requirements of the master via the IEC60870-5-101 protocol parameters. Most parameters have preallocated default values that do not have to be changed.

In our example we set the link address and link address octet size:

```
IF init THEN
    init := FALSE;
...

    server.protPara.linkAddr := 220;(* slave link address *)
    server.protPara.eLinkAddrSize := eIEC870_LinkAddr_TwoOctets; (* link address octet size *)
...

ELSE
    server( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all transfer protocol parameters can be found here: ST_IEC870_5_101PotocolParams [▶ 407].

7.1.7 System parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The common ASDU address and the user functions of the substation are configured via the system parameters.

In our introduction we configure the following system parameters:

- The common ASDU address is set to 7. (*asduAddr*)
- During initialisation the system time of the substation is synchronised with the system time of the local TwinCAT PC (*bUsePCTime*).
- Synchronisation of the substation system time via the time synchronisation command is activated (*bSyncTime*).
- The system time of the TwinCAT PC (*bSyncPCTime*) should not be synchronised during synchronisation of the substation system time.
- Sending of M_EI_NA_1 (End of init) to the central station is activated (*bEndOfInit*).
- Sending of the periodic/cyclic data is deactivated (*bPerCyclic*). The base time for sending of these data is set to 5 s.
- Background scan is deactivated (*bBackScan*). The background scan cycle time is set to 30 s.
- Local freeze and resetting of the counter readings is activated (*bPerFRZ*), and the cycle time for freeze and resetting is set to 15 s.
- Logging of debugging messages in the application log is activated (*dbgMode*). Logging of device state changes is enabled.

Add the following PLC code to your PLC project:

```
IF init THEN
    init := FALSE;
...


    server.sysPara.asduAddr := 7;
    server.sysPara.bUsePCTime := TRUE;
    server.sysPara.bSyncTime := TRUE;
    server.sysPara.bSyncPCTime := FALSE;
    server.sysPara.bEndOfInit := TRUE;
    server.sysPara.bPerCyclic := FALSE;
    server.sysPara.tPerCyclicBase := T#5s;
    server.sysPara.bBackScan := FALSE;
    server.sysPara.tBackScanCycle := T#30s;
    server.sysPara.bPerFRZ := TRUE;
    server.sysPara.tPerFRZCycle := T#15s;
    server.sysPara.dbgMode := (*IEC870_DEBUGMODE_ASDU OR*) IEC870_DEBUGMODE_DEVSTATE;

...

ELSE
    server( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all system parameters can be found here: ST_IEC870_5_101SystemParams [▶ 317].

7.1.8 Station interrogation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The station interrogation command is initiated by the central station. The ID field of the command also contains the group (1 to 16 or general). The substation transfers the application objects associated with this group to the central station with cause of transmission *<20>* to *<36>*. Application objects with time tags are transferred without time tags.

Configuration of the system parameters:

- It's not necessery to set additional system parameters.

Configuration of the application objects:

- The application must assign the data point to one or more than one interrogation group. Here you can find all available group parameters: Group configuration flags [▶ 348].

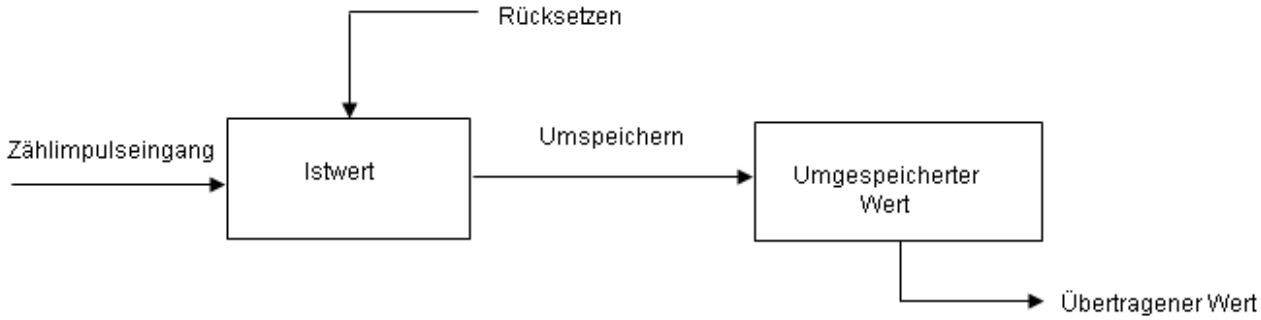
Example of data point configuration. The data point belongs to group: 1 and group: general.

```
F_iecInitAOEntry( M_SP_NA_1, 100, IEC870_GRP_INROGEN OR IEC870_GRP_INRO1, 0, MAP_AREA_MEMORY, 100, 0
, AODB[0] );
```

7.1.9 Transfer of integrated totals (counter interrogation)

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

General count transfer model:



The actual values are added by counters. Via a re-store command that is either received by the central station or generated locally (in the substation), the actual values can be re-stored (copied) periodically into re-stored values. After freezing, the recorded value is either reset to zero (logging of incremental values), or the counter continues adding up (logging of counter readings).

Application objects with counts are assigned to groups. The groups are frozen individually, reset, or transferred. The central station sends count query commands to the substation. The task to be carried out (FRZ) and the group (RQT) are specified in an ID field of the command (QCC).

The allocation of the application objects to the individual groups (1 to 4 or general) is specified via the group flag parameter during configuration. There are four operating modes for recording counter readings and incremental values. Each mode includes notes about the configuration of the system parameters or the application objects.

Mode A: local freeze with spontaneous transfer

The substation internally initiates freeze or freeze with reset. The frozen counts are transferred spontaneously, once the freeze or freeze with reset function was executed. In this mode the central station does not issue any count query commands.

Configuration of the system parameters:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

The first parameter activates local freeze or freeze with reset. The second parameter specifies the cycle time with which freeze or freeze with reset is carried out (e.g. every 60 seconds).

Configuration of the application objects:

- The IEC870_GRP_SPONTOFF group parameter must not be set, since it would prevent spontaneous data transfer of the counts.
- The count is frozen if the IEC870_GRP_LOCFREEZE group parameter was set.
- The count is reset if the IEC870_GRP_LOCRESET group parameter was set.
- Local freeze or freeze with reset is carried out simultaneously for all groups (1 to 4 or general).

Operating mode B: Local freeze with counter interrogation

The substation internally initiates freeze or freeze with reset. The central station queries the frozen counts via count query commands. In this case the central station must not use freeze or freeze with reset in the command ID field (FRZ=0). The counts are queried generally or in groups 1 to 4.

Configuration of the system parameters:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

The first parameter activates local freeze or (and) reset. The second parameter specifies the cycle time with which freeze or freeze with reset is carried out (e.g. every 60 seconds).

Configuration of the application objects:

- The IEC870_GRP_SPONTOFF group parameter must be set. The counts are not to be transferred spontaneously to the central station.

- The count is frozen if the IEC870_GRP_LOCFREEZE group parameter was set.
- The count is reset if the IEC870_GRP_LOCRESET group parameter was set.
- Local freeze or freeze with reset is carried out simultaneously for all groups (1 to 4 or general).

Operating mode C: The central station initiates freeze, freeze with reset, or reset

The central station periodically issues a count query command to the substation for controlling the freeze or (and) reset process. This command does not result in a count transfer. The central station sends a subsequent count query command for collecting the frozen counts. This is similar to operating mode B.

Configuration of the system parameters:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Local freeze or (and) reset must be deactivated. The second parameter is ignored.

Configuration of the application objects:

- IEC870_GRP_SPONTOFF must be set. The counts are not to be transferred spontaneously to the central station.
- The IEC870_GRP_LOCFREEZE and IEC870_GRP_LOCRESET group parameters must not be set. The central station initiates freeze or (and) reset.
- The counts can be assigned to individual groups (1 to 4 or general) and queried (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1, IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

Operating mode D: The central station initiates freeze and (or) reset, and the frozen values are transferred spontaneously

This mode is a combination of the count command from the central station as in mode C and spontaneous transfer of the counts as in mode A.

Configuration of the system parameters:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Local freeze or (and) reset must be deactivated. The second parameter is ignored.

Configuration of the application objects:

- The IEC870_GRP_SPONTOFF group parameter must not be set, since it would prevent spontaneous data transfer of the counts.
- The IEC870_GRP_LOCFREEZE and IEC870_GRP_LOCRESET group parameters must not be set. The central station initiates freeze or (and) reset.
- The counts can be assigned to individual groups (1 to 4 or general) and queried (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1, IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

7.1.10 Clock (time) synchronisation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The way in which the system time of the substation is to be synchronised can be configured via the system parameters.

- During initialisation the system time of the substation can be synchronised with the system time of the local TwinCAT PC;
- When a time synchronisation command is received from the central station, the system time of the substation can also be synchronised;

- The system time of the local TwinCAT PC can also be synchronised when a time synchronisation command is received.

7.1.11 Background scan

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The background scan is used for refreshing the process information sent from the substation to the central station as an additional safety contribution to the station interrogation and for spontaneous transfers. Application objects with the same type IDs as for the station interrogation may be transferred continuously with low priority, and with <2> background scan as the cause of transmission. The valid ASDU type IDs are listed in the compatibility list for the station (table type ID <-> cause of transmission). The background scan is initiated by the substation and is independent of the station interrogation commands.

Configuration of the system parameters:

The transfer cycle is specified via system parameters [▶ 317] in the substation.

```
bBackScan := TRUE;
tBackScanCycle := T#30s;
```

Configuration of the application objects:

Application objects whose process data are to be transferred as a background scan have to be configured via the IEC870_GRP_BACKGROUND group flag.

Example:

```
F_iecInitAOEntry( M_SP_NA_1, 100, IEC870_GRP_INROGEN OR IEC870_GRP_BACKGROUND, 0, MAP_AREA_MEMORY, 1
00, 0, AODB[0] );
```

7.1.12 Cyclic data transmission

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

Cyclic data transfer is initiated in a similar way as the background scan from the substation. It is independent of other commands from the central station. Cyclic data transfer continuously refreshes the process data of the central station. The process data are usually measured values that are recorded at regular intervals. Cyclic data transfer is often used for monitoring non-time-critical or relatively slowly changing process data (e.g. temperature sensor data). Cyclic/periodic data are transferred to the central station with cause of transmission *<1> periodic/cyclic*. The valid ASDU type IDs are listed in the compatibility list for the station (table type ID <-> cause of transmission). Cyclic data transfer can be configured via the system parameters and the configuration parameters of the application objects.

Configuration of the system parameters:

```
bPerCyclic      : BOOL := TRUE;
tPerCyclicBase  : TIME := T#60s;
```

The first parameter activates cyclic transfer. The second parameter is the base time for the cyclic/periodic data transfer (in this case 60 seconds).

Configuration of the application objects:

- IEC870_GRP_PERCYC group parameter has to be set;
- The multiplier parameter *(multiplier)* of the F_iecInitAOEntry function has to be set to a zero <> value. Example: With a multiplier = 2 and a base time of 60 seconds the process data of the application object are sent to the central station every 120 seconds.

Example of measured value configuration. The value must be transferred every 120 seconds to the central station (measured value, normalized value without time tag, M_ME_NA_1).

```
F_iecInitAOEntry( M_ME_NA_1, 222, IEC870_GRP_INROGEN OR IEC870_GRP_PERCYC, 2, MAP_AREA_MEMORY, 6, 0,
 AODB[2] );
```

7.1.13 Command transmission

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

Commands can be sent from the central station in control direction (to the substation). A single command with type ID 45 (C_SC_NA_1) is used for controlling an application object that is transferred in monitoring direction as a single-point information (M_SP_NA_1, M_SP_TA_1 or M_SP_TB_1). A dual command (C_DC_NA_1) is used for controlling an application object that is transferred in monitoring direction as a double-point information (M_DP_NA1, M_DP_TA_1 or M_DP_TB_1), etc.

Configuration of system parameters:

- It's not necessery to set additional system parameters.

Configuration of application objects:

- Configure the application object type as process data in control direction.
- The information object addresses (IOA's) should be equal to the addresses configured in control station (master);

Examples:

Single command with the IOA = 10. The command value is mapped to memory area buffer, byte offset = 100, bit offset = 0.

```
F_iecInitAOEntry( C_SC_NA_1, 10, 0, 0, MAP_AREA_MEMORY, 100, 0, AODB[24] );
```

Double command with the IOA = 20. The command value is mapped to memory area buffer, byte offset = 200, bit offset = 0..1.

```
F_iecInitAOEntry( C_DC_NA_1, 20, 0, 0, MAP_AREA_MEMORY, 200, 0, AODB[27] );
```

Set point, scaled value with the IOA = 60. The command value is mapped to memory area buffer, byte offset = 600..601, bit offset = 0.

```
F_iecInitAOEntry( C_SE_NB_1, 60, 0, 0, MAP_AREA_MEMORY, 600, 0, AODB[39] )
```

7.1.14 Interrogation / Read command

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The central station sends interrogation commands to the substation. The interrogation command contains the address of the application object to be interrogated. The data of this application object are to be sent to the central station. The substation sends the data with cause of transmission *<5> interrogation or interrogated*. The valid ASDU type IDs are listed in the compatibility list for the station (table type ID <-> cause of transmission).

Configuration of system parameters:

- It's not necessary to set additional system parameters.

Configuration of application objects:

- It's not necessary to set additional system parameters.

7.1.15 Double transmission

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

All application objects (information objects) that are transferred with cause of transmission <*3*> *spontaneous* may be transferred twice, with or without time tag. This mode is referred to as "double transmission". Double transmission is currently not supported by the substation.

7.1.16 Quality Flags

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The quality flags (quality descriptor) provide additional information for the central station on the quality of an application object. The quality flags can be set/reset independent of each other from the PLC application via the F_iecSetAOQuality [▶ 282] function. The F_iecGetAOQuality [▶ 283] function can be used to interrogate the state of the quality flags. Any change in the quality flags leads to a spontaneous transfer of the data to the central station.

The following quality flags are internally analyzed by the substation at runtime:

- IECQ_BL_ON (blocked). If the process data of the application object were blocked for the transfer, mapping of the PLC and IEC process data is not executed for this application.

The following quality flags are internally set/reset by the substation at runtime:

- IECQ_IV_ON (Invalid). The substation sets the invalid flag if mapping of the PLC and IEC process data could not be carried out (e.g. due to faulty configuration of the application object). This behavior can be deactivated by setting group parameter IEC870_GRP_IV_OFF.

All other quality flags are sent to the central station without change.

7.1.17 Testing the communication

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

You can simulate the data point value changes. Setting the *bChangeIO* variable to TRUE enables the simulation and to FALSE disables. The new values are send cyclicaly every 3 seconds to the control station.

```
PROGRAM MAIN
VAR
    ...

    bChangeIO : BOOL; (* TRUE => simulate/modify plc process data *)
 timer : TON;
    i : INT;

    ...
END_VAR
```

```
...


(*modify plc process data *)
timer( IN := bChangeIO, PT := T#3s );
IF timer.Q THEN
    timer( IN := FALSE );

    msgSingle_0 := NOT msgSingle_0;
    msgSingle_1 := NOT msgSingle_1;
    msgSingle_2 := NOT msgSingle_2;

    FOR i:= 0 TO 3 DO
        IF F_iecGetDPI(msgDouble_0, i) = eIEC870_DPI_ON THEN (* the value of double point allready O
N? *)
            msgDouble_0 := F_iecSetDPI(     msgDouble_0, i, eIEC870_DPI_OFF ); (* change ON => OFF *
)
        ELSE
            msgDouble_0 := F_iecSetDPI(     msgDouble_0, i, eIEC870_DPI_ON );(* change OFF => ON *)
        END_IF
    END_FOR

    F_iecIncVTI( msgStep_0 );
    F_iecDecVTI( msgStep_1 );

    msgBitStr_0 := ROL( msgBitStr_0, 1 );
    msgBitStr_1 := ROR( msgBitStr_1, 1 );

    msgNormalized_0 := msgNormalized_0 + 1;
    msgNormalized_1 := msgNormalized_1 + 2;

    msgScaled_0 := msgScaled_0 + 3;
    msgScaled_1 := msgScaled_1 - 3;

    msgFloating_0 := msgFloating_0 + 0.1;
    msgFloating_1 := msgFloating_1 + 1.5;

    msgTotal_0 := msgTotal_0 + 1;
    msgTotal_1 := msgTotal_1 + 2;
END_IF


...
```

7.1.18 Protocol and data transmission errors

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

The station error messages are placed into a FIFO. Up to 10 error messages can be stored. At fatal communication errors (e.g. error of the linking layer, the check sum of the frame doesn't fit) the connection is cut and has to be build on new. Errors within the application layer (e.g. the ADSU send buffer is overflowed by to many frames) are only logged and doesn't cut the connection.

For this errors it is also possible to cut the connection out of the application. In addition to the error code also the error source is notified in the error message. This simplifies the localization of an error.

Example

The occurring error messages of an IEC 60870-5-101 sub station can be read out via the request:

```
PROGRAM MAIN
VAR
...

    server : FB_IEC870_5_101Slave;
...

END_VAR
```

```
...

REPEAT
    server.system.device.errors.RemoveError();
```

```
    IF server.system.device.errors.bOk  THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'IEC60870-5-101 slave error: 0x%s',
            DWORD_TO_HEXSTR( server.system.device.errors.getError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT server.system.device.errors.bOk
END_REPEAT

...
```

7.2 Quick Start

Simple projects including complete sources is to be found here: IEC60870-5-101 substation [▶ 396].

Interoperability list is to be found here: Interoperability check list

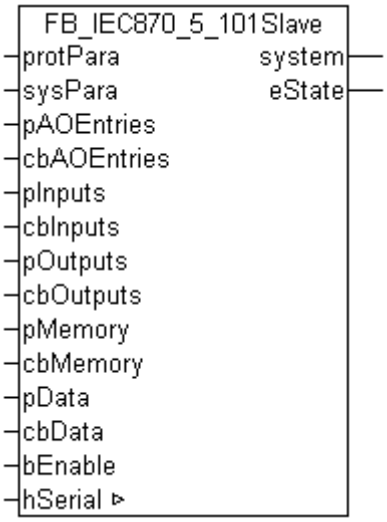
Overview of error codes is to be found here: Error codes

A detailed instruction about implementing the substation within the PLC is to be found here: TUTORIAL [▶ 373]

Brief instruction

1. Create a new PLC project and link the PLC library **TcIEC870_5_101Slave.Lib**.
2. Apply two PLC tasks, one fast (e.g. with T#1ms cycle time) and one slow (e.g. with T#1ms cycle time). Apply two function blocks (e.g. *P_SerialComm_HighSpeed* and *P_MAIN_LowSpeed*). *P_SerialComm_HighSpeed* is called by the fast and *P_MAIN_LowSpeed* is called by the slow task.
3. Apply an instance of the function block FB_IEC870_SerialLineCtrl [▶ 404] at *P_SerialComm_HighSpeed*, configure it and call it.
   In case of communication via Beckhoff's serial bus terminals or the serial interface of the PC, apply the buffers *KL6inData5B, KL6outData5B* or *PcComInData, PcComOutData* and link with the according IO process data at the TwinCAT System Manager.
4. The instance of the T_HSERIALCTRL [▶ 415] variable is used for exchanging Tx/Rx telegrams between both tasks (programs). Apply this variable for example as a global variable.
5. Configuration of data points: Apply an array variable of ST_IEC870_5_101AODBEntry [▶ 313] type. Every array element corresponds to a data point. Configure the data points via the function F_iecInitAOEntry [▶ 280] at runtime (e.g. at an init step).
6. Apply an instance of the des protocol block FB_IEC870_5_101Slave [▶ 392] at the *P_MAIN_LowSpeed*, configure it and call it.
7. The system and protocol parameter are to be configured according to the parameter of the control station.

7.2.1 FB_IEC870_5_101Slave



The IEC60870-5-101 substation (Slave) can be implemeted in TwinCAT PLC with the instance of FB_IEC870_5_101Slave.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial         : T_HSERIALCTRL;
END_VAR
```

hSerial : Connection handle [▶ 415] to FB_IEC870_SerialLineCtrl [▶ 404]. Via this variable the data to be send and received are exchanged with FB_IEC870_SerialLineCtrl.

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_101ProtocolParams;  (* IEC60870-5-101 serial link protocol communicati
on params *)
    sysPara     : ST_IEC870_5_101SystemParams;  (* IEC60870-5-101 slave system params *)
    pAOEntries  : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF ST_IEC870_5_101AODBEntry;  (* Pointer
 to the first element of application database object array *)
    cbAOEntries : UDINT;  (* Byte size (length) of application database object array *)
    pInputs     : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbInputs    : UDINT;
    pOutputs    : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbOutputs   : UDINT;
    pMemory     : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbMemory    : UDINT;
    pData       : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbData      : UDINT;
    bEnable     : BOOL := TRUE;
END_VAR
```

protPara: IEC60870-5-101-protocol parameter [▶ 407].

sysPara: System parameter [▶ 317].

pAOEntries: Address of application database object [▶ 313] array..

cbAOEntries: Byte size of application database object array.

pInputs: Address of PLC process data range of the inputs.

cbInputs: Byte size of PLC process data range of the inputs

pOutputs: Address of PLC process data range of the outputs.

cbOutputs: Byte size of PLC process data range of the outputs.

pMemory: Address of PLC process data range of the flags

cbMemory: Byte size of PLC process data range of the.

pData: Address of PLC data range .

cbData: Byte size of PLC data range.

bEnable : Enables/Disables the function block (Communication and Connection).

VAR_OUTPUT

```
VAR_OUTPUT
    system      : ST_IEC870_5_101SystemInterface;
    eState      : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED; (* Serial link connecti
on state *)
END_VAR
```

system: System interface [▶ 395]. This variable is used for communication interface to IEC device (here: Substation).

- Member variable *system.device* is expected e.g. from function F_iecSetAOQuality [▶ 282] as parameter VAR_IN_OUT.
- Member variable *system.device.errors* is a device error Fifo. The registered errors can be read from thePLC application.

eState: Connetion status [▶ 413] to the master.

Example:

Projects: IEC60870-5-101 controlled station [▶ 396]

Call in ST:

```
PROGRAM test
VAR
    slave1AODB      : ARRAY[1..50] OF ST_IEC870_5_101AODBEntry;

    inputs AT%IB0   : ARRAY[0..999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..999] OF BYTE;
    memory AT%MB0   : ARRAY[0..999] OF BYTE;
    data        : ARRAY[0..999] OF BYTE;

    server1         : FB_IEC870_5_101Slave;

    bEnable         : BOOL;
    eState      : E_IEC870_5_101SerialLinkState;

    bError      : BOOL;
    iecError    : ST_IEC870_5_101ErrorFifoEntry;
END_VAR
```

```
server1.protPara.linkAddr := 220;
server1.protPara.eLinkMode := eIEC870_LinkMode_Unbalanced;
server1.protPara.elinkAddrSize := eIEC870_LinkAddr_TwoOctets;

server1.sysPara.asduFmt.eAsduAddrSize := eIEC870_AsduAddr_TwoOctets;
server1.sysPara.asduFmt.eObjAddrSize := eIEC870_ObjAddr_ThreeOctets;
server1.sysPara.asduFmt.eCOTSize := eIEC870_COT_TwoOctets;
server1.sysPara.asduAddr := 7;
server1.sysPara.bUsePCTime := TRUE;
server1.sysPara.bSyncTime := TRUE;
server1.sysPara.bSyncPCTime := FALSE;
server1.sysPara.bEndOfInit := TRUE;
server1.sysPara.bPerCyclic := FALSE;
server1.sysPara.tPerCyclicBase := T#5s;
server1.sysPara.bBackScan := FALSE;
server1.sysPara.tBackScanCycle := T#30s;
server1.sysPara.bPerFRZ := TRUE;
server1.sysPara.tPerFRZCycle := T#15s;
server1.sysPara.dbgMode := IEC870_DEBUGMODE_LINKLAYER;
(* OR IEC870_DEBUGMODE_DEVSTATE OR IEC870_DEBUGMODE_ASDU;*)
server1.sysPara.bTimeCOT3 := FALSE;
```

```
server1( pInputs := ADR( inputs ),
    cbInputs := SIZEOF( inputs ),
    pOutputs := ADR( outputs ),
    cbOutputs := SIZEOF( outputs ),
    pMemory := ADR( memory ),
    cbMemory := SIZEOF( memory ),
    pData := ADR( data ),
    cbData := SIZEOF( data ),
    pAOEntries := ADR( slave1AODB ),
    cbAOEntries := SIZEOF( slave1AODB ),
    hSerial := P_SerialComm_HighSpeed.hSerial,
    bEnable := bEnable,
    eState=>eState );
```

The device error Fifo is read cyclically and the registered errors are written in the Application Log.

```
REPEAT
    server1.system.device.errors.RemoveError( getError=>iecError, bOk=>bError );
    IF bError THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG, 'IEC60870-5-101 slave error: 0x%s', D
WORD_TO_HEXSTR( iecError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT bError
END_REPEAT
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Slave.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101Link.Lib; TcIEC870_5_101.Lib; COMLibV2.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

7.2.2 F_GetVersionTcIEC870_5_101Slave

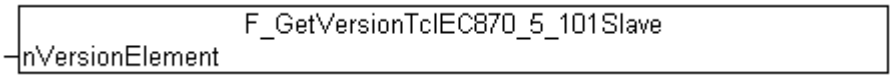


Fig. 8: F_GetVersionTcIEC870_5_101Slave

This function reads version information from the PLC library.

FUNCTION F_GetVersionTcIEC870_5_101Slave: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Slave.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101Link.Lib; TcIEC870_5_101.Lib; COMLibV2.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

7.2.3 ST_IEC870_5_101SystemInterface

```
TYPE ST_IEC870_5_101SystemInterface :
STRUCT
    device  : ST_IEC870_5_101DeviceInterface;
    service     : ST_IEC870_5_101SlaveServices;
END_STRUCT
END_TYPE
```

device: Communication interface [▶ 319] of IEC device. This variable is used for communication interface to IEC device (here: Substation).

service: IEC device service;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Slave.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101Link.Lib; TcIEC870_5_101.Lib; COMLibV2.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

7.3 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.).

2. Compare the interoperability check list of controlled station and control station.

3. Check the IO configuration and mapping of PLC variables in TwinCAT System Manager (configuration of serial interface, baudrate, parity, stoppbits settings). Compare them to the parameters set in the controlling station.

4. Check whether the function block issues an error code. The documentation for the error codes can be found here: Overview of error codes.

5. Check the protocol parameters [▶ 407] that are transferred to the function block (link address, link address octet size, FRAMELength etc.). Compare them to the parameters set in the controlling station.

6. Check the system parameter [▶ 317] that are transferred to the function block (ASDU address, ASDU address octet size, information object address octet size, cause of transfer octet size, etc.). Compare them to the the parameters set in the controlling station.

7. Check the configured data points (type, information object address etc.).

8. Check if the controlling station issues an error code.

9. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings.

7.4 Examples

Used controlled station configuration parameters:

- Standard PC interface: **COM1**
- **19200 Baud**
- Link address: **220**
- Link address size: **2 octets**
- Cause of transfer size: **2 octets**
- Originator address: **1**
- Common ASDU address: **7**
- Common ASDU address size: **2 octets**
- Information object address size: **3 octets**

Requirements

PLC Project	TwinCAT System Manager Configuration	Description	Download
MiniSlaveSample.pro	MiniSlaveSample.tsm	Simple IEC60870-5-101 controlled station. Very **small** PLC project with one single point in monitoring (IOA = 100) and one single command in control direction (IOA = 10).	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699281419/.zip
TutorialSample.pro	TutorialSample.tsm	Simple IEC60870-5-101 controlled station.	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11699280011/.zip

8 TcIEC870_5_101Link: IEC 60870-5-101 Serial Link Interface (master/slave)

The TwinCAT PLC library **TcIEC870_5_101Link.Lib** implements the transmission preocedures for the transport of the ASDUs via the serial interfaces of the pc and the Beckhoff KL6xxx/EL6xxx bus terminals.

'Low level' interface: IEC 60870-5-101 Serial Link Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/ Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

This interface is recommended if you:

- are familiar with the protocol standard;
- are implementing a protocol converter application;
- are implementing virtually all available standard functions in the application;
- are using special functions, such as the relaying of the time stamp from a Modbus device or the gaining of control over the command execution;
- require functions that are not supported according to the compatibility list;
- have many data points (>1000) and need high performance;

This interface is placed inside the protocol structure above the link layer (2) and implements necessary transport functions and frame formats. Application functions like for example the general interrogation, time synchronisation or counter interrogation are not implemented in the interface, but the user can implement them by himself.

Protocol structure of endsystem:

Requirements

Selection of application functions from IEC 60870-5-5	Application process
Selection of ASDU (service data unit of application layer) from	Application layer (7)
IEC 60870-5-3, IEC 60870-5-4 and IEC 60870-5-101	
TwinCAT PLC Library: IEC 60870-5-101 Serial Link (low level) Transport Interface	
N/A	Presentation layer (6)
	Session layer (5)
	Transport layer (4)
	Network layer (3)
unbalanced / balanced IEC 60870-5-2 IEC 60870-5-1 (FT 1.2)	Link layer (2)
EIA RS485, RS232 (V.24), Fibre Optics	Physical layer (1)

Remark: The layers 3..6 are not used.

8.1 Introduction

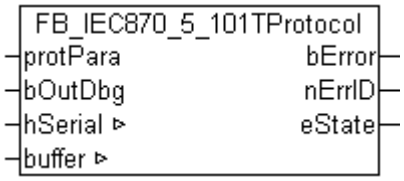
Simple projects with complete sources can be found under: Serial-Link-Interface-Examples.Serial-Link-Interface-Examples [▶ 415]

A PLC application which should communicate with a substation or control station via the transport interface needs the following resources:

- Instance of communication function block: FB_IEC870_5_101TProtocol [▶ 398];
- Instance of TX/RX data buffer: ST_IEC870_5_101TBuffer [▶ 37];
- Instance of function block to manipulate the TX/RX data buffer: FB_IEC870_5_101TBufferCtrl [▶ 37];

8.1.1 FB_IEC870_5_101TProtocol

From product version: TwinCAT PLC Library IEC60870-5-101 Substation v2.0.2 and higher



The FB_IEC870_5_101TProtocol communication block implements the transmission procedures of the connection layer according to IEC 60870-5-1 and IEC 60870-5-2 standard. In the event of a protocol error an associated error code is issued at the function block output and the data transfer is interrupted. The data exchange can be reactivated by calling the INIT task. E.g. the TX/RX data buffers are reset. The communication block expects a TX/RX data buffer variable and a connection handle to FB_IEC870_SerialLineCtrl. These variablen must be transferred to the block via VAR_IN_OUT.

The function block features the following tasks:

- **INIT** (initializes the function block);

Protocol configuration

The communication block has a structured protPara variable. Protocol parameters such as timeout times or connection addresses etc. can be configured via this variable

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial     : T_HSERIALCTRL;           (* Serial link control handle *)
    buffer      : ST_IEC870_5_101TBuffer; (* TX/RX data buffer *)
END_VAR
```

hSerial: Connection handle [▶ 415] to FB_IEC870_SerialLineCtrl [▶ 404].. The data to be sent and received are excahged with the FB_IEC870_SerialLineCtrl function block.

buffer: TX/RX data buffer.

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_101ProtocolParams; (* IEC60870-5-101 protocol parameters *)
    bOutDbg     : BOOL;       (* Enable/disable debug output *)
END_VAR
```

protPara: IEC60870-5-101protocol parameters [▶ 407]

bOutDbg:: Activates/deactivates the debug output of the frames in the TwinCAT System Manager logger view.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED; (* Data transfer state
*)
ND_VAR
```

bError: This output is switched to TRUE if an error occurs during data transfer.

nErrID: Returns an error code if the bError output is set;

eState: status [▶ 413] of the data exchange to the master

Example:

Sample projects: IEC60870-5-101 Serial Link Interface [▶ 415]

Requirements

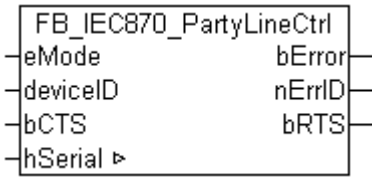
Development Environment	Target System Type	PLC Libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)

Also see about this

- M_SP_NA_1 [▶ 37]

8.1.2 FB_IEC870_PartyLineCtrl

From product version: TwinCAT PLC Library IEC870-5-101 Substation v2.0.2 and higher.



The function block FB_IEC870_PartyLineCtrl handles the data exchange to the controlled station in partyline mode.

The function block must be called cyclically in the PLC task. The *eMode* input activates/deactivates the partyline mode.Partyline mode is not used if you do not use the function block (default setting). The deviceID input variable must match the device ID from the TwinCAT System Manager IO configuration (general COM port tab). It is only required if you use the serial PC interface for communication in partyline mode.
The hSerial variable is a structure and is used for internal data exchange between the fast and slow communication task. Every time the IEC slave wants to transmit, the bRTS output (request to send) is first set to TRUE. Once transmit mode has been switched on the IEC slave is notified by setting the bCTS input (clear to send) to TRUE. The IEC slave then starts transmitting. Once the data have been sent (internal hardware buffers are empty) the IEC slave resets the bRTS output to FALSE. The send line can now be released for another device. Once this has occurred the IEC slave must be notified by setting the bCTS input to FALSE. In other words, the state of the bCTS input always follows the state of the bRTS output.

In the serial PC interface the bRTS output is only set to FALSE (data sent) when the ADS query of the internal hardware send buffer returns zero bytes in the buffer.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial         : T_HSERIALCTRL;
ND_VAR
```

hSerial: Connection handle [▶ 415] to FB_IEC870_SerialLineCtrl [▶ 404]. Via this variable the data to be sent and received are exchanged with FB_IEC870_SerialLineCtrl.

VAR_INPUT

```
VAR_INPUT
    eMode       : E_IEC870_5_101PartylineMode := eIEC870_PartylineMode_Off; (* Partyline modes (On/
Off)*)
    deviceID    : UDINT := 0; (*Used by SERIALLINEMODE_PC_COM_PORT only. DeviceId specifies the devi
ce on which the function is to be executed.
    The device Ids are specified by the TwinCAT System Manager during the hardware configuration.*)
    bCTS        : BOOL := FALSE; (* Clear to send *)
END_VAR
```

eMode : Partyline activation mode [▶ 414].

deviceID : TwinCAT System Manager device ID. This parameter is only needed for communication via the serial PC interface. If activation mode is set to *eIEC870_PartylineMode_Ext_On* the *deviceID* parameter is not used.

bCTS : Clear to send (for link layer).

VAR_OUTPUT

```
VAR_IN_OUT
    bError      : BOOL;
    nErrID      : UDINT;
    bRTS        : BOOL := FALSE; (* Request to send *)
ND_VAR
```

bError : This output is switched to TRUE if an error occurs.

nErrID: Returns an error code if the bError output is set.

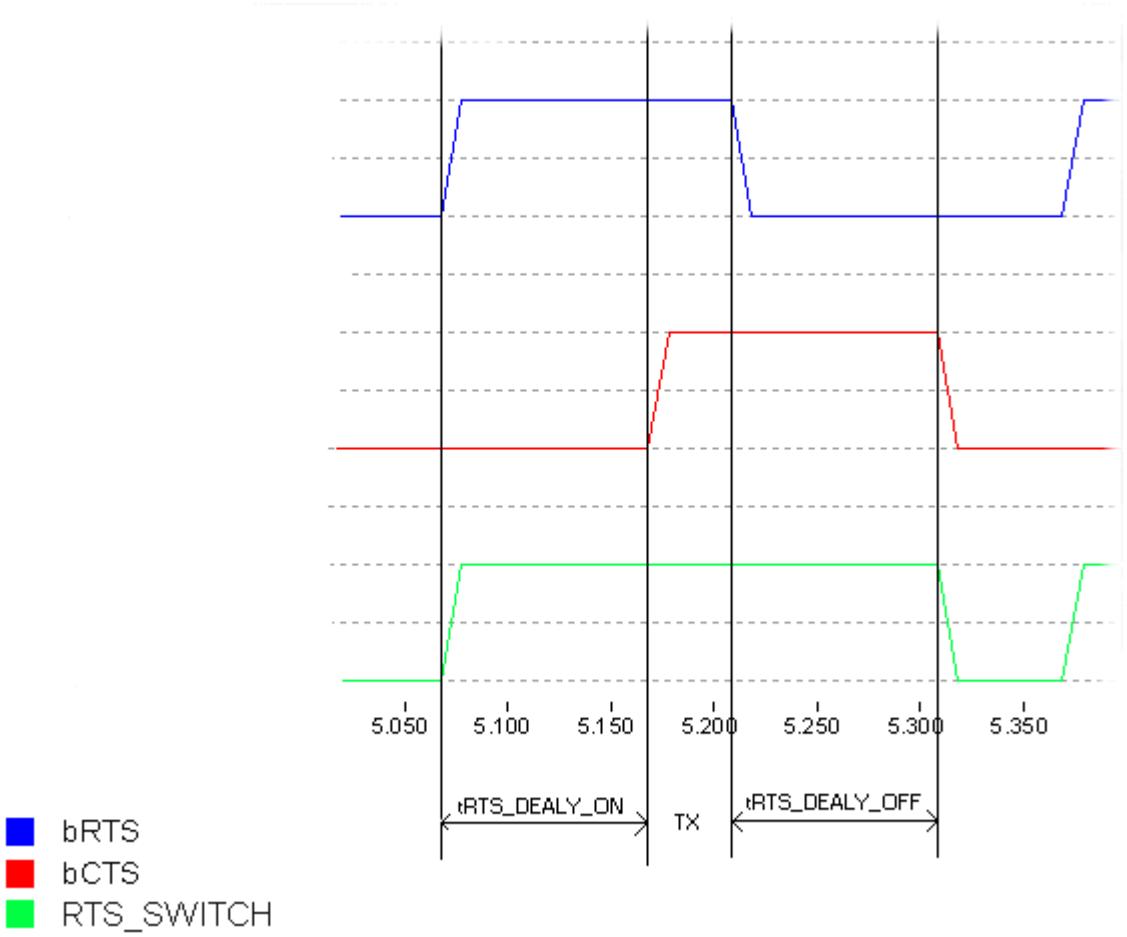
bRTS: Request to send (from link layer).

Example for TwinCAT v2.10 Build <1313 and older (CE image < v2.16 or older):

Implementation of partyline in the fast communication task: The connection for operation mode is switched ON / OFF via the *RTS_SWITCH* variable.

The *tRTS_DEALY_ON* delay time assures that the connection of the cord of IEC-Slave has been finished. The *tRTS_DELAY_OFF* delay time assures, that the last sent data byte was received by the controlled station.

FB_IEC870_PartyLineCtrl



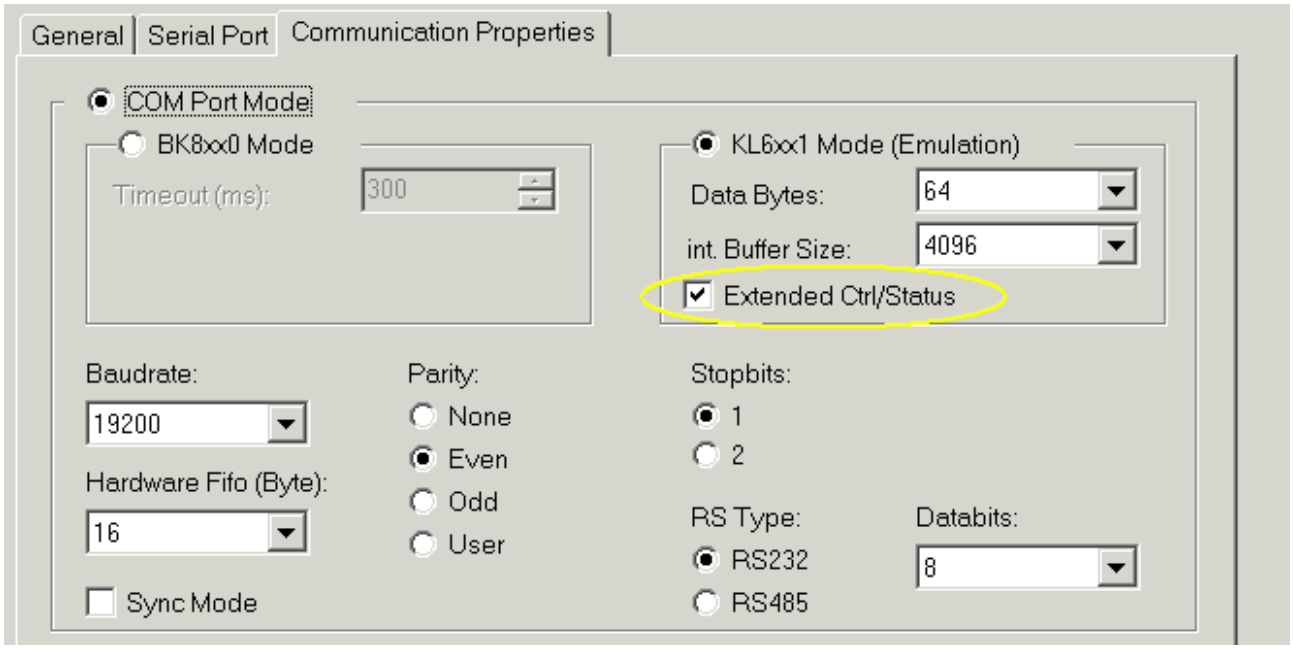

```
PROGRAM P_SerialComm_HighSpeed
VAR
    fbSerialLineCtrl    : FB_IEC870_SerialLineCtrl;
    Mode        : ComSerialLineMode_t := SERIALLINEMODE_PC_COM_PORT; (* SERIALLINEMODE_KL6_5B_STANDA
RD *)

    serial_in AT%IB0    : PcComInData;
    serial_out AT%QB0   : PcComOutData;
    KL6_in AT%IB100     : KL6inData5B;
    KL6_out AT%QB100    : KL6outData5B;

    hSerial         : T_HSERIALCTRL;


    fbPartyLineCtrl     : FB_IEC870_PartyLineCtrl;
    delay       : TON;
    tRTS_DEALY_ON    : TIME := T#100ms;
    tRTS_DELAY_OFF   : TIME := T#100ms;
    RTS_SWITCH AT%QX200.0 : BOOL; (* RTS line switch *)

END_VAR
```

```
fbSerialLineCtrl( Mode := Mode,
        Baudrate := 19200,
        NoDatabits := 8,
        Parity := PARITY_EVEN,
        Stopbits := 1,
        Handshake := HANDSHAKE_NONE,
        ContinousMode := FALSE,
        pComIn := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR( serial_in ), ADR( KL6_in ) ),
        pComOut := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR( serial_out ), ADR( KL6_out ) ),
        SizeComIn := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, SIZEOF( serial_in ), SIZEOF( KL6_in
```

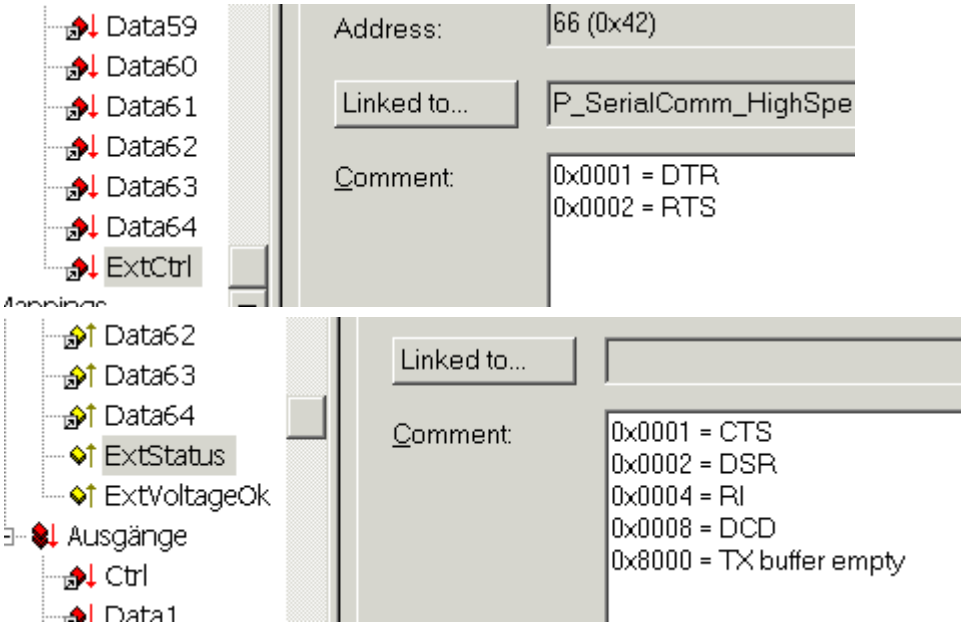
```
 ) ),
        hSerial := hSerial );


(* The deviceID may vary! *)
fbPartyLineCtrl( eMode:= eIEC870_PartylineMode_On, hSerial:= hSerial, deviceID := 1 );
IF fbPartyLineCtrl.bRTS <> fbPartyLineCtrl.bCTS THEN
    IF fbPartyLineCtrl.bRTS THEN
        RTS_SWITCH := TRUE; (* switch RTS line ON *)
        delay( in := TRUE, PT := tRTS_DEALY_ON ); (* wait until line enabled *)
        IF delay.Q THEN
            delay( in := FALSE );
            fbPartyLineCtrl.bCTS := TRUE; (* set clear to send *)
        END_IF
    ELSE
        delay( in := TRUE, PT := tRTS_DELAY_OFF ); (* wait until all data send *)
        IF delay.Q THEN
            delay( in := FALSE );
            RTS_SWITCH := FALSE; (* switch RTS line OFF *)
            fbPartyLineCtrl.bCTS := FALSE; (* reset clear to send *)
        END_IF
    END_IF
END_IF
```

Example for TwinCAT v2.10 Build >= 1313 or newer (CE image v2.16 or newer):

In an newer TwinCAT Version, the RTS line can be mapped directly as IO output in the PLC. To do this, you have to enable the Extended Ctrl/Status option in the TwinCAT System Manager. The eMode parameter at the functionsblock has to be set *to eMode := eIEC870_PartylineMode_Ext_On* . The *deviceID* parameter is not used in this modus and has not to be configurated.



The RTS_SWITCH variable has to be linked with Bit 1 in ExtCtrl (RTS output) and TX_BUFFER_EMPTY variable has to be linked with Bit 15 in ExtStatus.




```
PROGRAM P_SerialComm_HighSpeed
VAR
    fbSerialLineCtrl    : FB_IEC870_SerialLineCtrl;
    Mode        : ComSerialLineMode_t := SERIALLINEMODE_PC_COM_PORT; (* SERIALLINEMODE_KL6_5B_STANDA
RD *)

    serial_in AT%IB0    : PcComInData;
    serial_out AT%QB0   : PcComOutData;
    KL6_in AT%IB100     : KL6inData5B;
    KL6_out AT%QB100    : KL6outData5B;

    hSerial         : T_HSERIALCTRL;
    fbPartyLineCtrl     : FB_IEC870_PartyLineCtrl;
    delay       : TON;
    tRTS_DEALY_ON    : TIME := T#100ms;
    tRTS_DELAY_OFF   : TIME := T#100ms;
    RTS_SWITCH AT%QX200.0 : BOOL; (* RTS line switch *)
    TX_BUFFER_EMPTY AT%IX200.0 : BOOL; (* UART's tx buffer is empty *)
END_VAR
```

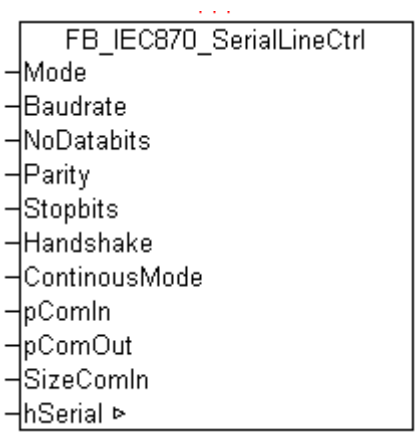
```
fbSerialLineCtrl( Mode := Mode,
        Baudrate := 19200,
        NoDatabits := 8,
        Parity := PARITY_EVEN,
        Stopbits := 1,
        Handshake := HANDSHAKE_NONE,
        ContinousMode := FALSE,
        pComIn := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR( serial_in ), ADR( KL6_in ) ),
        pComOut := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR( serial_out ), ADR( KL6_out ) ),
        SizeComIn := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, SIZEOF( serial_in ), SIZEOF( KL6_in
 ) ),
        hSerial := hSerial );



fbPartyLineCtrl( hSerial:= hSerial, eMode := eIEC870_PartylineMode_Ext_On );
IF fbPartyLineCtrl.bRTS <> fbPartyLineCtrl.bCTS THEN
    IF fbPartyLineCtrl.bRTS THEN
        RTS_SWITCH := TRUE; (* switch RTS line ON *)
        delay( in := TRUE, PT := tRTS_DEALY_ON ); (* wait until line enabled *)
     IF delay.Q THEN
            delay( in := FALSE );
            fbPartyLineCtrl.bCTS := TRUE; (* set clear to send *)
        END_IF
    ELSE
        IF TX_BUFFER_EMPTY THEN
            delay( in := TRUE, PT := tRTS_DELAY_OFF ); (* wait until all data send *)
            IF delay.Q THEN
                delay( in := FALSE );
                RTS_SWITCH := FALSE; (* switch RTS line OFF *)
                fbPartyLineCtrl.bCTS := FALSE; (* reset clear to send *)
            END_IF
```

```
        END_IF
    END_IF
END_IF
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.3 FB_IEC870_SerialLineCtrl



The function block FB_IEC870_SerialLineCtrl handles communication between a serial interface (KL60xx, EL60xx or COM port) and the IEC60870-5-101 PLC blocks.
If a serial **KL60xx** bus terminal is used for the communication, the bus terminal is initialised and configured first (baud rate, parity etc). The **PC COM port and the serial EL60xx** bus terminal has to be configured in the TwinCAT System Manager. The data received and to be sent are held in the internal buffers of the hSerial variable. The function block has to be called cyclically in the PLC task.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial         : T_HSERIALCTRL;
ND_VAR
```

hSerial [▶ 415] :

VAR_INPUT

```
VAR_INPUT
    Mode            : ComSerialLineMode_   := SERIALLINEMODE_PC_COM_PORT; (* or SERIALLINEMODE_PC_COM
_PORT *)
    Baudrate        : UDINT             := 19200;       (* KL60xx only: 115200, 57600, 38400, 19200,
 9600, 4800, 2400, 1200 *)
 NoDatabits         : BYTE      := 8;       (* KL60xx only: 7 or 8 *)
    Parity          : ComParity_t  := PARITY_EVEN;     (* KL60xx only: PARITY_NONE=0, PARITY_EVEN=1,
 PARITY_ODD=2 *)
    Stopbits        : BYTE          := 1;       (* KL60xx only: 1 or 2 *)
    Handshake       : ComHandshake_t       := HANDSHAKE_NONE; (* KL60xx only: HANDSHAKE_NONE=0, HAND
SHAKE_RTSCTS=1, HANDSHAKE_XONXOFF=2 *)
    ContinousMode   : BOOL;                 (* KL60xx only: Don't start transmission before transmit
 buffer is filled *)
    pComIn      : POINTER TO BYTE;
```

```
    pComOut         : POINTER TO BYTE;
    SizeComIn       : UINT;
END_VAR
```

Mode : The Mode input specifies unambiguously which serial hardware is being used.

Baudrate: The baud rate, provided it is supported by the serial hardware.

NoDatabits: The number of user data bits in one data byte.

Parity: The type of the parity bit in a data byte.

Stopbits: The number of stop bits per data byte.

Handshake: The type of handshake used, provided it is supported by the serial hardware.

ContinousMode: Switches on continuous transmission, provided this is supported by the serial hardware.

If ContinousMode is TRUE, transmitted data is not sent out by the serial hardware until the hardware transmit buffer is full. This means that there are no time gaps in the transmission, provided the quantity of data is similar in size to the hardware transmit buffer. Continuous mode is only necessary in special cases in which the end device reacts to time gaps with a time-out.

pComIn: Universal pointer to the input variable of the process data for the serial hardware (data types KL6inData, KL6inData5b, PcComInData, EL6inData22B). The pointer is assigned with the *ADR*() function.

pComOut: Universal pointer to the output variable of the process data for the serial hardware (data types KL6outData, KL6outData5b, PcComOutData, EL6outData22B). The pointer is assigned with the *ADR*() function.

SizeComIn: Size of the input process image of the serial hardware being used. The size is determined and assigned with the *SIZEOF*() function.

Example 1:

The example shows an ST call. Switching between two communication routes can be affected by setting the mode variables.

For mode = SERIALLINEMODE_PC_COM_PORT communication is via a serial COM port of the PC, for mode = SERIALLINEMODE_KL6_5B_STANDARD via a KL6001 bus terminal (5-byte mode).

```
PROGRAM P_SerialComm_HighSpeed
VAR
    fbSerialLineCtrl  : FB_IEC870_SerialLineCtrl;
    Mode              : ComSerialLineMode_t := SERIALLINEMODE_KL6_5B_STANDARD;
(* SERIALLINEMODE_PC_COM_PORT *)

    serial_in AT%IB0  : PcComInData;
    serial_out AT%QB0 : PcComOutData;
    KL6_in AT%IB100   : KL6inData5B;
    KL6_out AT%QB100  : KL6outData5B;

    hSerial           : T_HSERIALCTRL;
END_VAR
```

```
fbSerialLineCtrl( Mode := Mode,
        Baudrate := 19200,
        NoDatabits := 8,
        Parity := PARITY_EVEN,
        Stopbits := 1,
        Handshake := HANDSHAKE_NONE,
        ContinousMode := FALSE,
        pComIn := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR( serial_in ), ADR( KL6_in ) ),
        pComOut := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR( serial_out ), ADR( KL6_out ) ),
        SizeComIn := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, SIZEOF( serial_in ), SIZEOF( KL6_in
 ) ),
        hSerial := hSerial );
```

Example 2:

The example shows communication via a EL6001 bus terminal (22-byte mode). The serial EL6001 bus terminal must be configured in the TwinCAT System Manager (baud rate, parity etc).

```
PROGRAM P_SerialComm_HighSpeed
VAR
    fbSerialLineCtrl    : FB_IEC870_SerialLineCtrl := ( Mode := SERIALLINEMODE_EL6_22B );
    EL6_in      AT%IB4100  : EL6inData22B;
    EL6_out     AT%QB4100  : EL6outData22B;
    hSerial     : T_HSERIALCTRL;
END_VAR
```

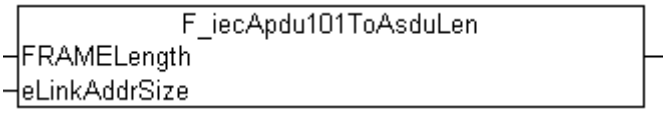
```
fbSerialLineCtrl( pComIn := ADR( EL6_in ),
        pComOut := ADR( EL6_out ),
        SizeComIn := SIZEOF( EL6_in ),
        hSerial := hSerial );
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.4 F_iecApdu101ToAsduLen

From product version: TwinCAT PLC Library IEC60870-5-101 Substation v2.0.2 and higher.



The function calculates for the IEC 60870-5-101 protocol the maximum available ASDU byte length with the configurated frame and link address field length. The maximum available ASDU length is for example required for the configuration of the ST_IEC870_5_101T buffer variable. This data structure (TX/RX data buffer) is used for the data exchange via the IEC60870-5-101 Serial Link Interface.

FUNCTION F_iecApdu101ToAsduLen: BYTE

```
VAR_INPUT
    FRAMELength     : BYTE;
    eLinkAddrSize   : E_IEC870_5_101LinkAddrSize;
END_VAR
```

FRAMELength : Max. available frame length (see compatibility list).

eLinkAddrSize: Link address field octet length (see compatibility list).

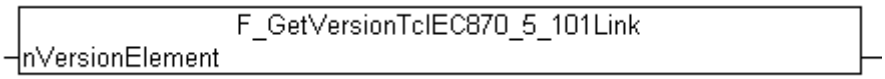
Requirements

Development Environment	Target System Type	PLC Libraries to include
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Also see about this

M_SP_NA_1 [▶ 37]

8.1.5 F_GetVersionTcIEC870_5_101Link



This function reads version information from the PLC library.

FUNCTION F_GetVersionTcIEC870_5_101Link: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.6 ST_IEC870_5_101ProtocolParams

Link layer protocol parameters. The meaning of every parameter depends on protocol or device type.

```
TYPE ST_IEC870_5_101ProtocolParams :
STRUCT
    eType           : E_IEC870_DEVICE_TYPE := eIEC870_101_SLAVE;
    eLinkReset      : E_IEC870_5_101LinkReset := eIEC870_LinkReset_CU;
    linkAddr        : DWORD := 1;
    eLinkAddrSize   : E_IEC870_5_101LinkAddrSize := eIEC870_LinkAddr_TwoOctets;
    eLinkMode       : E_IEC870_5_101LinkMode := eIEC870_LinkMode_Unbalanced;
    eFrameType      : E_IEC870_5_101FrameType := eIEC870_FrameType_FT1_2;
    tRxTimeout      : TIME := T#5s;
    tTxTimeout      : TIME := T#5s;
    bForceC1Res     : BOOL := TRUE;
    bForceC2Res     : BOOL := TRUE;
    tClass1Poll     : TIME := T#200ms;
    tClass2Poll     : TIME := T#200ms;
    nRetries        : BYTE := 3;
    tRetry          : TIME := T#100ms;
    tResponse       : TIME := T#1s;
    tTestLink       : TIME := T#5s;
    tPollDFC        : TIME := T#1s;
    FRAMELength     : BYTE(MIN_IEC870_5_101Link_FRAMELEN..MAX_IEC870_5_101Link_FRAMELEN)
                := MAX_IEC870_5_101Link_FRAMELEN;
    bRetainBuffer   : BOOL := FALSE;
    tMaxPollDelay   : TIME := T#0s;
    tLinkPollCycle  : TIME := T#10s;
END_STRUCT
END_TYPE
```

The lower table shows the different device type parameter configuration. Fixed values can not be configured.

Legend:

- **X** The parameter is used and can be configured.
- **N/A** The parameter is not used and cannot be configured.

Parameter name	Init/default value	IEC 60870-5-101 controlled station (slave, unbalanced mode)	IEC 60870-5-101 control station (master, unbalanced mode)	IEC 60870-5-103 control station (master, unbalanced mode)	IEC 60870-5-102 control station (master, unbalanced mode)	Description
eType	eIEC870_101_SLAVE	fixed value, eIEC870_101_SLAVE	fixed value, eIEC870_101_MASTER	fixed value, eIEC870_103_MASTER	fixed value, eIEC870_102_MASTER	Configures the protocol and device type [▶ 414].
eLinkReset	eIEC870_LinkReset_CU	N/A	X	X	fixed value, eIEC870_LinkReset_CU	Configures the link layer (initialization) reset type [▶ 414].
linkAddr	1	own address	address of remote communication unit	address of remote communication unit	address of remote communication unit	Link address.
eLinkAddrSize	eIEC870_LinkAddr_TwoOctets	X	X	X	X	Link adress octet size [▶ 332].
eLinkMode	eIEC870_LinkMode_Unbalanced	fixed value, eIEC870_LinkMode_Unbalanced	fixed value, eIEC870_LinkMode_Unbalanced	fixed value, eIEC870_LinkMode_Unbalanced	fixed value, eIEC870_LinkMode_Unbalanced	Link mode [▶ 413] (balanced/ unbalanced).
eFrameType	eIEC870_FrameType_FT1_2	fixed value, eIEC870_FrameType_FT1_2	fixed value, eIEC870_FrameType_FT1_2	fixed value, eIEC870_FrameType_FT1_2	fixed value, eIEC870_FrameType_FT1_2	Telegramm frame format [▶ 412]. Only F1.2 is supported.
tRxTimeout	T#5s	X	X	X	X	Max. frame receive timeout time. This parameter specifies the time to wait until frame is received before a timeout condition is set. If the timeout condition is set, all bytes in the frame received will be discarded.

Parameter name	Init/default value	IEC 60870-5-101 controlled station (slave, unbalanced mode)	IEC 60870-5-101 control station (master, unbalanced mode)	IEC 60870-5-103 control station (master, unbalanced mode)	IEC 60870-5-102 control station (master, unbalanced mode)	Description
tTxTimeout	T#5s	X	X	X	X	Max. frame send timeout time.This parameter specifies the time to wait until frame is send before a timeout condition is set.
bForceC1Res	TRUE	X	N/A	N/A	N/A	If set (TRUE), in response to a class 2 poll, a controlled station may respond with class 1 data when there is class 1 data available (optimized, higher class 1 data throughput). If not set (FALSE), in response to class 2 poll, a controlled station may respond with NOUSERDATA when there is class 1 data available (empty telegramms).
bForceC2Res	TRUE	X	N/A	N/A	N/A	If set (TRUE), in response to a class 2 poll, a controlled station may respond with class 2 data even tough class 1 data is available (optimized, higher class 2 data throughput).

Parameter name	Init/default value	IEC 60870-5-101 controlled station (slave, unbalanced mode)	IEC 60870-5-101 control station (master, unbalanced mode)	IEC 60870-5-103 control station (master, unbalanced mode)	IEC 60870-5-102 control station (master, unbalanced mode)	Description
						If not set (FALSE), in response to class 2 poll, a controlled station may respond with NOUSERDATA even tough class 1 data is available (empty telegramms).
tClass1Poll	T#200ms	N/A	X	X	X	How often to check for class 1 data (unbalanced mode only).
tClass2Poll	T#200ms	N/A	X	X	X	How often to check for class 2 data. If the RTU has class 1 data, this is shown in the response to the poll for class 2 data, in this case the device will automatically poll for class 1 data (unbalanced mode only).
nRetries	3	Not used in v3.0.6 and lower(error after first disturbed frame), configurable in v3.0.8 and higher	X	X	X	This parameter specifies the number of retries (0 to 255) if a response is not received.
tRetry	T#100ms	N/A	X	X	X	Repeated frames delay time.
tResponse	T#1s	N/A	X	X	X	This parameter specifies the time to wait for a response to a primary

Parameter name	Init/default value	IEC 60870-5-101 controlled station (slave, unbalanced mode)	IEC 60870-5-101 control station (master, unbalanced mode)	IEC 60870-5-103 control station (master, unbalanced mode)	IEC 60870-5-102 control station (master, unbalanced mode)	Description
						message. If the timeout is recognized, the message will be retransmitted up to the number of times specified in the *nRetries* parameter.
tTestLink	T#5s	N/A	N/A	N/A	N/A	How often to send test link messages. Note that this has effect only for balanced mode, because test link messages are undefined in unbalanced mode and will not be sent.
tPollDFC	T#1s	N/A	X	X	X	How often to read link status if DFC bit is set. Remote communication unit sets the DFC bit if further messages may cause data overflow.
FRAMELength	MAX_IEC870_5_101Link_FRAMELEN	X	X	X	X	Max. frame length L.
bRetainBuffer	FALSE	available from product version v3.0.2 and higher	X	X	X	The internal TX/RX buffer are reset (cleared) during the initialisation of the link layer if this parameter is set.
tMaxPollDelay	T#0s (= 0: not used, <> 0: used	available from product version v3.0.2 and higher	N/A	N/A	N/A	Max. time between link polls before

Parameter name	Init/default value	IEC 60870-5-101 controlled station (slave, unbalanced mode)	IEC 60870-5-101 control station (master, unbalanced mode)	IEC 60870-5-103 control station (master, unbalanced mode)	IEC 60870-5-102 control station (master, unbalanced mode)	Description
						station is declared OFFLINE.
tLinkPollCycle	T#10s	N/A	X	X	X	How often to read link status if the connection is offline.

Additional information for **bForceC1Res** and **bForceC2Res** parameters:

Irrespective whether you set both parameters, one of them or none of them, the ACD bit is set accordingly in order to notify the control station whether Class 1 or Class 2 data should be queried next. These parameters only influence the behaviour of the substation for Class 2 queries.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Also see about this

M_SP_NA_1 [▶ 37]

8.1.7 E_IEC870_5_101FrameType

```
TYPE E_IEC870_5_101FrameType :
(
    eIEC870_FrameType_FT1_2
);
END_TYPE
```

IEC 60870-5-101 Frame Type. Only FT1.2 is supported.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Development Environment	Target System	PLC libraries to include
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)

8.1.8 E_IEC870_5_101LinkMode

Configures the data transfer mode.

```
TYPE E_IEC870_5_101LinkMode :
(
    eIEC870_LinkMode_Unbalanced,
    eIEC870_LinkMode_Balanced
);
END_TYPE
```

eIEC870_LinkMode_Unbalanced: Unbalanced transmission (asymmetric data transfer). The controlling station controls the data transfer by polling the controlled outstations sequentially. The controlling station initiates the message transfers while the controlled outstations can transmit only in response to the message from the controlling station.

eIEC870_LinkMode_Balanced: Balanced transmission (symmetric data transfer). Available only in newer product versions:

- TwinCAT PLC Library IEC 60870-5-101 Master v2.0.1 and higher;
- TwinCAT PLC Library IEC 60870-5-101 Slave v4.0.1 and higher;

When using balanced transmission, each station can initiate message transfer. The balanced transmission is restricted to point-to-point and to multiple point-to-point configurations.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.9 E_IEC870_5_101SerialLinkState

```
TYPE E_IEC870_5_101SerialLinkState :
(
    eSERIALLINK_DISCONNECTED,
    eSERIALLINK_ESTABLISHED,
    eSERIALLINK_SUSPENDED
);
END_TYPE
```

Connection status

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Development Environment	Target System	PLC libraries to include
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)

8.1.10 E_IEC870_5_101PartyLineMode

From product version: TwinCAT PLC Library IEC60870-5-101 Substation v2.0.2 and higher

```
TYPE E_IEC870_5_101PartyLineMode :
(
    eIEC870_PartylineMode_Off := 0, (* Disabled *)
 eIEC870_PartylineMode_On (* Enabled *)
    eIEC870_PartylineMode_Ext_On := 2 (* Enabled, with extended serial state/control interface *)
 );
END_TYPE
```

Partyline communication mode. See in the description of function block: FB_IEC870_PartyLineCtrl [▶ 399].

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.11 E_IEC870_5_101LinkReset

```
TYPE E_IEC870_5_101LinkReset :
(
    eIEC870_LinkReset_None  := 0,   (* Disabled *)
    eIEC870_LinkReset_CU    := 1,   (* Reset communication unit *)
    eIEC870_LinkReset_UP    := 2,   (* Reset user process *)
    eIEC870_LinkReset_FCB   := 3    (* Reset FCB bit *)
);
END_TYPE
```

Configures the link layer (initialization) reset type.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1307	PC or CX (x86, ARM)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)

8.1.12 E_IEC870_DEVICE_TYPE

```
TYPE E_IEC870_DEVICE_TYPE :
(
    eIEC870_101_SLAVE := 0, (* Secondary (responding) station *)
    eIEC870_101_MASTER := 1, (* Primary (initiating) station *)
```

```
    eIEC870_102_SLAVE := 2, (* Secondary (responding) station *)
    eIEC870_102_MASTER := 3, (* Primary (initiating) station *)

    eIEC870_103_SLAVE := 4, (* Secondary (responding) station *)
    eIEC870_103_MASTER := 5 (* Primary (initiating) station *)
);
END_TYPE
```

Protocol and device type.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1307	PC or CX (x86, ARM)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)

8.1.13 T_HSERIALCTRL

A variable of this type represents a connection handle of the serial interface. The structural elements should not be written to or modified directly. Variables of this type are used for exchanging the data to be sent or received between the function block FB_IEC870_SerialLineCtrl [▶ 404] and other IEC 60870-5-101 PLC blocks.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.2 Examples

Used controlled or controlling station configuration parameters:

- Standard PC interface: **COM1**
- **19200 Baud**
- Link address: **1**
- Link address size: **1 octet**
- Cause of transfer size: **1 octet**
- Originator address: **not used**
- Common ASDU address: **7**
- Common ASDU address size: **1 octet**
- Information object address size: **2 octets**

To compile this sample code you need the TwinCAT Version 2.10 Build > 1328!

The example applications contain two program parts:
P_MAIN_LowSpeed() implements the protocoll and P_SerialComm_HighSpeed implements the TwinCAT communication over serial communication interface.

Requirements

PLC Project	TwinCAT System Manager Configuration	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700173963/.zip (unbalanced mode) https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700175371/.zip (balanced mode)	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700173963/.zip (unbalanced mode) https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700175371/.zip (balanced mode)	Simple **Slave-Application** which uses the TwinCAT IEC60870-5-101 Serial Link Interface. The function block FB_IEC870_5_101SServices implements test command, clock synchronisation command, general interrogation command and counter interrogation command (mode C). The **bitstring datapoint (M_BO_TB_1, with time tag CP56Time2a, information object address: 100)** is send every second to the controlling station with the cause of transfer: spontanous. One **counter value (M_IT_NA_1, without time tag, information object address: 400)** is send as response to counter interrogation command (mode C) to the controlling station.
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700176779/.zip (unbalanced mode) https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700178187/.zip (balanced mode)	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700176779/.zip (unbalanced mode) https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11700178187/.zip (balanced mode)	Simple **Master-Application** which uses the TwinCAT IEC60870-5-101 Serial Link Interface. The function block FB_IEC870_5_101MServices implements test command, clock synchronisation command, general interrogation command and counter interrogation command.

8.3 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.);
2. Compare the interoperability check list of control station and controlled station;
3. Check the IO configuration and mapping of PLC variables in TwinCAT System Manager (configuration of serial interface [▶ 505], baudrate, parity, stoppbits settings). Compare them to the parameters set in the communication partner;
4. Check whether the FB_IEC870_5_101TProtocol [▶ 398]() function block issues an error code. The documentation for the error codes can be found here: Overview of error codes;
5. Check the protocol parameters [▶ 407] that are transferred to the function block (link address, link address octet size, FRAMELength etc.). Compare them to the parameters set in the communication partner;
6. Check the configured address lengths (instance of ST_IEC870_5_101TBuffer [▶ 37], TX/RX data buffer): ASDU address octet size, information object address octet size, cause of transfer octet size, max ASDU size. Compare them to the parameters set in the communication partner;

7. Check the configured data points (type, information object address etc.);
8. Check if the other communication partner issues an error code;
9. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings;

9 TcIEC870_5_102Link: IEC 60870-5-102 Serial Link Interface (master)

The control stations (masters) according to IEC60870-5-102 can be realised in the TwinCAT PLC with the PLC functions and function blocks. The IEC60870-5-102 transmission protocol is companion standard tor the transmission of integrated totals in electric power systems.

The end application is imposed on the software interface of the PLC library. Unlike the implementation of other 101/104 TwinCAT products only one "so called" 'Low level' interface is available. The reason: The IEC 60870-5-102 devices are featuring a lot of vendor specific data. The 'Low level' interface allows the access to this data. The characteristics of both interfaces are described briefly below.

'Low level' interface: IEC 60870-5-102 Serial Link Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/ Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

Interoperability check list

for TwinCAT PLC Library: IEC 60870-5-102 control station (master). Here you can https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11763594379/.zip

System requirements

Development environment:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT installation level: TwinCAT PLC or higher;
- TwinCAT System version 2.10.0 Build >= 1319 or higher;

Target system:

- Industrie PC oder Embedded PC/CX (x86, ARM);
- Operating system:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86, ARM) (image v2.16 or higher);
- TwinCAT PLC runtime system version 2.10.0 or higher;
- Serial COM port or KL6xxx or EL6xxx bus terminal (**EL6xxx support implemented in product version 1.0.4 and higher**);

Product components

- **TcIEC870_5_102Link.Lib** (implements the transfer procedures for the transport of IEC 60870-5-102 ASDU's);
  This library have to be included in the PLC project. All other libraries are included automatically.
- TcIEC870_5_101Link.Lib (base library, implements the transfer procedures for the transport of ASDU's via the serial interfaces of PC and Beckhoff KL6xxx/EL6xxxbus terminal);
- TcIEC870_5_101.Lib (implements the connection functions and common data types);

- COMLibV2.Lib (implements the functions for the serial COM- or KL6xxx/EL6xxx communication);

Installation on Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

The PLC libraries are copied into the folder: ..\TwinCAT\Plc\Lib.

Installation on Windows CE

You don't have to install any additional components on the CE platform.

This document is not full product manual. Please install the full version of the Beckhoff Information System.

Hint:

You will find it

- on the Beckhoff Product DVDs
- on our Web-Server: http://www.beckhoff.com under download.

Examples

Example projects are in the Beckhoff Information System documentation of TwinCAT PLC library.

Link to example overview page ('low level' interface): IEC 60870-5-102 Serial Link Interface [▶ 431];

Further Documentation

- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 397];
- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
- Documentation of TwinCAT PLC Library: Serial communication;
- IEC 60870-5-1 Transmission frame formats;
- IEC 60870-5-2 Link transmission procedures;
- IEC 60870-5-3 General structure of application data;
- IEC 60870-5-4 Definition and coding of application information elements;
- IEC 60870-5-5 Basic application functions;
- IEC 60870-5-101 Companion Standard for basic telecontrol tasks;
- IEC 60870-5-101:1995/A1:2000 Companion Standard for basic telecontrol tasks. Amendment 1;
- IEC 60870-5-101:1995/A2:2001 Companion Standard for basic telecontrol tasks. Amendment 2;
- IEC 60870-5-102:1996 Transmission protocols - Companion Standard for the transmission of integrated totals in electric power systems;

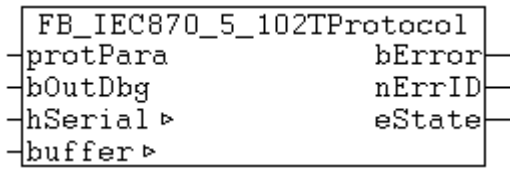
9.1 Introduction

Simple projects with complete sources can be found under: Transport-Interface-Examples.Transport-Interface-Examples [▶ 431]

A PLC application which should communicate with a or sub station via the transport interface needs the following resources:

- Instance of communication function block: FB_IEC870_5_102TProtocol [▶ 420];
- Instance of TX/RX data buffer: ST_IEC870_5_102TBuffer [▶ 423];
- Instance of function block to manipulate the TX/RX data buffer: FB_IEC870_5_102TBufferCtrl [▶ 421];

9.1.1 FB_IEC870_5_102TProtocol



The FB_IEC870_5_102TProtocol communication block implements the transport functions of the IEC 60870-5-1 and IEC 60870-5-2 link layer. In the event of a protocol error an associated error code is issued at the function block output and the data transfer is interrupted. The data exchange can be reactivated by calling the INIT function block action. The communication block expects a TX/RX data buffer variable. This variable must be transferred to the block via VAR_IN_OUT.

The function block features the following actions:

- **INIT** (initialises the function block). In the default configuration the TX/RX data buffer are cleared. The bRetainBuffer member variable of the protPara structure controls the reset bahaviour of TX/RX data buffer.

Protocol configuration

The communication block has a structured protPara variable. Protocol parameters such as timeout times or connection addresses etc. can be configured via this variable.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial     : T_HSERIALCTRL;         (* Serial link control handle *)
    buffer      : ST_IEC870_5_102TBuffer; (* TX/RX data buffer *)
END_VAR
```

hSerial: Connection handle [▶ 415] to FB_IEC870_SerialLineCtrl [▶ 404] function block. The data to be sent and received are exchanged with the FB_IEC870_SerialLineCtrl function block.

buffer: TX/RX data buffer.

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_101ProtocolParams; (* IEC60870-5-101/102 protocol parameters *)
    bOutDbg     : BOOL;        (* Enable/disable debug output *)
END_VAR
```

protPara: IEC60870-5-101/102 protocol parameters [▶ 407].

bOutDbg:: Activates/deactivates the debug output of the frames in the TwinCAT System Manager logger view.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED; (* Data transfer state
*)
ND_VAR
```

bError: This output is switched to TRUE if an error occurs during data transfer.

nErrID: Returns an error code if the bError output is set.

eState: Status [▶ 413] of the data exchange to the master.

Example:

Sample projects: IEC60870-5-102 Serial Link Interface [▶ 431]

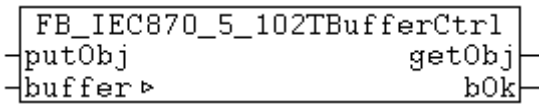
Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

Also see about this

- ST_IEC870_5_102TBuffer [▶ 423]

9.1.2 FB_IEC870_5_102TBufferCtrl



This function block changes the content of the TX/RX data buffer. This data buffer ist used by the communication via the IEC60870-5-102 Serial Link Interface.

The function block has the following actions:

- **RxRemoveObj** (removes the oldest Fifo entry from the RX-Fifo);
- **RxReset** (deletes all RX-Fifo entries, resets the RX-Fifo);
- **TxAddObj** (inserts a new Fifo entry in the TX-Fifo);
- **TxReset** (deletes all TX-Fifo entries, resets the TX-Fifo);

The content of the TX/RX data buffer can be changed by the call of one of the listed actions above.

VAR_IN_OUT

```
VAR_IN_OUT
    buffer  : ST_IEC870_5_102TBuffer; (* TX/RX data buffer *)
END_VAR
```

buffer: TX/RX data buffer [▶ 423]. The TX/RX buffer parameters (like e.g. asduSize) have to be configured before using.

VAR_INPUT

```
VAR_INPUT
    putObj  : ST_IEC870_5_102AOGen; (* ASDU to send *)
END_VAR
```

putObj: Data unit (ASDU) to be sent.

VAR_OUTPUT

```
VAR_OUTPUT
    getObj  : ST_IEC870_5_102AOGen; (* received ASDU *)
    bOk     : BOOL; (* TRUE = action succesfully, FALSE=Fifo overflow or fifo empty *)
END_VAR
```

getObj: Received data unit (ASDU).

bOk: This variable becomes TRUE, if a new entry was inserted or removed successfully from the Fifo. This variable becomes FALSE at a buffer overflow and if no entry could be removed, because the Fifo was already empty.

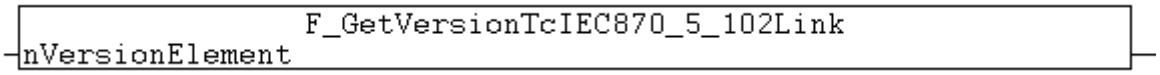
Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

Also see about this

- ST_IEC870_5_102AOGen [▶ 424]
- ST_IEC870_5_102AOGen [▶ 424]

9.1.3 F_GetVersionTcIEC870_5_102Link



This function reads version information from the plc library.

FUNCTION F_GetVersionTcIEC870_5_102Link: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

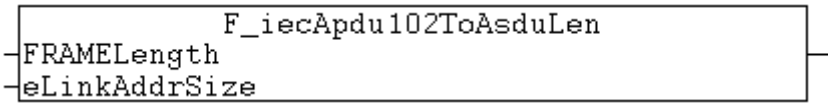
nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib, (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

9.1.4 F_iecApdu102ToAsduLen



The function calculates for the IEC 60870-5-102 protocol the maximum available ASDU byte length with the configurated frame and link address field length. The maximum available ASDU length is for example required for the configuration of the ST_IEC870_5_102T buffer variable. This data structure (TX/RX data buffer) is used for the data exchange via the IEC60870-5-102 Serial Link Interface.

FUNCTION F_iecApdu102ToAsduLen: BYTE

```
VAR_INPUT
    FRAMELength     : BYTE;
    eLinkAddrSize   : E_IEC870_5_101LinkAddrSize;
END_VAR
```

FRAMELength : Max. available APDU frame length (see compatibility list).

eLinkAddrSize: Link address field octet length (see compatibility list).

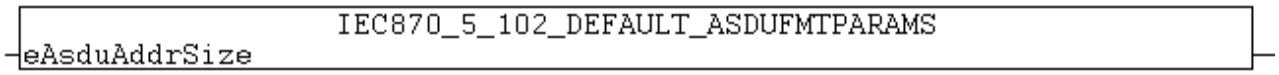
Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib, (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

Also see about this

M_IT_TA_2 [▶ 215]

9.1.5 IEC870_5_102_DEFAULT_ASDUFMTPARAMS



This function sets the default IEC 60870-5-102 frame format parameters:

- Cause of transfer octet length;
- Common ASDU address octet length;
- Information Object address octet length;

FUNCTION IEC870_5_102_DEFAULT_ASDUFMTPARAMS: ST_IEC870_5_101AsduFmtParams

ST_IEC870_5_101AsduFmtParams [▶ 215]

```
VAR_INPUT
    eAsduAddrSize : E_IEC870_5_101AsduAddrSize;
END_VAR
```

eAsduAddrSize : Common ASDU address octet length.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib, (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

9.1.6 ST_IEC870_5_102TBuffer

This data structure is used for data exchange (TX/RX data buffer) via the IEC60870-5-102 Serial Link Interface.

```
TYPE ST_IEC870_5_102TBuffer :
STRUCT
    eDbg           : E_IEC870_5_101FifoDbgFlags :=eIEC870_FIFO_DBG_OFF; (* enable/
disable log view hex output *)
    eAsduAddrSize  : E_IEC870_5_101AsduAddrSize;(* Common ASDU address field size (1..2 octets) *)
    asduSize       : BYTE := 253; (* max. length of ASDU data ( IEC 60870-5-101/102/103 = 253 octets
, IEC 60870-5-104 = 249 octets )*)
    mode           : DWORD := 0;

    dataLink       : ST_IEC870_5_101DataLink; (* internal tx/rx buffer *)
ND_STRUCT
END_TYPE
```

eDbg: Debug output parameter.

eAsduAddrSize: Common asdu address octet size.

asduSize : Max. octet length of the asdu.

mode: Reserved, not used. This value has to be zero.

dataLink: The access to the elements of this data structure should only occur with an instance of the function block FB_IEC870_5_102TBufferCtrl [▶ 421], not directly.

The TX/RX data buffer uses internal two send Fifos and one receive Fifo:

1. Class 1 send Fifo with (high prior) data;
2. Class 2 send Fifo with (low prior) data;
3. Receive Fifo (for Class 1 and Class 2 data);

The lower transport functions of the library emtpy first the class 1 Fifo and afterwards the class 2 Fifo. The class 2 data are only sent, if the class 1 Fifo contains no data to be sent.

Each internal Fifo hast a fixed size of 10000 bytes. This is enough for the most applications, because the number of frames which can once be sent is limited by the iK and iW protocol parameters.
According to experience it is possible to store 200 ADSUs with an information element (object), or approx. 20 ADSUs with with a sequence of 100 information elements (objects) in each Fifo.

If a higher number of frames (to be sent or to be received) should be buffered (e.g. ~20000), they can be kept in external buffers/Fifos which are defined by the PLC programmer. The PLC application can refill the TwinCAT send Fifos with own Fifo entries, or empty the TwinCAT receive Fifo.

As an alternative it is possible to use two buffers and to fill/read them alternately and give them to the communication block .

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMLibV2.Lib are included automatically)

Also see about this

- M_IT_TA_2 [▶ 215]

9.1.7 ST_IEC870_5_102AOGen

Variables of this type represent an ASDU object.

```
TYPE ST_IEC870_5_102AOGen:
STRUCT
    head    : ST_IEC870_5_101FifoHead :=(      source := ( link := 0, addr := 0 ),
                  target := ( link := 0, addr := 0 ),
```

```
                    ctrl := 0 ); (* Header *)

    ident   : ST_IEC870_5_102DataUnit_Ident := ( eType := ASDU_TYPEUNDEF_2,
                        nObj := 0,
                        bSQ := FALSE,
                        bT := FALSE,
                        bPN := FALSE,
                        eCOT := 0,
                        asduAddr := 0,
                        eClass := eIEC870_Class_None ); (* Data unit identifier *)

    info    : ST_IEC870_5_102AOInfoObj  := (    rcdAddr     := 0,
                        n       := 0,
                        stream  := ( length := 0 ) ); (* information object *)
END_STRUCT
END_TYPE
```

head: Header (reserved).

ident: Identification field [▶ 426] in the data unit (ASDU).

info: Information object [▶ 425] /Information element data field.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMLibV2.Lib are included automatically)

9.1.8 ST_IEC870_5_102AOInfoObj

Information object description.

```
TYPE ST_IEC870_5_102AOInfoObj :
STRUCT
    rcdAddr : BYTE; (* Record address <0..255> *)
    stream  : ST_IEC870_5_101Stream; (* Information element of max. length *)
END_STRUCT
END_TYPE
```

rcdAddr: Record address or list address of accounting period.

stream: Information element data of max. length (byte array buffer).

The list addresses are specified as follows: <0..255>

- <0> Default, if no other value is specified
- <1> List address for counter states from the beginning of the accounting period
- <2..10> Reserved for further compatible specifications
- <11> List address for counter states from measurement period 1
- <12> List address for counter states from measurement period 2
- <13> List address for counter states from measurement period 3
- <14..20> Reserved for further compatible specifications
- <21> List address for counter states (daily values) from measurement period 1
- <22> List address for counter states (daily values) from measurement period 2
- <23> List address for counter states (daily values) from measurement period 3
- <24..30> Reserved for further compatible specifications
- <31> List address for counter states (monthly values) from measurement period 1
- <32> List address for counter states (monthly values) from measurement period 2

- <33> List address for counter states (monthly values) from measurement period 3
- <34..40> Reserved for further compatible specifications
- <41> List address for counter states (yearly values) from measurement period 1
- <42> List address for counter states (yearly values) from measurement period 2
- <43> List address for counter states (yearly values) from measurement period 3
- <44..49> Reserved for further compatible specifications
- <50> Oldest single message
- <51> Complete list with single messages
- <52> Partial list 1 with single messages
- <53> Partial list 2 with single messages
- <54> Partial list 3 with single messages
- <55> Partial list 4 with single messages
- <56..127> Reserved for further compatible specifications
- <128 - 255> For particular applications (private area)

The size of a partial list is a system parameter.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMLibV2.Lib are included automatically)

Also see about this

M_IT_TA_2 [▶ 215]

9.1.9 ST_IEC870_5_102DataUnit_Ident

Identification field of data unit (ASDU).

```
TYPE ST_IEC870_5_102DataUnit_Ident:
STRUCT
    eType    : E_IEC870_5_102TypeID; (* ASDU type identifier <0..255> *)
 bSQ     : BOOL; (* Single/Sequence *)
    nObj     : BYTE(0..127); (* Number of information objects or elements <0..127> *)
    bT       : BOOL; (* Test bit *)
    bPN      : BOOL; (* Positive/Negative confirmation, FALSE = positive, TRUE = negative *)
    eCOT     : E_IEC870_5_102COTType; (* Cause of transmission <0..63> *)
    asduAddr : DWORD; (* Common address of ASDU (1..2 octets) *)
    eClass   : E_IEC870_5_101ClassType; (* Object class *)
END_STRUCT
END_TYPE
```

eType: ASDU type identifier [▶ 427]: <0..255>.

bSQ: Sequence flag.

nObj: Number of information objects or information elements: <0..127>.

bT: Test flag.

bPN: Positive/negative confirmation. FALSE = positive, TRUE = negative.

eCOT: Cause of transmission [▶ 428]: <0..63>.

asduAddr: Common ASDU address (one or two octets).

eClass: Priority class.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMLibV2.Lib are included automatically)

Also see about this

M_IT_TA_2 [▶ 215]

9.1.10 E_IEC870_5_102TypeID

```
TYPE E_IEC870_5_102TypeID:
(
    ASDU_TYPEUNDEF_2 := 0, (* (000) not allowed *)(* Process inormation in monitor direction *)
    M_SP_TA_2 := 1, (* (001) Single-point information with time tag *)
    M_IT_TA_2 := 2, (* (002) Accounting integrated totals, 4 octets each *)
    M_IT_TB_2 := 3, (* (003) Accounting integrated totals, 3 octets each *)
    M_IT_TC_2 := 4, (* (004) Accounting integrated totals, 2 octets each *)
    M_IT_TD_2 := 5, (* (005) Periodical reset accounting integrated totals, 4 octets each *)
    M_IT_TE_2 := 6, (* (006) Periodical reset accounting integrated totals, 3 octets each *)
    M_IT_TF_2 := 7, (* (007) Periodical reset accounting integrated totals, 2 octets each *)
    M_IT_TG_2 := 8, (* (008) Operational integrated totals, 4 octets each *)
    M_IT_TH_2 := 9, (* (009) Operational integrated totals, 3 octets each *)
    M_IT_TI_2 := 10, (* (010) Operational integrated totals, 2 octets each *)
    M_IT_TK_2 := 11, (* (011) Periodical reset operational integrated totals, 4 octets each *)
    M_IT_TL_2 := 12, (* (012) Periodical reset operational integrated totals, 3 octets each *)
    M_IT_TM_2 := 13, (* (013) Periodical reset operational integrated totals, 2 octets each *)
    (* System information in monitor direction *)
    M_EI_NA_2 := 70, (* (070) End of initialization *)
    P_MP_NA_2 := 71, (* (071) Manufacturer and product specification of integrated total DTE *)
    M_TI_TA_2 := 72, (* (072) Current system time of integrated total DTE *)
    (* System information in control direction *)
    C_RD_NA_2 := 100, (* (100) Read manufacturer and product specification *)
    C_SP_NA_2 := 101, (* (101) Read record of single-point information with time tag *)
    C_SP_NB_2 := 102, (* (102) Read record of single-
point information with time tag of a selected time range *)
    C_TI_NA_2 := 103, (* (103) Read current system time of integrated total DTE *)
    C_CI_NA_2 := 104, (* (104) Read accounting integrated totals of the oldest integration period *)
    C_CI_NB_2 := 105, (* (105) Read accounting integrated totals of the oldest integration period an
d of a selected range of addresses *)
    C_CI_NC_2 := 106, (* (106) Read accounting integrated totals of a specific past integration peri
od *)
    C_CI_ND_2 := 107, (* (107) Read accounting integrated totals of a specific past integration peri
od and of a selected range of addresses *)
    C_CI_NE_2 := 108, (* (108) Read periodical reset accounting integrated totals of the oldest inte
gration period *)
    C_CI_NF_2 := 109, (* (109) Read periodical reset accounting integrated totals of the oldest inte
gration period and of a selected range of addresses *)
    C_CI_NG_2 := 110, (* (110) Read periodical reset accounting integrated totals of a specific past
 integration period *)
    C_CI_NH_2 := 111, (* (111) Read periodical reset accounting integrated totals of a specific past
 integration period and of a selected range of addresses *)
    C_CI_NI_2 := 112, (* (112) Read operational integrated totals of the oldest integration period *
)
    C_CI_NK_2 := 113, (* (113) Read operational integrated totals of the oldest integration period a
nd of a selected range of addresses *)
    C_CI_NL_2 := 114, (* (114) Read operational integrated totals of a specific past integration per
iod *)
    C_CI_NM_2 := 115, (* (115) Read operational integrated totals of a specific past integration per
iod and of a selected range of addresses *)
    C_CI_NN_2 := 116, (* (116) Read periodical reset operational integrated totals of the oldest int
egration period *)
    C_CI_NO_2 := 117, (* (117) Read periodical reset operational integrated totals of the oldest int
egration period and of a selected range of addresses *)
    C_CI_NP_2 := 118, (* (118) Read periodical reset operational integrated totals of a specific pas
t integration period *)
```

```
    C_CI_NQ_2 := 119, (* (119) Read periodical reset operational integrated totals of a specific pas
t integration period and of a selected range of addresses *)
    C_CI_NR_2 := 120, (* (120) Read accounting integrated totals of a specific past integration peri
od of a selected time range and of a selected range of addresses *)
    C_CI_NS_2 := 121, (* (121) Read periodical reset accounting integrated totals of a specific past
 integration period of a selected time range and of a selected range of addresses *)
    C_CI_NT_2 := 122, (* (122) Read operational integrated totals of a specific past integration per
iod of a selected time range and of a selected range of addresses *)
    C_CI_NU_2 := 123, (* (123) Read periodical reset operational integrated totals of a specific pas
t integration period of a selected time range and of a selected range of addresses *)

    M_DS_TA_2 := 128, (* (128) *)
    P_ME_NA_2 := 129, (* Parameters of the measuring point *)
    M_DS_TB_2 := 130, (**)
    M_CH_TA_2 := 131, (**)
    C_PK_2 := 132, (* Load private key *)
    C_TA_VC_2 := 133, (* Read tariff information ( current values ) *)
    C_TA_VM_2 := 134, (* Read tariff information ( stored values ) *)
    M_TA_VC_2 := 135, (* Tariff information ( current values ) *)
    M_TA_VM_2 := 136, (* Tariff information ( stored values ) *)
    C_TA_CP_2 := 137, (* Close accounting period *)
    M_IB_TG_2 := 139, (* Block of operational integrated totals ( absolute values ) *)
    M_IB_TK_2 := 140, (* Block of periodical reset operational integrated totals ( increment values
) *)
    C_RM_NA_2 := 141, (* Read configuration data of the meter device *)
    M_RM_NA_2 := 142, (* Configuration of the meter device *)
    C_MR_NA_2 := 143, (* Change configuration data of the meter device *)
    C_PC_NA_2 := 144, (* (144) *)
    M_PC_NA_2 := 145, (* (145) *)
    C_MC_NA_2 := 146, (* (146) *)
    C_DF_NA_2 := 147, (* (147) *)
    M_DF_NA_2 := 148, (* (148) *)
    C_MF_NA_2 := 149, (* (149) *)

    C_DS_TA_2 := 180, (* (180) *)
    C_CS_TA_2 := 181, (* (181) Change date and time ( Time syncronization ) *)
    C_PI_NA_2 := 182, (* (182) Read parameters of the measuring point *)
    C_AC_NA_2 := 183, (* (183) Start session and send access key *)

    C_DS_TB_2 := 184, (* (184) *)
    C_CH_TA_2 := 185, (* (185) *)
    C_MH_TA_2 := 186, (* (186) *)
    C_FS_NA_2 := 187, (* (187) Finish session *)
    C_MP_NA_2 := 188, (* (188) *)
    C_CB_NT_2 := 189, (* (189) Read a block of operational integrated totals of a time period and a
selected address *)
    C_CB_UN_2 := 190 (* (190) Read a block of periodical reset operational integrated totals of a ti
me period and a selected address *)
);
END_TYPE
```

ASDU type identifier in monitor and control direction.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

9.1.11 E_IEC870_5_102COTType

```
TYPE E_IEC870_5_102COTType:
(
    eIEC870_2COT_0      := 0,
    eIEC870_2COT_1      := 1,
    eIEC870_2COT_2      := 2,
    eIEC870_2COT_SPONTAN    := 3, (* spontanous *)
    eIEC870_2COT_INIT       := 4, (* initialization *)
    eIEC870_2COT_REQ        := 5, (* request or requested *)
```

```
    eIEC870_2COT_ACT        := 6, (* command activation *)
    eIEC870_2COT_ACT_CON    := 7, (* command activation confirmation *)
    eIEC870_2COT_DEACT      := 8, (* command deactivation *)
    eIEC870_2COT_DEACT_CON  := 9, (* command deactivation confirmation *)
    eIEC870_2COT_ACT_TERM   := 10, (* command activation termination *)
    eIEC870_2COT_11         := 11,
    eIEC870_2COT_12         := 12,
    eIEC870_2COT_RECORD_NOT_FOUND := 13, (* requested record list is not available *)
    eIEC870_2COT_UNKNOWN_ASDU_TYPE := 14, (* unknown ASDU type idetifier *)
    eIEC870_2COT_UNKNOWN_RECORD_NUMBER := 15, (* uknown record number *)
    eIEC870_2COT_UNKNOWN_RECORD_ADDRESS := 16, (* unknown record address *)
    eIEC870_2COT_OBJECT_NOT_FOUND := 17, (* information object not available *)
    eIEC870_2COT_PERIOD_NOT_AVAILABLE := 18, (* requested measurement period not available *)
    eIEC870_2COT_19 := 19,
    eIEC870_2COT_20 := 20,
    eIEC870_2COT_21 := 21,
    eIEC870_2COT_22 := 22,
    eIEC870_2COT_23 := 23,
    eIEC870_2COT_24 := 24,
    eIEC870_2COT_25 := 25,
    eIEC870_2COT_26 := 26,
    eIEC870_2COT_27 := 27,
    eIEC870_2COT_28 := 28,
    eIEC870_2COT_29 := 29,
    eIEC870_2COT_30 := 30,
    eIEC870_2COT_31 := 31,
    eIEC870_2COT_32 := 32,
    eIEC870_2COT_33 := 33,
    eIEC870_2COT_34 := 34,
    eIEC870_2COT_35 := 35,
    eIEC870_2COT_36 := 36,
    eIEC870_2COT_37 := 37,
    eIEC870_2COT_38 := 38,
    eIEC870_2COT_39 := 39,
    eIEC870_2COT_40 := 40,
    eIEC870_2COT_41 := 41,
    eIEC870_2COT_42 := 42,
    eIEC870_2COT_43 := 43,
    eIEC870_2COT_44 := 44,
    eIEC870_2COT_45 := 45,
    eIEC870_2COT_46 := 46,
    eIEC870_2COT_47 := 47,
    eIEC870_2COT_48 := 48,
    eIEC870_2COT_49 := 49,
    eIEC870_2COT_50 := 50,
    eIEC870_2COT_51 := 51,
    eIEC870_2COT_52 := 52,
    eIEC870_2COT_53 := 53,
    eIEC870_2COT_54 := 54,
    eIEC870_2COT_55 := 55,
    eIEC870_2COT_56 := 56,
    eIEC870_2COT_57 := 57,
    eIEC870_2COT_58 := 58,
    eIEC870_2COT_59 := 59,
    eIEC870_2COT_60 := 60,
    eIEC870_2COT_61 := 61,
    eIEC870_2COT_62 := 62,
    eIEC870_2COT_63 := 63
);
END_TYPE
```

Cause of transfer (in monitoring and/or control direction) according to IEC 60870-5-102 norm.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

9.1.12 T_CP40Time2a

5 octets binary time format.

```
TYPE T_CP40Time2a :
STRUCT
    IVTisMinute     : BYTE := 16#00; (* Bit 7 = IV (1=invalid time, 0=valid time), Bit 6 = Tarif inf
ormation, Bit 0..5 = Minutes <0..59> *)
    SURes1Hour      : BYTE := 16#00; (* Bit 7 = SU (1=summer time, 0=normal time), Bits 5..6 = spare
 bits <0>, Bits 0..4 = Hours <0..23> *)
 DOWDay      : BYTE := 16#01; (* Bits 5..7 = Day of week <0 = not used, 1..7>, Bits 0..4 = Day of mo
nth <1..31>*)
    PtiEtiMonth     : BYTE := 16#01; (* Bits 6..7 = Power tarif information <0..3>, Bits 4..5 Energi
e tarif info <0..3>, Bits 0..3 = Month <1..12> *)
    Res2Year    : BYTE := IEC870_DEFAULT_CP56TIME2A_YEAR; (* Bit 7 = spare bit <0>, Bits 0..6 = Year
 <0..99> *)
END_STRUCT
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

9.1.13 T_CP56Time2b

7 octets binary time format.

```
TYPE T_CP56Time2b :
STRUCT
    MsSecond    : WORD := 16#00; (* Bits 11.15 Seconds <0..59>, Bits 0..9 Milliseconds <0..999> *)
    IVTisMinute     : BYTE := 16#00; (* Bit 7 = IV (1=invalid time, 0=valid time), Bit 6 = Tarif inf
ormation, Bit 0..5 = Minutes <0..59> *)
    SURes1Hour      : BYTE := 16#00; (* Bit 7 = SU (1=summer time, 0=normal time), Bits 5..6 = spare
 bits <0>, Bits 0..4 = Hours <0..23> *)
    DOWDay      : BYTE := 16#01; (* Bits 5..7 = Day of week <0 = not used, 1..7>, Bits 0..4 = Day of
 month <1..31>*)
    PtiEtiMonth     : BYTE := 16#01; (* Bits 6..7 = Power tarif information <0..3>, Bits 4..5 Energi
e tarif info <0..3>, Bits 0..3 = Month <1..12> *)
    Res2Year    : BYTE := IEC870_DEFAULT_CP56TIME2A_YEAR; (* Bit 7 = spare bit <0>, Bits 0..6 = Year
 <0..99> *)
END_STRUCT
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1319	PC or CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib are included automatically)

9.2 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.);
2. Compare the interoperability check list of control station and substation;

3. Check the IO configuration and mapping of PLC variables in TwinCAT System Manager (configuration of serial interface, baudrate, parity, stoppbits settings). Compare them to the parameters set in the substation;
4. Check whether the FB_IEC870_5_102TProtocol [▶ 420]() function block issues an error code. The documentation for the error codes can be found here: Overview of error codes;
5. Check the protocol parameters [▶ 407] that are transferred to the function block (link address, FRAMELength etc.). Compare them to the parameters set in the substation;
6. Check the configured address lengths (instance of ST_IEC870_5_102TBuffer [▶ 423], TX/RX data buffer): ASDU address octet size, max ASDU size. Compare them to the parameters set in the substation;
7. Check the configured data points (type, information object address etc.);
8. Check if the substation issues an error code;
9. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings;

9.3 Examples

Used controlled or controlling station configuration parameters:

- Standard PC interface: **COM1**
- **19200 Baud**
- Link address: **1**
- Link address size: **2 octets**
- Common ASDU address: **1**
- Common ASDU address size: **2 octets**

Requirements

PLC project	TwinCAT System Manager Configuration	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11717569675/.zip	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11717569675/.zip	Simple **Master-Applikation** which uses the TwinCAT IEC60870-5-102 Serial Link Interface.

10 TcIEC870_5_103Link: IEC 60870-5-103 Serial Link Interface (master)

The control stations (masters) according to IEC60870-5-103 can be realised in the TwinCAT PLC with the PLC functions and function blocks.

The end application is imposed on the software interface of the PLC library. Unlike the implementation of other 101/104 TwinCAT products only one "so called" 'Low level' interface is available. The reason: The IEC 60870-5-102 devices are featuring a lot of vendor specific data. The 'Low level' interface allows the access to this data. The characteristics of both interfaces are described briefly below.

'Low level' interface: IEC 60870-5-103 Serial Link Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/ Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

Interoperability check list

for TwinCAT PLC Library: IEC 60870-5-103 control station (master). Here you can https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11763843211/.zip

System requirements

Development environment:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT installation level:TwinCAT PLC or higher;
- TwinCAT System version 2.10.0 Build >= 1313 or higher;

Target system:

- Industrial PC or Embedded PC/CX (x86, ARM);
- Operating system:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86, ARM) (image v2.16 or higher);
- TwinCAT PLC runtime system version 2.10.0 or higher;
- Serial COM port or KL6xxx or EL6xxx bus terminal (**EL6xxx support implemented in product version 1.0.11 and higher**);

Product components

- **TcIEC870_5_103Link.Lib** (implements the transfer procedures for the transport of IEC 60870-5-103 ASDU's); This library have to be included in the PLC project. All other libraries are included automatically.
- TcIEC870_5_101Link.Lib (base library, implements the transfer procedures for the transport of ASDU's via the serial interfaces of PC and Beckhoff KL6xxx/EL6xxx bus terminal);

- TcIEC870_5_101.Lib (implements the connection functions and common data types);
- COMLibV2.Lib (implements the functions for the serial COM- or KL6xxx/EL6xxx communication);

Installation on Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

The PLC libraries are copied into the folder: ..\TwinCAT\Plc\Lib.

Installation on Windows CE

You don't have to install any additional components on the CE platform.

This document is not full product manual. Please install the full version of the Beckhoff Information System.

You will find it

- on the Beckhoff Product DVDs
- on our Web-Server: http://www.beckhoff.com under download.

Examples

Example projects are in the Beckhoff Information System documentation of TwinCAT PLC library.

Link to 'low level' example overview page: IEC 60870-5-103 Serial Link Interface [▶ 442];

Further Documentation

- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 397];
- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
- Documentation of TwinCAT PLC Library: Serial communication;
- IEC 60870-5-1 Transmission frame formats;
- IEC 60870-5-2 Link transmission procedures;
- IEC 60870-5-3 General structure of application data;
- IEC 60870-5-4 Definition and coding of application information elements;
- IEC 60870-5-5 Basic application functions;
- IEC 60870-5-101 Companion Standard for basic telecontrol tasks;
- IEC 60870-5-101:1995/A1:2000 Companion Standard for basic telecontrol tasks. Amendment 1;
- IEC 60870-5-101:1995/A2:2001 Companion Standard for basic telecontrol tasks. Amendment 2;
- IEC 60870-5-103:1997 Transmission protocols - Companion Standard for the informative interface of protection equipment;

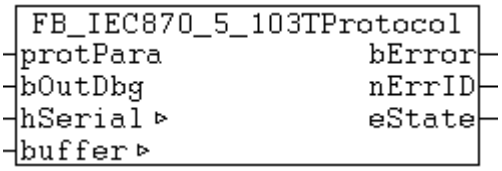
10.1 Introduction

Simple projects with complete sources can be found under: Transport-Interface-Examples.Transport-Interface-Examples [▶ 442]

A PLC application which should communicate with a or sub station via the transport interface needs the following resources:

- Instance of communication function block: FB_IEC870_5_103TProtocol [▶ 434];
- Instance of TX/RX data buffer: ST_IEC870_5_103TBuffer [▶ 437];
- Instance of function block to manipulate the TX/RX data buffer: FB_IEC870_5_103TBufferCtrl [▶ 435];

10.1.1 FB_IEC870_5_103TProtocol



The FB_IEC870_5_103TProtocol communication block implements the transport functions of the IEC 60870-5-1 and IEC 60870-5-2 link layer. In the event of a protocol error an associated error code is issued at the function block output and the data transfer is interrupted. The data exchange can be reactivated by calling the INIT function block action. The communication block expects a TX/RX data buffer variable. This variable must be transferred to the block via VAR_IN_OUT.

The function block features the following actions:

- **INIT** (initialises the function block). In the default configuration the TX/RX data buffer are cleared. The bRetainBuffer member variable of the protPara structure controls the reset bahaviour of TX/RX data buffer.

Protocol configuration

The communication block has a structured protPara variable. Protocol parameters such as timeout times or connection addresses etc. can be configured via this variable.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial     : T_HSERIALCTRL;         (* Serial link control handle *)
    buffer      : ST_IEC870_5_103TBuffer; (* TX/RX data buffer *)
END_VAR
```

hSerial: Connection handle [▶ 415] to FB_IEC870_SerialLineCtrl [▶ 404] function block. The data to be sent and received are excahged with the FB_IEC870_SerialLineCtrl function block.

buffer: TX/RX data buffer [▶ 437].

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_101ProtocolParams; (* IEC60870-5-101/103 protocol parameters *)
    bOutDbg     : BOOL;        (* Enable/disable debug output *)
END_VAR
```

protPara: IEC60870-5-101/103 protocol parameters [▶ 407].

bOutDbg:: Activates/deactivates the debug output of the frames in the TwinCAT System Manager logger view.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED; (* Data transfer state
*)
ND_VAR
```

bError: This output is switched to TRUE if an error occurs during data transfer.

nErrID: Returns an error code if the bError output is set.

eState: Status [▶ 413] of the data exchange to the master.

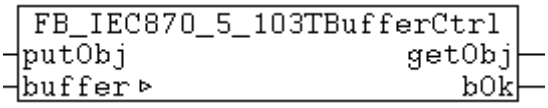
Example:

Sample projects: IEC60870-5-103 Serial Link Interface [▶ 442]

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.2 FB_IEC870_5_103TBufferCtrl



This function block changes the content of the TX/RX data buffer. This data buffer ist used by the communication via the IEC60870-5-103 Serial Link Interface.

The function block has the following actions:

- **RxRemoveObj** (removes the oldest Fifo entry from the RX-Fifo);
- **RxReset** (deletes all RX-Fifo entries, resets the RX-Fifo);
- **TxAddObj** (inserts a new Fifo entry in the TX-Fifo);
- **TxReset** (deletes all TX-Fifo entries, resets the TX-Fifo);

The content of the TX/RX data buffer can be changed by the call of one of the listed actions above.

VAR_IN_OUT

```
VAR_IN_OUT
    buffer  : ST_IEC870_5_103TBuffer; (* TX/RX data buffer *)
END_VAR
```

buffer: TX/RX data buffer [▶ 437]. The TX/RX buffer parameters (like e.g. asduSize) have to be configured before using.

VAR_INPUT

```
VAR_INPUT
    putObj  : ST_IEC870_5_103AOGen; (* ASDU to send *)
END_VAR
```

putObj: Data unit [▶ 438] (ASDU) to be sent.

VAR_OUTPUT

```
VAR_OUTPUT
    getObj  : ST_IEC870_5_103AOGen; (* received ASDU *)
 bOk     : BOOL; (* TRUE = action succesfully, FALSE=Fifo overflow or fifo empty *)
END_VAR
```

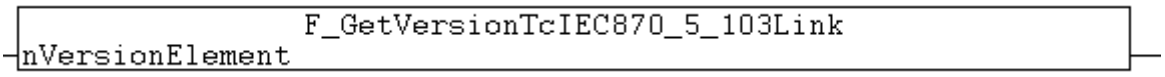
getObj: Received data unit [▶ 438] (ASDU).

bOk: This variable becomes TRUE, if a new entry was inserted or removed successfully from the Fifo. This variable becomes FALSE at a buffer overflow and if no entry could be removed, because the Fifo was already empty.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.3 F_GetVersionTcIEC870_5_103Link



This function reads version information from the plc library.

FUNCTION F_GetVersionTcIEC870_5_103Link: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

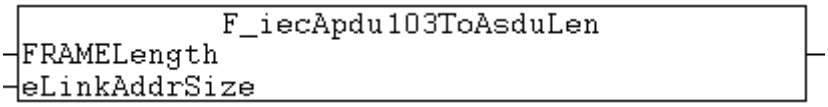
nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.4 F_iecApdu103ToAsduLen



The function calculates for the IEC 60870-5-103 protocol the maximum available ASDU byte length with the configurated frame and link address field length. The maximum available ASDU length is for example required for the configuration of the ST_IEC870_5_103T buffer variable. This data structure (TX/RX data buffer) is used for the data exchange via the IEC60870-5-103 Serial Link Interface.

FUNCTION F_iecApdu103ToAsduLen: BYTE

```
VAR_INPUT
    FRAMELength     : BYTE;
    eLinkAddrSize   : E_IEC870_5_101LinkAddrSize;
END_VAR
```

FRAMELength : Max. available APDU frame length (see compatibility list).

eLinkAddrSize: Link address field octet length [▶ 332] (see compatibility list).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.5 ST_IEC870_5_103TBuffer

This data structure is used for data exchange (TX/RX data buffer) via the IEC60870-5-103 Serial Link Interface.

```
TYPE ST_IEC870_5_103TBuffer :
STRUCT
    eDbg     : E_IEC870_5_101FifoDbgFlags :=eIEC870_FIFO_DBG_OFF; (* enable/
disable log view hex output *)
    asduSize : BYTE := 249; (* max. length of ASDU data *)
    mode     : DWORD := 0; dataLink : ST_IEC870_5_101DataLink; (* internal tx/rx buffer *)
ND_STRUCT
END_TYPE
```

eDbg: Debug output parameter [▶ 335].

asduSize: Common asdu address octet size.

mode: Reserved, not used. This value must be zero.

dataLink: Internal Tx/Rx buffer. The access to the elements of this data structure should only occur with an instance of the function block FB_IEC870_5_103TBufferCtrl [▶ 435], not directly.

The TX/RX data buffer uses internal two send Fifos and one receive Fifo:

1. Class 1 send Fifo with (high prior) data;
2. Class 2 send Fifo with (low prior) data;
3. Receive Fifo (for Class 1 and Class 2 data);

The lower transport functions of the library emtpy first the class 1 Fifo and afterwards the class 2 Fifo. The class 2 data are only sent if the class 1 Fifo contains no data to be sent.

Each internal Fifo hast a fixed size of 10000 bytes. This is enough for the most applications, because the number of frames which can once be sent is limited by the iK and iW protocol parameters.
According to experience it is possible to store 200 ADSUs with an information element (object), or approx. 20 ADSUs with with a sequence of 100 information elements (objects) in each Fifo.

If a higher number of frames (to be sent or to be received) should be buffered (e.g. ~20000), they can be kept in external buffers/Fifos which are defined by the PLC programmer. The PLC application can refill the TwinCAT send Fifos with own Fifo entries, or empty the TwinCAT receive Fifo.

As an alternative it is possible to use two buffers and to fill/read them alternately and give them to the communication block .

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib;

Development Environment	Target System	PLC libraries to include
		TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.6 ST_IEC870_5_103AOGen

Variables of this type represent an ASDU object.

```
TYPE ST_IEC870_5_103AOGen:
STRUCT
    head    : ST_IEC870_5_101FifoHead :=(      source := ( link := 0, addr := 0 ),
                  target := ( link := 0, addr := 0 ),
                  ctrl := 0 ); (* Header *)

    ident   : ST_IEC870_5_103DataUnit_Ident := ( eType     := 0,
                        nObj     := 0,
                        bSQ      := FALSE,
                        eCOT     := 0,
                        eClass   := eIEC870_Class_None,
                        asduAddr     := 0 ); (* Data unit identifier *)

    info    : ST_IEC870_5_103AOInfoObj  := (    fc       := 0,
                        n        := 0,
                        stream   := ( length := 0 ) ); (* information object *)
END_STRUCT
END_TYPE
```

head: Header (reserved).

ident: Identification field in the data unit (ASDU).

info: Information object /Information element data field.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

Also see about this

- ST_IEC870_5_103DataUnit_Ident [▶ 439]
- ST_IEC870_5_103AOInfoObj [▶ 438]

10.1.7 ST_IEC870_5_103AOInfoObj

Information object description.

```
TYPE ST_IEC870_5_103AOInfoObj :
STRUCT
    fc      : BYTE;
    n       : BYTE;
    stream  : ST_IEC870_5_101Stream; (* Information element of max. length *)
END_STRUCT
END_TYPE
```

fc: Function number (code).

n: Information number.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.8 ST_IEC870_5_103DataUnit_Ident

Identification field of data unit (ASDU).

```
TYPE ST_IEC870_5_103DataUnit_Ident:
STRUCT
    eType   : INT(0..255);    (* ASDU type identifier *)
    bSQ     : BOOL;       (* Single/Sequence *)
    nObj    : BYTE(0..127);   (* Number of information objects or elements *)
    eCOT    : INT(0..255);    (* Cause of transmission *)
    asduAddr: BYTE;       (* Common address of ASDU *)
    eClass  : E_IEC870_5_101ClassType; (* Object class *)
END_STRUCT
END_TYPE
```

eType: ASDU type identifier (0..255).

bSQ: Sequence flag.

nObj: Number of information objects or information elements (0..127).

eCOT: Cause of transmission (0..255).

asduAddr: Common ASDU address (one octet).

eClass: Priority class [▶ 333].

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.9 E_IEC870_5_103MTypeID

```
TYPE E_IEC870_5_103MTypeID:
(
    M_TYPEUNDEF_3 := 0, (* Not used *)
    M_TTM_TA_3 := 1, (* Time-tagged message *)
    M_TMR_TA_3 := 2, (* Time-tagged message with relative time *)
    M_MEI_NA_3 := 3, (* Measurands I *)
    M_TME_TA_3 := 4, (* Time-tagged measurands with relative time *)
    M_IRC_NA_3 := 5, (* Identification *)
    M_SYN_TA_3 := 6, (* Time synchronisation *)
    M_TGI_NA_3 := 8, (* General interrogation *)
    M_MEII_NA_3 := 9, (* Measurands II *)
    M_GD_XA_3 := 10,(* Generic data *)
    M_GI_XA_3 := 11,(* Generic identification *)
    M_LRD_TA_3 := 23,(* List of recorded disturbances *)
    M_RTD_TA_3 := 26,(* Ready for transmission of disturbance data *)
    M_RTC_NA_3 := 27,(* Ready for transmission of channel *)
    M_RTT_NA_3 := 28,(* Ready for transmission of tags *)
```

```
    M_TOT_NA_3 := 29,(* Transmission of tags *)
    M_TOV_NA_3 := 30,(* Transmission of disturbance values *)
    M_EOT_NA_3 := 31 (* End of transmission *)
);
END_TYPE
```

ASDU type identifier in monitoring direction (slave -> master).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.10 E_IEC870_5_103CTypeID

```
TYPE E_IEC870_5_103CTypeID:
(
    C_TYPEUNDEF_3 := 0, (* Not used *)
    C_SYN_TA_3 := 6, (* Time synchronisation *)
    C_IGI_NA_3 := 7, (* General interrogation *)
    C_GD_NA_3 := 10,(* Generic data *)
    C_GRC_NA_3 := 20,(* General command *)
    C_GC_NA_3 := 21,(* Generic command *)
    C_ODT_NA_3 := 24,(* Order for disturbance data transmission *)
    C_ADT_NA_3 := 25 (* Acknowledgement for disturbance data transmission *)
);
END_TYPE
```

ASDU type identifier in control direction (master -> slave).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.11 E_IEC870_5_103MCOT

```
TYPE E_IEC870_5_103MCOT:
(
 eIEC870_MCOT_UNUSED := 0, (* Not used *)
 eIEC870_MCOT_SPONTAN := 1, (* Spontanous data *)
 eIEC870_MCOT_CYCLIC := 2, (* Cyclic data *)
 eIEC870_MCOT_FCB := 3, (* Reset FCB bit *)
 eIEC870_MCOT_CU := 4, (* Reset communication unit *)
 eIEC870_MCOT_SR := 5, (* Start/Restart *)
 eIEC870_MCOT_ON := 6, (* Power on *)
 eIEC870_MCOT_TST := 7, (* Test mode *)
 eIEC870_MCOT_SYN := 8, (* Time synchronisation *)
 eIEC870_MCOT_GI := 9, (* General interrogation *)
 eIEC870_MCOT_TGI := 10,(* Termination of general interrogation *)
 eIEC870_MCOT_LO := 11,(* Local operation *)
 eIEC870_MCOT_RO := 12,(* Remote operation *)
 eIEC870_MCOT_CP := 20,(* Positive ack of command *)
 eIEC870_MCOT_CN := 21,(* Negative ack of command *)
 eIEC870_MCOT_TOV := 31,(* Transmission of disturbance values *)
 eIEC870_MCOT_WP := 40,(* Positive ack of generic write command *)
 eIEC870_MCOT_WN := 41,(* Negative ack of generic write command *)
```

```
 eIEC870_MCOT_RP := 42,(* Valid data response to genric read command *)
 eIEC870_MCOT_RN := 43,(* Invalid data response to generic read command *)
 eIEC870_MCOT_CWC := 44 (* Confirmation of generic write *)
);
END_TYPE
```

Cause of transfer in monitoring direction (slave -> master).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.1.12 E_IEC870_5_103CCOT

```
TYPE E_IEC870_5_103CCOT:
(
 eIEC870_CCOT_UNUSED := 0, (* Not used *)
 eIEC870_CCOT_SYN := 8, (* Time synchronisation *)
 eIEC870_CCOT_IGI := 9, (* Initialisation of general interrogation *)
 eIEC870_CCOT_GRC := 20, (* General command *)
 eIEC870_CCOT_TOV := 31, (* Transmission of disturbance values *)
 eIEC870_CCOT_WC := 40, (* Generic write command *)
 eIEC870_CCOT_RC := 42 (* Generic read command *)
);
END_TYPE
```

Cause of transfer in control direction (master -> slave).

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_103Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMlibV2.Lib; are included automatically)

10.2 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.);
2. Compare the interoperability check list of control station and substation;
3. Check the IO configuration and mapping of PLC variables in TwinCAT System Manager (configuration of serial interface, baudrate, parity, stoppbits settings). Compare them to the parameters set in the substation;
4. Check whether the FB_IEC870_5_103TProtocol [▶ 434]() function block issues an error code. The documentation for the error codes can be found here: Overview of error codes;
5. Check the protocol parameters [▶ 407] that are transferred to the function block (link address, link address octet size, FRAMELength etc.). Compare them to the parameters set in the substation;
6. Check the max. configured ASDU length (instance of ST_IEC870_5_103TBuffer [▶ 437], TX/RX data buffer). Compare them to the parameters set in the substation.
7. Check the configured data points (type, ASDU address, function number, information number etc.);

8. Check if the other communication partner issues an error code;
9. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings;

10.3 Examples

Used controlled or controlling station configuration parameters:

- Standard PC interface: **COM1**
- **19200 Baud**
- Link address: **220**
- Link address size: **1 octet (fix)**
- Common ASDU address: **220**
- Common ASDU address size: **1 octet (fix)**

Requirements

PLC project	TwinCAT System Manager Configuration	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11724892683/.zip	https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11724892683/.zip	Simple **Master-Applikation** which uses the TwinCAT IEC60870-5-103 Serial Link Interface.

11 TcIEC870_5_104: IEC 60870-5-104 Transport Interface (master/slave

The library **TcIEC870_5_104.Lib** implements an interface. Via this interface single ASDUs (service data unitof the application layer) can be sent and received. This interface is placed inside the protocol structure above the transport layer (4) and implements already the APCI functions (protocol control information of the application layer, see lower table). Application functions like for example the general request and count request are not implemented in the interface, but the user can implement them by himself.

'Low level' interface: IEC 60870-5-104 Transport Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/ Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

This interface is recommended if you:

- are familiar with the protocol standard;
- are implementing a protocol converter application;
- are implementing virtually all available standard functions in the application;
- are using special functions, such as the relaying of the time stamp from a Modbus device or the gaining of control over the command execution;
- require functions that are not supported according to the compatibility list;
- have many data points (>1000) and need high performance;

Protocol structure of endsystem:

Requirements

Selection of application functions from IEC 60870-5-5 to IEC 60870-5-101	Initialisation	Application process
Selection of ASDU (service data unit of application layer) from IEC 60870-5-101 and IEC 60870-5-104		Application layer (7)
TwinCAT PLC Library: IEC 60870-5-104 (low level) Transport Interface		
APCI (Protocol control information of the application layer) Transport Interface (user to TCP interface)		
Selection of TCP/IP protocol collection (RFC 2200)		Transport layer (4) Relaying layer (3) Connection layer (2) Physical layer (1)

Remark: The layers 5 and 6 are not used.

11.1 Introduction

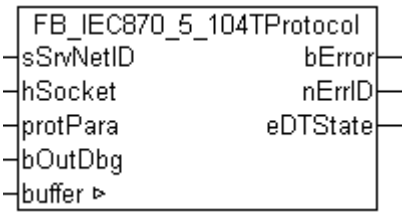
Simple projects with complete sources can be found under: Transport-Interface Examples.Transport-Interface Examples [▶ 450]

A PLC application which should communicate with a control station or substation via the transport interface needs the following resources:

- Instance of communication function block: FB_IEC870_5_104TProtocol [▶ 444];
- Instance of function block to create a TCP/IP connection: FB_ServerClientConnection or FB_ClientServerConnection;
- Instance of TX/RX data buffer: ST_IEC870_5_101TBuffer;
- Instance of function block to manipulate the TX/RX data buffer: FB_IEC870_5_101TBufferCtrl;

11.1.1 FB_IEC870_5_104TProtocol

From product version TwinCAT PLC Library IEC870-5-104 Substation v2.0.6 and higher.



The FB_IEC870_5_104TProtocol communication block implements the ACPI functions of the IEC60870-5-104-standard (start/stop data transfer, test frames, send/receive frame counter etc.). In the event of a protocol error an associated error code is issued at the function block output and the data transfer is interrupted. The data exchange can be reactivated by calling the INIT task. The frame counters and the send and TX/RX data buffers are reset. The communication block expects a TX/RX data buffer variable. This variable must be transferred to the block via VAR_IN_OUT.

The function block features the following tasks:

- **INIT** (initialises the function block);
- **STARTDT** (sends a start data transfer frame to the communication partner);
- **STOPDT** (sends a stop data transfer frame to the communication partner);

Establishing the connection

The TCP/IP connection has to be established and closed via a separate block, e.g. FB_ServerClientConnection. The PLC application can respond independently to possible protocol errors and close the connection or implement the process reset service, for example. At the output this block provides a connection handle, the connection status and information about errors that have occurred during establishment/closing of the connection.
The connection handle is required by the communication block.

Protocol configuration

The communication block has a structured protPara variable. Protocol parameters such as iK, iW, start/stop data transfer behaviour etc. can be configured via this variable.

VAR_IN_OUT

```
VAR_IN_OUT
    buffer      : ST_IEC870_5_101TBuffer; (* TX/RX data buffer *)
END_VAR
```

buffer: TX/RX data buffer.

VAR_INPUT

```
VAR_INPUT
    sSrvNetID       : T_AmsNetID;     (* TwinCAT TCP/IP Connection Server netID *)
    hSocket         : T_HSOCKET;      (* TCP/IP socket connection handle *)
    protPara        : ST_IEC870_5_104ProtocolParams; (* IEC60870-5-104 protocol parameters *)
    bOutDbg         : BOOL;       (* Enable/disable debug output *)
END_VAR
```

sSrvNetID: String containing the network address of the TwinCAT TCP/IP Connection Server. For the local computer (default) an empty string may be specified.

hSocket: TCP/IP connection handle of the communication partner for receiving or sending data.

protPara: IEC60870-5-104 protocol parameters [▶ 446]

bOutDbg:: Activates/deactivates the debug output of the TCP/IP frames in the TwinCAT System Manager logger view.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_104DataTransferState := eIEC870_STOPDT; (* Data transfer state *)
ND_VAR
```

bError: This output is switched to TRUE if an error occurs during data transfer.

nErrID: Returns an error code if the bError output is set;

eDTState: status [▶ 449] of the IEC60870-5-104 data exchange (STARTDT, STOPDT)

Example:

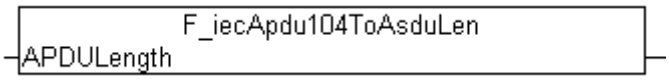
Sample projects: IEC60870-5-104 Transport Interface [▶ 450]

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib are included automatically)

11.1.2 F_iecApdu104ToAsduLen

From product version: TwinCAT PLC Library IEC60870-5-104 Substation v2.0.6 and higher.



The function calculates for the IEC 60870-5-104 protocol the maximum available ASDU byte length with the configurated APDU length. The maximum available ASDU length is for example required for the configuration of the ST_IEC870_5_101T buffer variable. This data structure (TX/RX data buffer) is used for the data exchange via the IEC60870-5-104 Transport Interface.

FUNCTION F_iecApdu104ToAsduLen: BYTE

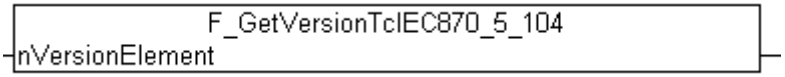
```
VAR_INPUT
    APDULength : BYTE;
END_VAR
```

APDULength : Available APDU length (see compatibility list).

Requirements

Development Environment	Target System Type	PLC Libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

11.1.3 F_GetVersionTcIEC870_5_104



This function reads version information from the PLC library.

FUNCTION F_GetVersionTcIEC870_5_104: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Libare included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

11.1.4 ST_IEC870_5_104ProtocolParams

```
TYPE ST_IEC870_5_104ProtocolParams :
STRUCT
    t0              : TIME := T#30s; (* MASTER only: Connection establishment *)
    t1              : TIME := T#15s; (* Response Timeout (STARTDTAct, STOPDTAct, TESTFRAct only ) *)
    t2              : TIME := T#10s; (* Send ACK in case of no data frames *)
    t3              : TIME := T#20s; (* Time for sending test frames *)
    iK              : INT := 12; (* Max. difference RSN to send ACK *)
    iW              : INT := 8; (* Latest ACK after receiving I-frames *)

    bSFrameACK      : BOOL := TRUE; (* Send S-Frame ACK *)
    bTESTFRAct      : BOOL := TRUE; (* Send TESTFR *)
    bDTControlled   : BOOL := TRUE; (* STARTDT, STOPDT controlled (wait for STARTDT, STOPDT) *)
    bControlDT      : BOOL := FALSE; (* Send START, STOPDT frames *)

    bSTARTDTCon     : BOOL := TRUE; (* Send STARTDT confirmations *)
    bSTOPDTCon      : BOOL := TRUE; (* Send STOPDT confirmations *)
```

```
    bTESTFRCon      : BOOL := TRUE; (* Send TESTFR confirmations *)

    eAcceptMode     : E_SocketAcceptMode := eACCEPT_ALL; (* Connection accept flags *)
    sRemoteHost     : STRING(15) := ''; (* Remote (server) address. String containing an (Ipv4) Inte
rnet Protocol dotted address. *)
    nRemotePort     : UDINT := 0; (* Remote (server) Internet Protocol (IP) port. *)

    APDULength      : BYTE(MIN_IEC870_5_104_APDULEN..MAX_IEC870_5_104_APDULEN) := MAX_IEC870_5_104_A
PDULEN;(* Defaults:
               Max. length of APDU = 255 bytes - 1 start octet - 1 length octet = 253 octets,
                Max. length of ASDU = 253 - 4 control octets = 249 octets*)
    bThrottleMode   : BOOL := FALSE; (* If set reduces the number of polling socket read requests *)
    bPackFrames     : BOOL := FALSE; (* Binds more than one APDU frames to one big TCP/IP frame *)
    bRetainBuffer   : BOOL := FALSE; (* TRUE => Don't reset the tx/
rx buffer in offline mode, FALSE => reset tx/rx buffer in offline mode *)

    bCOTFilter      : BOOL := TRUE; (* Only used if bRetainBuffer = TRUE. If TRUE = filter asdu by C
OT and session ID, FALSE = don't filter the asdu's *)
    cotFilter       : T_IEC870_5_101COTBits := 2#10001111, 7(16#FF);
(* COT (cause of transfer) filter, default: eIEC870_COT_CYCLIC or eIEC870_COT_BACKGROUND or eIEC870_
COT_SPONTAN *)
END_STRUCT
END_TYPE
```

t0 : Not used, reserved.

t1 : Max. Timeout time for an answer to a STARTDTAct-, STOPDTAct- or TESTFRAct-Frame.

t2 : Latest time to send a S- Frame.

t3 : Latest time to send a Test- Frame.

iK : Latest after these sent APDU's in I-Format, without acknowledgement, the connection is to be closed.

iW : Latest acknowledgement after receiving of APDU's in I-Format.

bSFrameACK : Send S- Frame.

bTESTFRAct : Send Test-Frames.

bDTControlled : Wait for STARTDT-, STOPDT-Frame from master.

bControlDT : Not used, reserved.

bSTARTDTCon : Send STARTDT confirmation.

bSTOPDTCon : Send STOPDT confirmation.

bTESTFRCon : Send TESTFR confirmation.

eAcceptMode : Defines if connections to all Remote Clients, or only to clients with specified Host and Port address are allowed. Default: accept all arriving connections.

sRemoteHost : Host address of Remote-Client. If eAcceptMode = eACCEPT_ALL this parameter will be ignored.

nRemotePort : Port address of Remote-Client. If eAcceptMode = eACCEPT_ALL this parameter will be ignored.

APDULength : Max. length of APDU.

bThrottleMode: Implemented from IEC870-5-104 slave library **v2.0.0** and higher. The TCP/IP sockets are polled by the PLC. This parameter reduces the number of polled read accesses and so the system load. This makes sense for infrequent data receiving (e.g. General query or time synchronisation query).

- FALSE: Each read accesss is followed by the next. Indenpendent from receiving new data (higher system load);
- TRUE: After each read access without new data a delay is inserted. The following read access will be done delayed (smaller system load).
  After each read access with new data, the next read access will be done without delay.

bPackFrames: Implemented from IEC870-5-104 slave library **v2.0.4** and higher. By default a TCP/IP send call only sends a single APDU. The send performance can be increased substantially (thereby reducing send buffer overflows) by setting this parameter to TRUE.

- FALSE: For an APDU a TCP/IP send call is required. A maximum of 1 APDU can be sent about every 3 PLC cycles;
- TRUE: Several APDUs are consolidated to form a larger TCP/IP block (no more than iK frames) and sent via a TCP/IP send call;

bRetainBuffer: Implemented from IEC870-5-104 slave library **v3.0.14** and higher. In the standard setting (FALSE) the internal Tc/Rx buffers are deleted when the connection is severed. If this flag is set to TRUE, the ASDUs in the internal send buffer that have not yet been sent are not deleted. Hence, offline buffering of approx. 100-200 measured values (dependent on the ASDU size) in the RAM memory is possible. The station then always removes the ASDUs from the send buffer once their receipt has been confirmed. Please note that the ASDUs that have already been sent but not yet confirmed also remain in the buffer and are sent again the next time. The other station may then receive the values twice. You can configure this behaviour via two further parameters: **bCOTFilter** and **cotFilter**.

bCOTFilter: Implemented from IEC870-5-104 slave library **v3.0.14** and higher. Activates/deactivates the filter mask with the causes of transfer. This parameter is only valid when the *bRetainBuffer* parameter is also set to TRUE. A SessionID is incremented internally with each establishment of a new connection. This SessionID is always attached to the received and sent ASDUs. Hence, the ASDUs that have not yet been sent and remain in the offline buffer are assigned to the old connection. These ASDUs can then be filtered and discarded with the aid of the COT mask (**cotFilter**) (COT = Cause Of Transfer). This is sometimes necessary if the other communication partner does not accept the repeated ASDUs.

cotFilter: Implemented from IEC870-5-104 slave library **v3.0.14** and higher. Filter mask with causes of transfer (COT = Cause Of Transfer). This parameter is only valid when the *bRetainBuffer* and *bCOTFilter* parameters have also been set. The cause of transfer of the ASDUs to be sent is only checked if their SessionID does not correlate to the current SessionID (i.e. the ASDUs originating from the previous connection). Each bit corresponds to a cause of transfer. The cause of transfer is only checked if the appropriate bit has been set.

The causes of transfer are coded in the following way in the bits:

cotFilter[0].7 = eIEC870_COT_UNUSED
cotFilter[0].6 = eIEC870_COT_CYCLIC
cotFilter[0].5 = eIEC870_COT_BACKGROUND
cotFilter[0].4 = eIEC870_COT_SPONTAN
cotFilter[0].3 = eIEC870_COT_INIT
cotFilter[0].2 = eIEC870_COT_REQ
cotFilter[0].1 = eIEC870_COT_ACT
cotFilter[0].0 = eIEC870_COT_ACT_CON
cotFilter[1].7 = eIEC870_COT_DEACT
cotFilter[1].6 = eIEC870_COT_DEACT_CON
cotFilter[1].5 = eIEC870_COT_ACT_TERM

... and so on.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

11.1.5 E_IEC870_5_104DataTransferState

```
TYPE E_IEC870_5_104DataTransferState :
(
    eIEC870_STOPDT, (* data exchange deactivated *)
    eIEC870_STARTDT, (* data exchange activated *)
    eIEC870_STOPDT_PENDING, (* waiting for STOPDT confirmation (reserved)*)
    eIEC870_STARTDT_PENDING (* waiting for STARTDT confirmation (reserved)*)
);
END_TYPE
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

11.2 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.).

2. Check the software installation hints described in this documentation (e.g. installation of CAB files on CE platform).

3. In the event of connection problems the PING command can be used to ascertain whether the external communication partner can be reached via the network connection. If this is not the case, check the network configuration and firewall settings [▶ 509].

4. Check the network address and port number parameter that are transferred to the function F_CreateServerHnd() or to the FB_ClientServerConnection()/FB_ServerClientConnection() function block for correctness.

5. Check whether the function block issues an error code. The documentation for the error codes can be found here: Overview of error codes.

6. Check the protocol parameters [▶ 446] that are transferred to the function block (iK, iW, t0, t1, t2, t3, APDULength etc.). Compare them to the parameters set in the communication partner.

7. Check the configured address lengths (instance of ST_IEC870_5_101TBuffer, TX/RX data buffer): ASDU address octet size, information object address octet size, cause of transfer octet size, max ASDU size. Compare them to the parameters set in the communication partner.

8. Check the data point configuration (type, information object addresses etc.).

9. Check if the communication partner issues an error code.

10. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings.

11. Check the usage of FB_SocketCloseAll() function block and the usage of the LISTEN_MODE_CLOSEALL parameter in your PLC application.

If your application is working with more than one TCP/IP connections (server/clients) than you have to use one instance of FB_SocketCloseAll() function block to close old/opened connections. Activate this function block instance only once at program start and don't use the LISTEN_MODE_CLOSEALL parameter.

12. Sniffer tools such as Wireshark enable logging of the entire network communication. The log can then be analyzed by Beckhoff support staff.

11.3 Examples

Used controlled or control station parameters:

- Server host address: **127.0.0.1** (you have to adapt at least this parameter to your target system!!)
- Server port addres: **2404**
- k: **12**
- w: **8**
- t0: **30s**
- t1: **15s**
- t2: **10s**
- t3: **20s**
- Cause of transfer size: **2 octets**
- Common ASDU address size: **2 octets**
- Information object address size: **3 octets**
- Originator address: **1**
- Common ASDU address: **7**
- Max. APDU length: **253**

PLC project	Documentation	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11729046539/.zip		Simple **Slave-Application** which uses the TwinCAT IEC60870-5-104 Transport Interface. MAIN implements the link connection/ disconnection and simple TX/RX- data exchange: • Data points in monitoring direction: M_SP_NA_1(IOA:=100) and M_BO_TB_1(IOA:=400). Data points in control direction: C_SC_NA_1(IOA:=10); • Simple implementation of clock synchronisation command: C_CS_NA_1(IOA:=0); • Simple implementation of general interrogation command: C_IC_NA_1(IOA:=0); • Simple implementation of single command: C_SC_NA_1(IOA:=10); • Simple implementation of counter interrogation command: C_CI_NA_1(IOA:=0);
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11729047947/.zip		Simple **Master-Applikation** which uses the TwinCAT IEC60870-5-104 Transport Interface. MAIN implements the link connection/ disconnection and simple TX/RX- data exchange: • Simple general interrogation is send every 20 seconds: C_IC_NA_1(IOA:=0); • Rising edge at variable *bSCmd* starts simple single command execution: C_SC_NA_1(IOA:=10); • Rising edge at variable *bClockSyn* starts simple clock synchronisation command execution: C_CS_NA_1(IOA:=0);

12 TcIEC870_5_104Slave: IEC 60870-5-104 Controlled Station (slave)

Substations (Slaves) according to IEC60870-5-104 can be realised in TwinCAT PLC with PLC functions and function blocks.

The PLC library has two software interfaces. The end application is imposed on one of these interfaces. The choice of interface depends on the requirements for the end application. The characteristics of both interfaces are described briefly below.

'High level' interface: IEC 60870-5-104 Controlled Station

This interface is a so-called ‘single-block solution’. All functions are encapsulated in one PLC block. The block implements the most important services and functions. This implementation is sufficient for over 90% of applications.

Pro: Very little PLC programming work is required in order to create an application; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. are already implemented in the block and are executed automatically; the mapping of the IEC<->PLC process data and that of the data points is configured via function calls; the PLC programmer does not need to be very well acquainted with the protocol standard;

Contra: The PLC application has only a small influence on the execution of the protocol; no influence on the execution of the services – these are automatically implemented internally; time stamps are automatically generated by the block and cannot be changed (handed over by externals); only the direct command execution, for example, is possible; poorer performance if there are many data points.

This interface is recommended if you:

- are not familiar with the protocol standard;
- are implementing a simple application with few data points (<1000);
- are not placing any great performance demands on the application;
- are not sending any special command execution such as Select/Execute or data + time stamp from external devices;
- do not require any functions that are not supported according to the compatibility list;

'Low level' interface: IEC 60870-5-104 Transport Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronization, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

This interface is recommended if you:

- are familiar with the protocol standard;
- are implementing a protocol converter application;
- are implementing virtually all available standard functions in the application;

- are using special functions, such as the relaying of the time stamp from a Modbus device or the gaining of control over the command execution;
- require functions that are not supported according to the compatibility list;
- have many data points (>1000) and need high performance;

Interoperability check list

for TwinCAT PLC Library: IEC 60870-5-104 controlled station (relating to the "high level" interface). Here you can https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11764746379/.zip

System requirements

Development environment:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT installation level:TwinCAT PLC or higher;
- TwinCAT System version 2.9.0 Build >= 1030 (CX (ARM) Build >= 1301) or higher;

Target system:

- Industrial PC or Embedded PC/CX (x86, ARM);
- Operating system:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86) (image v1.75 or higher);
  - Windows CE (ARM): CE (image v2.13 or higher);
- TwinCAT PLC runtime system version 2.9.0 or higher;

Product components

- **TcIEC870_5_104Slave.Lib** (implements the Beckhoff IEC60870-5-104 substation). This library have to be included in the PLC project. All other libraries are included automatically.
- TcIEC870_5_104.Lib (implements the transfer protocol according to IEC60870-5-104);
- TcIEC870_5_101.Lib (implements the connection functions and common data types);
- TcSocketHelper.Lib (TCP/IP help funtions);
- TcpIp.Lib (TCP/IP base functions);
- TwinCAT TCP/IP Connection Server;

Installation on Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

The PLC libraries are copied into ..\TwinCAT\Plc\Lib folder. The TwinCAT TCP/IP Connection Server is entered in the TwinCAT Server list. The TCP/IP Connection Server is automatically started when TwinCAT is started and stopped when TwinCAT is stopped.

Installation on Windows CE

Product version for the runtime system under Windows CE is available as separate product. If you have CE version, please do the following steps:

- Install the product on your programming PC. The PLC libraries are copied into ..\TwinCAT\Plc\Lib folder.
- X86 CPU (CX1000, CX1020, PC)
  - The folder **...\TwinCAT\CE\TCPIP\Install** contains a Cabinet-File for the CE runtime system.

- Copy the file: **TcTCPIPSvrCe.I586.CAB** in a folder of the CE runtime system.

- ARM CPU (CX9000), TwinCAT PLC Library: IEC 60870-5-104 controlled station (slave) **v2.0.4** or higher:
  - The folder **...\TwinCAT\CE\TCPIP\Install** contains a Cabinet-File for the CE runtime system.
  - Copy the file: **TcTCPIPSvrCe.ARMV4I.CAB** in a folder of the CE runtime system.
- On CE system: Install (double click to Cabinet-File) CE components.
- Please suspend the CE device once after installation via "Start-> Suspend". The TwinCAT TCP/IP Connection Server starts with CE operating system.

This document is not full product manual. Please install the full version of the Beckhoff Information System.

You will find it

- on the Beckhoff Product DVDs
- on our Web-Server: http://www.beckhoff.com under download.

Examples

Example projects are in the Beckhoff Information System documentation of TwinCAT PLC library.

Link to 'high level' example overview page: IEC 60870-5-104 Controlled Station [▶ 479];

Link to 'low level' example overview page: IEC 60870-5-104 Transport Interface [▶ 450];

Further Documentation

- Documentation of TwinCAT PLC Library ('low level' interface): IEC 60870-5-104 Transport Interface [▶ 443];
- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
- Documentation: TwinCAT TCP/IP Connection Server;
- IEC 60870-5-1 Transmisstion frame formats;
- IEC 60870-5-2 Link transmission procedures;
- IEC 60870-5-3 General structure of application data;
- IEC 60870-5-4 Definition and coding of application information elements;
- IEC 60870-5-5 Basic application functions;
- IEC 60870-5-101 Companion Standard for basic telecontrol tasks;
- IEC 60870-5-101:1995/A1:2000 Companion Standard for basic telecontrol tasks. Amendment 1;
- IEC 60870-5-101:1995/A2:2001 Companion Standard for basic telecontrol tasks. Amendment 2;
- IEC 60870-5-104:2000 Network access for IEC 60870-5-101 using standard transport profiles;

12.1 Introduction (tutorial)

This introduction explains how to implement and configure an IEC60870-5-104 substation (slave) within the TwinCAT PLC.

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

12.1.1 Creating a PLC project and integrating PLC libraries

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

- Start TwinCAT PLC Control.

- Create a new PLC project via File -> New. Select PC or CX (x86 or ARM) as the target system.

- A new MAIN program block will now be created automatically. Select ST (Structured Text) as the language for the block. Confirm.

- From the menu select Window -> Library Management and then Insert -> Further Library...

- Select **TcIEC870_5_104Slave.Lib** from the list of TwinCAT libraries and confirm.

12.1.2 Defining and configuring an application object database of controlled station

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

Application objects = single points, double points, measured values, short floating point values, etc.

In this example the commands have been configured in a way that the process data are placed in the same memory area but on another byte/bit offset than the data of information in control direction. But you can also place the commands to the same byte/bit offset like information in control direction.

Example:

C_SC_NA_1 with IOA = 10 on the same byte/bit offset like M_SP_NA_1 with IOA = 100 (both byte ofset = 100 and bit offset = 0).
In this case a change of value via a command from the control station will cause a transfer of the M_SP_NA_1 with the object address 100 and transfer cause <11> (returned by remote command).

As an example we will configure the following application objects as part of the introductory project:

Requirements

Array element	ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
0	M_SP_NA_1	100	General interrogation group global	0	Memory	100	0	1 bit
1	M_SP_NA_1	101	General interrogation group global	0	Memory	100	1	1 Bit
2	M_SP_TB_1	102	General interrogation group global	0	Memory	100	2	1 Bit
3	M_DP_NA_1	200	General interrogation group global	0	Memory	200	0	2 Bits
4	M_DP_NA_1	201	General interrogation group global	0	Memory	200	2	2 Bits

Array element	ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
5	M_DP_TB_1	202	General interrogation group global	0	Memory	200	4	2 Bits
6	M_ST_NA_1	300	General interrogation group global	0	Memory	300	0	1 Byte
7	M_ST_NA_1	301	General interrogation group global	0	Memory	301	0	1 Byte
8	M_ST_TB_1	302	General interrogation group global	0	Memory	302	0	1 Byte
9	M_BO_NA_1	400	General interrogation group global	0	Memory	400	0	4 Byte
10	M_BO_NA_1	401	General interrogation group global	0	Memory	404	0	4 Byte
11	M_BO_TB_1	402	General interrogation group global	0	Memory	408	0	4 Byte
12	M_ME_NA_1	500	General interrogation group global	0	Memory	500	0	2 Byte
13	M_ME_NA_1	501	General interrogation group global	0	Memory	502	0	2 Byte
14	M_ME_TD_1	502	General interrogation group global	0	Memory	504	0	2 Byte
15	M_ME_NB_1	600	General interrogation group global	0	Memory	600	0	2 Byte
16	M_ME_NB_1	601	General interrogation group global	0	Memory	602	0	2 Byte
17	M_ME_TE_1	602	General interrogation group global	0	Memory	604	0	2 Byte

Array element	ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
18	M_ME_NC_1	700	General interrogation group global	0	Memory	700	0	4 Byte
19	M_ME_NC_1	701	General interrogation group global	0	Memory	704	0	4 Byte
20	M_ME_TF_1	702	General interrogation group global	0	Memory	708	0	4 Byte
21	M_IT_NA_1	800	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	800	0	4 Byte
22	M_IT_NA_1	801	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	804	0	4 Byte
23	M_IT_TB_1	802	General counter interrogation and mode A (local freeze with spontaneous transfer every 15s)	0	Memory	808	0	4 Byte
Commands								
24	C_SC_NA_1	10	-	0	Memory	2100	0	1 Bit
25	C_SC_NA_1	11	-	0	Memory	2100	1	1 Bit
26	C_SC_TA_1	12	-	0	Memory	2100	2	1 Bit
27	C_DC_NA_1	20	-	0	Memory	2200	0	2 Bit
28	C_DC_NA_1	21	-	0	Memory	2200	2	2 Bit

Array element	ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
29	C_DC_TA_1	22	-	0	Memory	2200	4	2 Bit
30	C_RC_NA_1	30	-	0	Memory	2300	0	1 Byte
31	C_RC_NA_1	31	-	0	Memory	2301	0	1 Byte
32	C_RC_TA_1	32	-	0	Memory	2302	0	1 Byte
33	C_BO_NA_1	40	-	0	Memory	2400	0	4 Byte
34	C_BO_NA_1	41	-	0	Memory	2404	0	4 Byte
35	C_BO_TA_1	42	-	0	Memory	2408	0	4 Byte
36	C_SE_NA_1	50	-	0	Memory	2500	0	2 Byte
37	C_SE_NA_1	51	-	0	Memory	2502	0	2 Byte
38	C_SE_TA_1	52	-	0	Memory	2504	0	2 Byte
39	C_SE_NB_1	60	-	0	Memory	2600	0	2 Byte
40	C_SE_NB_1	61	-	0	Memory	2602	0	2 Byte
41	C_SE_TB_1	62	-	0	Memory	2604	0	2 Byte
42	C_SE_NC_1	70	-	0	Memory	2700	0	4 Byte
43	C_SE_NC_1	71	-	0	Memory	2704	0	4 Byte
44	C_SE_TC_1	72	-	0	Memory	2708	0	4 Byte

Declaring a database variable

The application object database is an array variable of type ST_IEC870_5_101AODBEntry [▶ 313]. Each array element corresponds to an application object. The maximum number of application objects is freely selectable and is only limited by the available memory. During PLC programming you have to specify a constant maximum number. The maximum number of application objects cannot be changed at runtime.

In our example 50 application objects are declared. This number is sufficient for most applications. Please note that many application objects require adequate memory and runtime resources.

Define the following variable in MAIN:

```
PROGRAM MAIN
VAR
     AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
END_VAR
```

Configuring application objects

The object type (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 etc.), the object address and further object parameters are specified during configuration of the individual application objects.

The required application objects are configured during program runtime. Each application object (database array element) is configured by calling the F_iecInitAOEntry [▶ 280] function once. The array element to be configured is transferred to the function via VAR_IN_OUT. Configuration is usually carried out once during PLC program start-up via an Init routine. The F_iecInitAOEntry function expects the following function parameters (from left to right):

```
FUNCTION F_iecInitAOEntry : UDINT
VAR_INPUT
    eType           : E_IEC870_5_101TcTypeID;
    objAddr         : DWORD := 0;
    group           : DWORD := 0;
    multiplier      : BYTE := 0;
    ioMapType       : E_IEC870_5_101IOMappingType;
    byteOffs        : UDINT := 0;
    bitOffs         : UDINT := 0;
END_VAR
VAR_IN_OUT
    dbEntry         : ST_IEC870_5_101AODBEntry;
END_VAR
```

eType: application object type (ASDU identifier, e.g.: M_SP_NA_1 for single point or M_DP_NA_1 for double point). Please note that only the ASDU types listed in the compatibility list can be used. Invalid types are ignored.

objAddr : object address, e.g. 100. Each application object should be configured with a unique address.

group: group configuration parameters. The available group parameters are defined as constants and can be combined with an OR operator. E.g.: IEC870_GRP_INROGEN OR IEC870_GRP_PERCYC.

A description of all group configuration parameters [▶ 348] can be found here.

mutiplier: base time multiplier for cyclic/periodic data transfer (0=deactivated). The base time is configured via the system parameters. If the base time was set to T#10 s, and the multiplier to 2, for example, the periodic/cyclic data of the application object are sent every 20 seconds.

ioMapType: This parameter defines from or in which process data area of the TwinCAT PLC the IEC process data are to be mapped at runtime (inputs, outputs, memory, data).

byteOffs: process data area byte offset;

bitOffs: process data area bit offset;

dbEntry: application object to be configured (a database variable array element that is transferred to the function via VAR_IN_OUT).

In order to configure the application objects during program start-up, add the following PLC code in MAIN:

```
PROGRAM MAIN
VAR
    AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;

     init        : BOOL := TRUE;
    initError       : UDINT;
END_VAR
```

```
IF init THEN
    init := FALSE;

  (* Monitored Single Points *)
 initError := F_iecInitAOEntry( M_SP_NA_1, 100, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 100, 0, AODB
[0] );
    initError := F_iecInitAOEntry( M_SP_NA_1, 101, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 100, 1, A
ODB[1] );
    initError := F_iecInitAOEntry( M_SP_TB_1, 102, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 100, 2, A
ODB[2] );
  (* Double Points*)
 initError := F_iecInitAOEntry( M_DP_NA_1, 200, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 200, 0, AODB
[3] );
    initError := F_iecInitAOEntry( M_DP_NA_1, 201, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 200, 2, A
ODB[4] );
    initError := F_iecInitAOEntry( M_DP_TB_1, 202, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 200, 4, A
ODB[5] );
  (* Regulating step value *)
 initError := F_iecInitAOEntry( M_ST_NA_1, 300, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 300, 0, AODB
[6] );
```

```
    initError := F_iecInitAOEntry( M_ST_NA_1, 301, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 301, 0, A
ODB[7] );
    initError := F_iecInitAOEntry( M_ST_TB_1, 302, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 302, 0, A
ODB[8] );
  (* 32 bit string*)
 initError := F_iecInitAOEntry( M_BO_NA_1, 400, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 400, 0, AODB
[9 );
    initError := F_iecInitAOEntry( M_BO_NA_1, 401, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 404, 0, A
ODB[10] );
    initError := F_iecInitAOEntry( M_BO_TB_1, 402, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 408, 0, A
ODB[11] );
  (* Measured value, normalized value *)
 initError := F_iecInitAOEntry( M_ME_NA_1, 500, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 500, 0, AODB
[12] );
    initError := F_iecInitAOEntry( M_ME_NA_1, 501, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 502, 0, A
ODB[13] );
    initError := F_iecInitAOEntry( M_ME_TD_1, 502, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 504, 0, A
ODB[14] );
  (* Mesured value, scaled value *)
 initError := F_iecInitAOEntry( M_ME_NB_1, 600, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 600, 0, AODB
[15] );
    initError := F_iecInitAOEntry( M_ME_NB_1, 601, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 602, 0, A
ODB[16] );
    initError := F_iecInitAOEntry( M_ME_TE_1, 602, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 604, 0, A
ODB[17] );
  (* Measured value , short floating point value *)
 initError := F_iecInitAOEntry( M_ME_NC_1, 700, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 700, 0, AODB
[18] );
    initError := F_iecInitAOEntry( M_ME_NC_1, 701, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 704, 0, A
ODB[19] );
    initError := F_iecInitAOEntry( M_ME_TF_1, 702, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY, 708, 0, A
ODB[20] );
  (* Integrated totals *)
 initError := F_iecInitAOEntry( M_IT_NA_1, 800, IEC870_GRP_REQCOGEN OR IEC870_GRP_LOCFREEZE, 0, MAP_
AREA_MEMORY, 800, 0, AODB[21] );
    initError := F_iecInitAOEntry( M_IT_NA_1, 801, IEC870_GRP_REQCOGEN OR IEC870_GRP_LOCFREEZE, 0, M
AP_AREA_MEMORY, 804, 0, AODB[22] );
    initError := F_iecInitAOEntry( M_IT_TB_1, 802, IEC870_GRP_REQCOGEN OR IEC870_GRP_LOCFREEZE, 0, M
AP_AREA_MEMORY, 808, 0, AODB[23] );

  (* Single commands *)
 initError := F_iecInitAOEntry( C_SC_NA_1, 10, 0, 0, MAP_AREA_MEMORY, 2100, 0, AODB[24] );
    initError := F_iecInitAOEntry( C_SC_NA_1, 11, 0, 0, MAP_AREA_MEMORY, 2100, 1, AODB[25] );
    initError := F_iecInitAOEntry( C_SC_TA_1, 12, 0, 0, MAP_AREA_MEMORY, 2100, 2, AODB[26] );
  (* Double commands *)
 initError := F_iecInitAOEntry( C_DC_NA_1, 20, 0, 0, MAP_AREA_MEMORY, 2200, 0, AODB[27] );
    initError := F_iecInitAOEntry( C_DC_NA_1, 21, 0, 0, MAP_AREA_MEMORY, 2200, 2, AODB[28] );
    initError := F_iecInitAOEntry( C_DC_TA_1, 22, 0, 0, MAP_AREA_MEMORY, 2200, 4, AODB[29] );
  (* Regulating step commands *)
 initError := F_iecInitAOEntry( C_RC_NA_1, 30, 0, 0, MAP_AREA_MEMORY, 2300, 0, AODB[30] );
    initError := F_iecInitAOEntry( C_RC_NA_1, 31, 0, 0, MAP_AREA_MEMORY, 2301, 0, AODB[31] );
    initError := F_iecInitAOEntry( C_RC_TA_1, 32, 0, 0, MAP_AREA_MEMORY, 2302, 0, AODB[32] );
  (* 32 bit string commands *)
 initError := F_iecInitAOEntry( C_BO_NA_1, 40, 0, 0, MAP_AREA_MEMORY, 2400, 0, AODB[33] );
    initError := F_iecInitAOEntry( C_BO_NA_1, 41, 0, 0, MAP_AREA_MEMORY, 2404, 0, AODB[34] );
    initError := F_iecInitAOEntry( C_BO_TA_1, 42, 0, 0, MAP_AREA_MEMORY, 2408, 0, AODB[35] );
  (* Set point, normalized values*)
 initError := F_iecInitAOEntry( C_SE_NA_1, 50, 0, 0, MAP_AREA_MEMORY, 2500, 0, AODB[36] );
    initError := F_iecInitAOEntry( C_SE_NA_1, 51, 0, 0, MAP_AREA_MEMORY, 2502, 0, AODB[37] );
    initError := F_iecInitAOEntry( C_SE_TA_1, 52, 0, 0, MAP_AREA_MEMORY, 2504, 0, AODB[38] );
  (* Set point, scaled valuess *)
 initError := F_iecInitAOEntry( C_SE_NB_1, 60, 0, 0, MAP_AREA_MEMORY, 2600, 0, AODB[39] );
    initError := F_iecInitAOEntry( C_SE_NB_1, 61, 0, 0, MAP_AREA_MEMORY, 2602, 0, AODB[40] );
    initError := F_iecInitAOEntry( C_SE_TB_1, 62, 0, 0, MAP_AREA_MEMORY, 2604, 0, AODB[41] );
  (* Set point, short floating point values *)
 initError := F_iecInitAOEntry( C_SE_NC_1, 70, 0, 0, MAP_AREA_MEMORY, 2700, 0, AODB[42] );
    initError := F_iecInitAOEntry( C_SE_NC_1, 71, 0, 0, MAP_AREA_MEMORY, 2704, 0, AODB[43] );
    initError := F_iecInitAOEntry( C_SE_TC_1, 72, 0, 0, MAP_AREA_MEMORY, 2708, 0, AODB[44] );

END_IF
```

Also see about this

- E_IEC870_5_101TcTypeID [▶ 327]
- E_IEC870_5_101IOMappingType [▶ 331]

12.1.3 Mapping of PLC and IEC process data

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The TwinCAT PLC process data are cyclically mapped (copied) into the IEC process data (application objects) and vice versa at program runtime. For mapping of the IEC<->PLC process data up to 4 process data areas (IO inputs, IO outputs, memory area, data area) can be declared as buffer variables in the PLC program. The byte size of the buffers is freely selectable and may be different for each area. Its not necessary to declare unused buffer areas.

In our introductory example we declare 4 PLC process data areas with 3000 bytes each:

```
PROGRAM MAIN
VAR
    AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;

    init        : BOOL := TRUE;
    initError       : UDINT;

     inputs AT%IB0   : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0   : ARRAY[0..2999] OF BYTE;
    memory AT%MB0    : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;

END_VAR
```

How the process data are to be mapped at runtime is specified during configuration of the application objects via the F_iecInitAOEntry [▶ 280] function.
See also in: Declaration and configuration of application objects [▶ 454].

Now the buffer variables are declared as byte arrays. To have better access to the desired data the variables are declared a second time and assigned to the corresponding byte/bit offset addresses.
Now a change within the byte array causes a change of the corresponding variable and vice versa. This is not imperative necessary. You can also have access directly to the bits/bytes of the byte array buffer variables.

```
VAR_GLOBAL
(* Memory offset 0..99 unused *)
  (* Single points *)
 msgSingle_0     AT%MX100.0 : BOOL;
    msgSingle_1     AT%MX100.1 : BOOL;
    msgSingle_2     AT%MX100.2 : BOOL;

  (* Double points *)
    (*      Bit 0..1 = first double point,
        Bit 2..3 = second double point,
        Bit 4..5 = third double point,
        Bit 6..7 = fourth double point *)
    msgDouble_0     AT%MB200    : BYTE;

  (* Regulating step values *)
    msgStep_0           AT%MB300    : BYTE;
    msgStep_1           AT%MB301    : BYTE;
    msgStep_2           AT%MB302    : BYTE;

  (* 32 bit strings *)
 msgBitStr_0     AT%MD400    : DWORD := 2#10001000_10001000_10001000_10001000;
    msgBitStr_1     AT%MD404    : DWORD := 2#10001000_10001000_10001000_10001000;
    msgBitStr_2     AT%MD408    : DWORD := 2#10001000_10001000_10001000_10001000;

  (* Measured values, normalized values *)
 msgNormalized_0     AT%MW500    : WORD;
    msgNormalized_1     AT%MW502    : WORD;
    msgNormalized_2     AT%MW504    : WORD;

  (* Mesured values, scaled values *)
 msgScaled_0     AT%MW600    : INT;
    msgScaled_1     AT%MW602    : INT;
    msgScaled_2     AT%MW604    : INT;

  (* Measured values, short floating point values *)
 msgFloating_0   AT%MD700    : REAL;
    msgFloating_1   AT%MD704    : REAL;
    msgFloating_2   AT%MD708    : REAL;
```

```
  (* Integrated totals *)
 msgTotal_0      AT%MD800    : UDINT;
    msgTotal_1      AT%MD804    : UDINT;
    msgTotal_2      AT%MD808    : UDINT;

  (*#################################################################*)
    (* Single commands *)
 cmdSingle_0     AT%MX2100.0 : BOOL;
    cmdSingle_1     AT%MX2100.1 : BOOL;
    cmdSingle_2     AT%MX2100.2 : BOOL;

  (* Double commands *)
    (*      Bit 0..1 = first double command,
        Bit 2..3 = second double command,
        Bit 4..5 = third double command,
        Bit 6..7 = fourth double command *)
 cmdDouble_0     AT%MB2200       : BYTE;

  (* Regulating step commands *)
 cmdStep_0           AT%MB2300 : BYTE;
    cmdStep_1           AT%MB2301 : BYTE;
    cmdStep_2           AT%MB2302 : BYTE;

  (* 32 bit string commands *)
 cmdBitStr_0     AT%MD2400       : DWORD;
    cmdBitStr_1     AT%MD2404       : DWORD;
    cmdBitStr_2     AT%MD2408       : DWORD;

  (* Set point, normalized values *)
 cmdNormalized_0     AT%MW2500       : WORD;
    cmdNormalized_1     AT%MW2502       : WORD;
    cmdNormalized_2     AT%MW2504       : WORD;

  (* Set point, scaled values *)
 cmdScaled_0     AT%MW2600       : INT;
    cmdScaled_1     AT%MW2602       : INT;
    cmdScaled_2     AT%MW2604       : INT;

  (* Set point, short floating point values *)
 cmdFloating_0   AT%MD2700       : REAL;
    cmdFloating_1   AT%MD2704       : REAL;
    cmdFloating_2   AT%MD2708       : REAL;
END_VAR
```

Mapping of the IEC<->PLC process data in controlled station

Process data in monitoring direction (slave->master information)

Example 1

Single point information (M_SP_NA_1) with the IOA = 100, PLC memory area, byte offset = 100, bit offset = 0.

msgSingle_0 == memory[100].0 -> Controlled station FB -> ... -> Controlling station

Example 2

Measured value, short floating point value (M_ME_NC_1) with the IOA = 700, PLC memory area, byte offset = 700, bit offset = 0 (irrelevant).

msgFloating_0 == memory[700..703] -> Controlled station FB -> ... -> Controlling station

Process data in control direction (master->slave commands)

Example 1

Single command state (C_SC_NA_1) with the IOA = 10, PLC memory area, byte offset = 2100, bit offset = 0.

Controlling station -> ... -> Controlled station FB -> memory[2100].0 == cmdSingle_0

Example 2

Set point, short floating point value (C_SE_NC_1) with the IOA = 70, PLC memory area, byte offset = 2700, bit offset = 0 (irrelevant).

Controlling station -> ... -> Controlled station FB -> memory[2700..2703] == cmdFloating_0

12.1.4 Declaring and calling an instance of the IEC60870-5-104 substation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The complete functionality of a substation is encapsulated in the FB_IEC870_5_104Slave function block. An instance can be used to establish a connection to the master. For establishing a further connection a further instance can be declared, and the same server handle (*hServer* variable) can be transferred to this second instance or use the FB_IEC870_5-104SlaveGrp [▶ 470] function block (preferred!). You have change the host address (IP address) to the address of your target system.

Add the following PLC code to the declaration part of MAIN:

```
PROGRAM MAIN
VAR
    AODB        : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;

    init        : BOOL := TRUE;
    initError           : UDINT;

    inputs AT%IB0   : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..2999] OF BYTE;
    memory AT%MB0   : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;


     bEnable         : BOOL := TRUE;
    hServer         : T_HSERVERremoved link: <a href="mk:@MSITStore:TcPlcLibSocketHelper.chm::/HTML/
TcPlcLibSocketHelper_T_HSERVER.htm" lnkid="13_2">T_HSERVER</a>;
    server      : FB_IEC870_5_104Slaveremoved link: <a href="mk:@MSITStore:TcPlcLibIEC870_5_104Slave
.chm::/HTML/TcPlcLibIEC870_5_104Slave_FB_IEC870_5_104Slave.htm" lnkid="13_3">FB_IEC870_5_104Slave</
a>;
END_VAR
```

and the instance is called in the program part:

```
IF init THEN
    init := FALSE;
...
F_CreateServerHndremoved link: <a href="mk:@MSITStore:TcPlcLibSocketHelper.chm::/HTML/
TcPlcLibSocketHelper_F_CreateServerHnd.htm" lnkid="13_4">F_CreateServerHnd</
a>( '', '127.0.0.1'(* change this! *), 2404, nMode := LISTEN_MODE_CLOSEALL OR CONNECT_MODE_ENABLEDBG
, bEnable, hServer );
ELSE

    server( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
        cbOutputs := SIZEOF( outputs ),
        pMemory := ADR( memory ),
        cbMemory := SIZEOF( memory ),
        pData := ADR( data ),
        cbData := SIZEOF( data ),
        pAOEntries := ADR( AODB ),
        cbAOEntries := SIZEOF( AODB ),
        hServer := hServer,
        bEnable := bEnable );
END_IF
```

12.1.5 IEC60870-5-104 protocol parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The behaviour of the substation can be adapted to the requirements of the master via the IEC60870-5-104 protocol parameters. Most parameters have preallocated default values that do not have to be changed.

In our example we change the values of the iK and iW parameters:

```
IF init THEN
    init := FALSE;
...
    F_CreateServerHnd( '', '127.0.0.1'(* change this! *), 2404, nMode := LISTEN_MODE_CLOSEALL OR CON
NECT_MODE_ENABLEDBG, bEnable, hServer );

    server.protPara.iK := 12;
    server.protPara.iW := 8;

ELSE
    server( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all transfer protocol parameters can be found here: ST_IEC870_5_104PotocolParams [▶ 446].

12.1.6 System parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The common ASDU address and the user functions of the substation are configured via the system parameters.

In our introduction we configure the following system parameters:

- The common ASDU address is set to 7. (*asduAddr*)
- During initialisation the system time of the substation is synchronised with the system time of the local TwinCAT PC (*bUsePCTime*).
- Synchronisation of the substation system time via the time synchronisation command is activated (*bSyncTime*).
- The system time of the TwinCAT PC (*bSyncPCTime*) should not be synchronised during synchronisation of the substation system time.
- Sending of M_EI_NA_1 (End of init) to the central station is activated (*bEndOfInit*).
- Sending of the periodic/cyclic data is deactivated (*bPerCyclic*). The base time for sending of these data is set to 5 s.
- Background scan is deactivated (*bBackScan*). The background scan cycle time is set to 30 s.
- Local freeze and resetting of the counter readings is activated (*bPerFRZ*), and the cycle time for freeze and resetting is set to 15 s.
- Logging of debugging messages in the application log is activated (*dbgMode*). Logging of device state changes is enabled.

Add the following PLC code to your PLC project:

```
IF init THEN
    init := FALSE;
...


    server.sysPara.asduAddr := 7;
    server.sysPara.bUsePCTime := TRUE;
    server.sysPara.bSyncTime := TRUE;
    server.sysPara.bSyncPCTime := FALSE;
    server.sysPara.bEndOfInit := TRUE;
    server.sysPara.bPerCyclic := FALSE;
    server.sysPara.tPerCyclicBase := T#5s;
    server.sysPara.bBackScan := FALSE;
    server.sysPara.tBackScanCycle := T#30s;
    server.sysPara.bPerFRZ := TRUE;
    server.sysPara.tPerFRZCycle := T#15s;
    server.sysPara.dbgMode := (*IEC870_DEBUGMODE_ASDU OR*) IEC870_DEBUGMODE_DEVSTATE;

...
```

```
ELSE
    server( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all system parameters can be found here: ST_IEC870_5_101SystemParams [▶ 317].

12.1.7 Station interrogation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The station interrogation command is initiated by the central station. The ID field of the command also contains the group (1 to 16 or general). The substation transfers the application objects associated with this group to the central station with cause of transmission <*20*> to <*36*>. Application objects with time tags are transferred without time tags.

Configuration of the system parameters:

- It's not necessary to set additional system parameters;

Configuration of the application objects:

- The application has to assign the data point to one or more than one interrogation group. Here you can find all available group parameters: Group configuration flags [▶ 348].

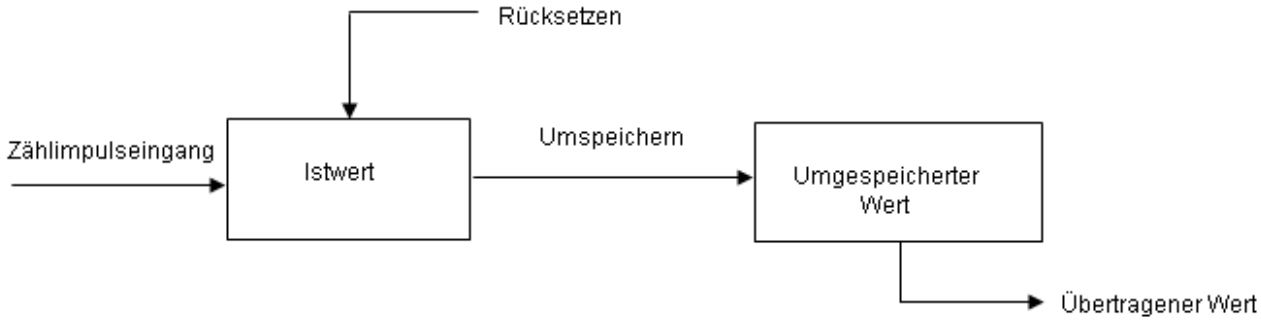
Example of data point configuration. The data point belongs to group: 1 and group: general.

```
F_iecInitAOEntry( M_SP_NA_1, 100, IEC870_GRP_INROGEN OR IEC870_GRP_INRO1, 0, MAP_AREA_MEMORY, 100, 0
, AODB[0] );
```

12.1.8 Transfer of integrated totals (counter interrogation)

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

General count transfer model:



The actual values are added by counters. Via a re-store command that is either received by the central station or generated locally (in the substation), the actual values can be re-stored (copied) periodically into re-stored values. After freezing, the recorded value is either reset to zero (logging of incremental values), or the counter continues adding up (logging of counter readings).

Application objects with counts are assigned to groups. The groups are frozen individually, reset, or transferred. The central station sends count query commands to the substation. The task to be carried out (FRZ) and the group (RQT) are specified in an ID field of the command (QCC).

The allocation of the application objects to the individual groups (1 to 4 or general) is specified via the group flag parameter during configuration. There are four operating modes for recording counter readings and incremental values. Each mode includes notes about the configuration of the system parameters or the application objects.

Mode A: local freeze with spontaneous transfer

The substation internally initiates freeze or freeze with reset. The frozen counts are transferred spontaneously, once the freeze or freeze with reset function was executed. In this mode the central station does not issue any count query commands.

Configuration of the system parameters:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

The first parameter activates local freeze or freeze with reset. The second parameter specifies the cycle time with which freeze or freeze with reset is carried out (e.g. every 60 seconds).

Configuration of the application objects:

- The IEC870_GRP_SPONTOFF group parameter must not be set, since it would prevent spontaneous data transfer of the counts.
- The count is frozen if the IEC870_GRP_LOCFREEZE group parameter was set.
- The count is reset if the IEC870_GRP_LOCRESET group parameter was set.
- Local freeze or freeze with reset is carried out simultaneously for all groups (1 to 4 or general).

Operating mode B: Local freeze with counter interrogation

The substation internally initiates freeze or freeze with reset. The central station queries the frozen counts via count query commands. In this case the central station must not use freeze or freeze with reset in the command ID field (FRZ=0). The counts are queried generally or in groups 1 to 4.

Configuration of the system parameters:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

The first parameter activates local freeze or (and) reset. The second parameter specifies the cycle time with which freeze or freeze with reset is carried out (e.g. every 60 seconds).

Configuration of the application objects:

- The IEC870_GRP_SPONTOFF group parameter must be set. The counts are not to be transferred spontaneously to the central station.
- The count is frozen if the IEC870_GRP_LOCFREEZE group parameter was set.
- The count is reset if the IEC870_GRP_LOCRESET group parameter was set.
- Local freeze or freeze with reset is carried out simultaneously for all groups (1 to 4 or general).

Operating mode C: The central station initiates freeze, freeze with reset, or reset

The central station periodically issues a count query command to the substation for controlling the freeze or (and) reset process. This command does not result in a count transfer. The central station sends a subsequent count query command for collecting the frozen counts. This is similar to operating mode B.

Configuration of the system parameters:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Local freeze or (and) reset must be deactivated. The second parameter is ignored.

Configuration of the application objects:

- IEC870_GRP_SPONTOFF must be set. The counts are not to be transferred spontaneously to the central station.
- The IEC870_GRP_LOCFREEZE and IEC870_GRP_LOCRESET group parameters must not be set. The central station initiates freeze or (and) reset.
- The counts can be assigned to individual groups (1 to 4 or general) and queried (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1, IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

Operating mode D: The central station initiates freeze and (or) reset, and the frozen values are transferred spontaneously

This mode is a combination of the count command from the central station as in mode C and spontaneous transfer of the counts as in mode A.

Configuration of the system parameters:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Local freeze or (and) reset must be deactivated. The second parameter is ignored.

Configuration of the application objects:

- The IEC870_GRP_SPONTOFF group parameter must not be set, since it would prevent spontaneous data transfer of the counts.
- The IEC870_GRP_LOCFREEZE and IEC870_GRP_LOCRESET group parameters must not be set. The central station initiates freeze or (and) reset.
- The counts can be assigned to individual groups (1 to 4 or general) and queried (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1, IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

12.1.9 Clock (time) synchronisation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The way in which the system time of the substation is to be synchronised can be configured via the system parameters.

- During initialisation the system time of the substation can be synchronised with the system time of the local TwinCAT PC;
- When a time synchronisation command is received from the central station, the system time of the substation can also be synchronised;
- The system time of the local TwinCAT PC can also be synchronised when a time synchronisation command is received.

12.1.10 Background scan

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The background scan is used for refreshing the process information sent from the substation to the central station as an additional safety contribution to the station interrogation and for spontaneous transfers. Application objects with the same type IDs as for the station interrogation may be transferred continuously with low priority, and with <2> background scan as the cause of transmission. The valid ASDU type IDs are listed in the compatibility list for the station (table type ID <-> cause of transmission). The background scan is initiated by the substation and is independent of the station interrogation commands.

Configuration of the system parameters:

The transfer cycle is specified via system parameters [▶ 317] in the substation.

```
bBackScan := TRUE;
tBackScanCycle := T#30s;
```

Configuration of the application objects:

Application objects whose process data are to be transferred as a background scan have to be configured via the IEC870_GRP_BACKGROUND group flag.

Example:

```
F_iecInitAOEntry( M_SP_NA_1, 100, IEC870_GRP_INROGEN OR IEC870_GRP_BACKGROUND, 0, MAP_AREA_MEMORY, 1
00, 0, AODB[0] );
```

12.1.11 Cyclic data transmission

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

Cyclic data transfer is initiated in a similar way as the background scan from the substation. It is independent of other commands from the central station. Cyclic data transfer continuously refreshes the process data of the central station. The process data are usually measured values that are recorded at regular intervals. Cyclic data transfer is often used for monitoring non-time-critical or relatively slowly changing process data (e.g. temperature sensor data). Cyclic/periodic data are transferred to the central station with cause of transmission *<1> periodic/cyclic*. The valid ASDU type IDs are listed in the compatibility list for the station (table type ID <-> cause of transmission). Cyclic data transfer can be configured via the system parameters and the configuration parameters of the application objects.

Configuration of the system parameters:

```
bPerCyclic       : BOOL := TRUE;
tPerCyclicBase   : TIME := T#60s;
```

The first parameter activates cyclic transfer. The second parameter is the base time for the cyclic/periodic data transfer (in this case 60 seconds).

Configuration of the application objects:

- IEC870_GRP_PERCYC group parameter has to be set;
- The multiplier parameter *(multiplier)* of the F_iecInitAOEntry function has to be set to a zero <> value. Example: With a multiplier = 2 and a base time of 60 seconds the process data of the application object are sent to the central station every 120 seconds;

Example of measured value configuration. The value has to be transferred every 120 seconds to the central station (measured value, normalized value without time tag, M_ME_NA_1).

```
F_iecInitAOEntry( M_ME_NA_1, 222, IEC870_GRP_INROGEN OR IEC870_GRP_PERCYC, 2, MAP_AREA_MEMORY, 6, 0,
 AODB[2] );
```

12.1.12 Command transmission

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

Commands can be sent from the central station in control direction (to the substation). A single command with type ID 45 (C_SC_NA_1) is used for controlling an application object that is transferred in monitoring direction as a single-point information (M_SP_NA_1, M_SP_TA_1 or M_SP_TB_1). A dual command (C_DC_NA_1) is used for controlling an application object that is transferred in monitoring direction as a double-point information (M_DP_NA1, M_DP_TA_1 or M_DP_TB_1), etc.

Configuration of system parameters:

- It's not necessery to set additional system parameters;

Configuration of application objects:

- Configure the application object type as process data in control direction;
- The information object addresses (IOA's) should be equal to the addresses configured in control station (master);

Examples:

Single command with the IOA = 10. The command value is mapped to memory area buffer, byte offset = 100, bit offset = 0.

```
F_iecInitAOEntry( C_SC_NA_1, 10, 0, 0, MAP_AREA_MEMORY, 100, 0, AODB[24] );
```

Double command with the IOA = 20. The command value is mapped to memory area buffer, byte offset = 200, bit offset = 0..1.

```
F_iecInitAOEntry( C_DC_NA_1, 20, 0, 0, MAP_AREA_MEMORY, 200, 0, AODB[27] );
```

Set point, scaled value with the IOA = 60. The command value is mapped to memory area buffer, byte offset = 600..601, bit offset = 0.

```
F_iecInitAOEntry( C_SE_NB_1, 60, 0, 0, MAP_AREA_MEMORY, 600, 0, AODB[39] )
```

12.1.13 Interrogation / Read command

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The central station sends interrogation commands to the substation. The interrogation command contains the address of the application object to be interrogated. The data of this application object are to be sent to the central station. The substation sends the data with cause of transmission *<5> interrogation or interrogated*. The valid ASDU type IDs are listed in the compatibility list for the station (table type ID <-> cause of transmission).

Configuration of system parameters:

- It's not necessery to set additional system parameters;

Configuration of application objects:

- It's not necessery to set additional system parameters;

12.1.14 Double transmission

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

All application objects (information objects) that are transferred with cause of transmission *<3> spontaneous* may be transferred twice, with or without time tag. This mode is referred to as "double transmission". Double transmission is currently not supported by the substation.

12.1.15 Quality Flags

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The quality flags (quality descriptor) provide additional information for the central station on the quality of an application object. The quality flags can be set/reset independent of each other from the PLC application via the F_iecSetAOQuality [▶ 282] function. The F_iecGetAOQuality [▶ 283] function can be used to interrogate the state of the quality flags. Any change in the quality flags leads to a spontaneous transfer of the data to the central station.

The following quality flags are internally analyzed by the substation at runtime:

- IECQ_BL_ON (blocked). If the process data of the application object were blocked for the transfer, mapping of the PLC and IEC process data is not executed for this application;

The following quality flags are internally set/reset by the substation at runtime:

- IECQ_IV_ON (Invalid). The substation sets the invalid flag if mapping of the PLC and IEC process data could not be carried out (e.g. due to faulty configuration of the application object). This behavior can be deactivated by setting group parameter IEC870_GRP_IV_OFF.

All other quality flags are sent to the central station without change.

12.1.16 Testing the communication

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

You can simulate the data point value changes. Setting the *bChangeIO* variable to TRUE enables the simulation and to FALSE disables. The new values are send cyclicaly every 3 seconds to the control station.

```
PROGRAM MAIN
VAR
    ...

    bChangeIO : BOOL; (* TRUE => simulate/modify plc process data *)
 timer : TON;
    i : INT;

    ...
END_VAR
```

```
...


(*modify plc process data *)
timer( IN := bChangeIO, PT := T#3s );
IF timer.Q THEN
    timer( IN := FALSE );

    msgSingle_0 := NOT msgSingle_0;
    msgSingle_1 := NOT msgSingle_1;
    msgSingle_2 := NOT msgSingle_2;

    FOR i:= 0 TO 3 DO
        IF F_iecGetDPI(msgDouble_0, i) = eIEC870_DPI_ON THEN (* the value of double point allready O
N? *)
            msgDouble_0 := F_iecSetDPI(     msgDouble_0, i, eIEC870_DPI_OFF ); (* change ON => OFF *
)
        ELSE
            msgDouble_0 := F_iecSetDPI(     msgDouble_0, i, eIEC870_DPI_ON );(* change OFF => ON *)
        END_IF
    END_FOR

    F_iecIncVTI( msgStep_0 );
    F_iecDecVTI( msgStep_1 );

    msgBitStr_0 := ROL( msgBitStr_0, 1 );
    msgBitStr_1 := ROR( msgBitStr_1, 1 );

    msgNormalized_0 := msgNormalized_0 + 1;
    msgNormalized_1 := msgNormalized_1 + 2;

    msgScaled_0 := msgScaled_0 + 3;
    msgScaled_1 := msgScaled_1 - 3;

    msgFloating_0 := msgFloating_0 + 0.1;
    msgFloating_1 := msgFloating_1 + 1.5;

    msgTotal_0 := msgTotal_0 + 1;
    msgTotal_1 := msgTotal_1 + 2;
END_IF


...
```

12.1.17 Protocol and data transmission errors

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip

The station error messages are placed into a FIFO. Up to 10 error messages can be stored. At fatal communication errors (e.g. error of the linking layer, the check sum of the frame doesn't fit) the connection is cut and has to be build on new. Errors within the application layer (e.g. the ADSU send buffer is overflowed by to many frames) are only logged and doesn't cut the connection.

For this errors it is also possible to cut the connection out of the application. In addition to the error code also the error source is notified in the error message. This simplifies the localization of an error.

Example

The occurring error messages of an IEC 60870-5-104 sub station can be read out via the request:

```
PROGRAM MAIN
VAR
...
```

```
    server : FB_IEC870_5_104Slave;
...

END_VAR
```

```
...

REPEAT
    server.system.device.errors.RemoveError();
    IF server.system.device.errors.bOk  THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'IEC60870-5-104 slave error: 0x%s',
            DWORD_TO_HEXSTR( server.system.device.errors.getError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT server.system.device.errors.bOk
END_REPEAT

...
```

12.2 Quick Start

Simple projects including complete sources is to be found here: IEC60870-5-104 substation [▶ 479].

Interoperability list is to be found here: Interoperability check list

Overview of error codes is to be found here: Error Error codes

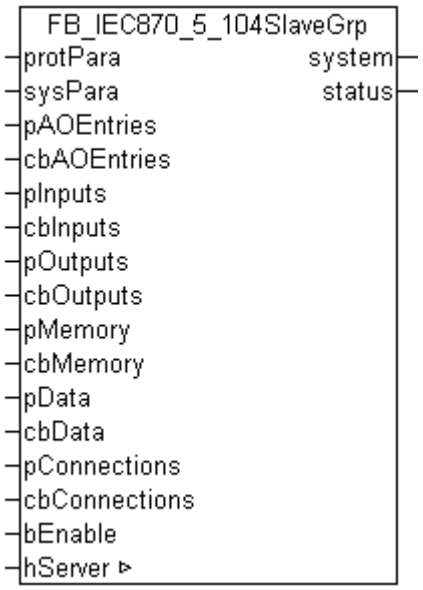
A detailed instruction about implementing the substation within the PLC is to be found here: TUTORIAL [▶ 453]

Brief instruction

1. Create a new PLC project and link the PLC library **TcIEC870_5_104Slave.Lib**.
2. Apply an instance of the T_HSERVER variable (connection handle) at the main program and initialize it once at the program part by calling of the F_CreateServerHnd function with the fitting parameters.
3. Configuration of data points: Apply an array variable of ST_IEC870_5_101AODBEntry [▶ 313]. Every array element corresponds to a data point. Configure the data points via the function F_iecInitAOEntry [▶ 280] at runtime (e.g. at an init step).
4. Apply an instance of the des protocol block FB_IEC870_5_104Slave [▶ 474] at the main program, configure it and call it.
   The system and protocol parameter are to be configured according to the parameter of the control station.

12.2.1 FB_IEC870_5_104SlaveGrp

Product Version: TwinCAT PLC Library: IEC60870-5-104 controlled station (slave) v2.0.2 and higher.



An instance of the FB_IEC870_5_104SlaveGrp function block can be used to implement an IEC60870-5-104 substation (slave) in the TwinCAT PLC. An instance of the function block can establish one connection with a master or more connections with a master (redundant systems)

The maximum number of connections can be defined by the number of ST_IEC870_5_104ServerConnection [▶ 477] array elements. The address and byte size of the array variables must be passed to the instance of FB_IEC870_5_104SlaveGrp function block.

VAR_IN_OUT

```
VAR_IN_OUT
    hServer : T_HSERVER;
ND_VAR
```

hServer : TCP/IP Server-Handle. The internal parameter of the Server-Handle variable has to be initialized with the function F_CreateServerHnd.

VAR_INPUT

```
VAR_INPUT
    protPara        : ST_IEC870_5_104ProtocolParams;                          (* IEC60870-5-104 pro
tocol communication params *)
    sysPara         : ST_IEC870_5_101SystemParams;                   (* IEC60870-5-104 slave syste
m params *)
    pAOEntries      : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF ST_IEC870_5_101AODBEntry := 0;
    (* Pointer to the first element of application database object array *)
    cbAOEntries     : UDINT := 0;                                  (* Byte size (length) of applicati
on database object array *)
    pInputs         : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbInputs        : UDINT := 0;
    pOutputs        : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbOutputs       : UDINT := 0;
    pMemory         : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbMemory        : UDINT := 0;
    pData           : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbData          : UDINT := 0;
    pConnections    : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF ST_IEC870_5_104ServerConnection :
= 0;
    cbConnections   : UDINT := 0;
    bEnable         : BOOL := TRUE;
END_VAR
```

protPara: IEC60870-5-104 protocol parameter [▶ 446].

sysPara: system parameter [▶ 317].

pAOEntries: address of the application object [▶ 313] database variable.

cbAOEntries: byte size of the application object database variable.

pInputs: address of the PLC process data area for the inputs.

cbInputs: byte size of the PLC process data area for the inputs.

pOutputs: address of the PLC process data area for the outputs.

cbOutputs: byte size of the PLC process data area for the outputs.

pMemory: address of the PLC process data area for the flags.

cbMemory: byte size of the PLC process data area for the flags.

pData: address of the PLC data area.

cbData: byte size of the PLC data area.

pConnections: address of the ST_IEC870_5_104ServerConnection [▶ 477] array variables.

cbConnections: byte size of the ST_IEC870_5_104ServerConnection array variables.

bEnable :activates/deactivates the function block (communication and connection).

The addresses can found out with the ADR operator, resp. the byte sizes can be found with the SIZEOF operator.

VAR_OUTPUT

```
VAR_OUTPUT
    system      : ST_IEC870_5_104SystemInterface;
    status      : ST_IEC870_5_104GrpStatus;
END_VAR
```

system: system interface [▶ 478]. This variable is used by other IEC application functions as a communication interface for the IEC device (here: substation).

- Member variable *system.device* is expected by the F_iecSetAOQuality [▶ 282] function as VAR_IN_OUT parameter, for example.
- Member variable *system.device.errors* is a device error FIFO. The PLC application can read and analyse registered errors.

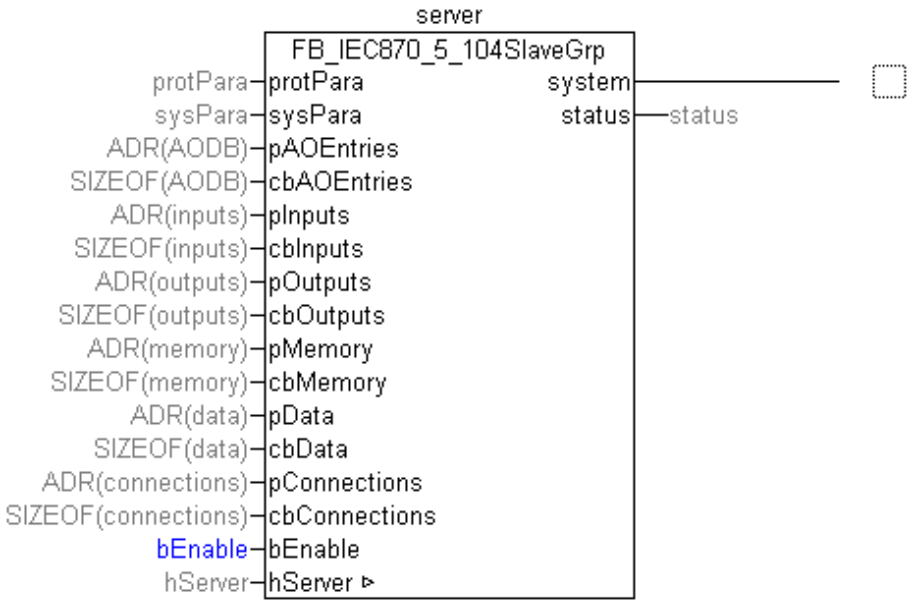
status: Connection and data transfer status [▶ 477] information.

Example:

Example projects: IEC60870-5-104 controlled station [▶ 479]

Call in FBD with maximum of 2 master connections:

```
PROGRAM test
VAR
    AODB        : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    inputs AT%IB0   : ARRAY[0..999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..999] OF BYTE;
    memory AT%MB0   : ARRAY[0..999] OF BYTE;
    data        : ARRAY[0..999] OF BYTE;

    hServer         : T_HSERVER;
    server      : FB_IEC870_5_104SlaveGrp;
    connections     : ARRAY[0..1] OF ST_IEC870_5_104ServerConnection; (* Two master connections *)


    bEnable         : BOOL := TRUE;
    protPara    : ST_IEC870_5_104ProtocolParams;
    sysPara         : ST_IEC870_5_101SystemParams := ( asduAddr := 7 );
    status      : ST_IEC870_5_104GrpStatus;
    bError      : BOOL;
    iecError    : ST_IEC870_5_101ErrorFifoEntry;
END_VAR
```



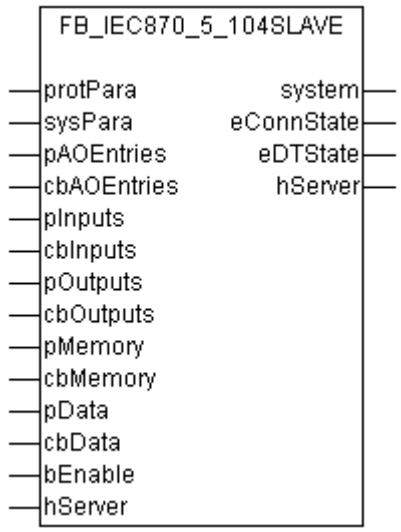
In the following structured text example the device error FIFO is read cyclically, and the registered errors are written into the Windows Application Log.

```
REPEAT
    server.system.device.errors.RemoveError( getError=>iecError, bOk=>bError );
    IF bError THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG, 'IEC60870-5-104 slave error: 0x%s', D
WORD_TO_HEXSTR( iecError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT bError
END_REPEAT
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104Slave.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_104.Lib, TcIEC870_5_101.Lib are included automatically)

12.2.2 FB_IEC870_5_104Slave



An instance of the FB_IEC870_5_104Slave function block can be used to implement an IEC60870-5-104 substation (slave) in the TwinCAT PLC. An instance of the function block can only establish a connection with a master.

Creating redundant systems

If you want to realize redundant systems with two or more connections use the FB_IEC870_5_104SlaveGrp [▶ 470] function block.

VAR_IN_OUT

```
VAR_IN_OUT
    hServer: T_HSERVER;
END_VAR
```

hServer : TCP/IP server handle. The internal parameters of the server handle variable first have to be initialized via the F_CreateServerHnd function.

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_104ProtocolParams;                          (* IEC60870-5-104 protoco
l communication params *)
    sysPara     : ST_IEC870_5_101SystemParams;                    (* IEC60870-5-104 slave system pa
rams *)
    pAOEntries  : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF ST_IEC870_5_101AODBEntry;    (* Point
er to the first element of application database object array *)
    cbAOEntries : UDINT;  (* Byte size (length) of application database object array *)
    pInputs     : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbInputs    : UDINT;
    pOutputs    : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbOutputs   : UDINT;
    pMemory     : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbMemory    : UDINT;
    pData       : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE;
    cbData      : UDINT;
    bEnable     : BOOL := TRUE;
END_VAR
```

protPara: IEC60870-5-104 protocol parameter [▶ 446].

sysPara: system parameter [▶ 317].

pAOEntries: address of the application object [▶ 313] database variable.

cbAOEntries: byte size of the application object database variable.

pInputs: address of the PLC process data area for the inputs.

cbInputs: byte size of the PLC process data area for the inputs.

pOutputs: address of the PLC process data area for the outputs.

cbOutputs: byte size of the PLC process data area for the outputs.

pMamory: address of the PLC process data area for the flags.

cbMamory: byte size of the PLC process data area for the flags.

pData: address of the PLC data area.

cbData: byte size of the PLC data area.

bEnable :activates/deactivates the function block (communication and connection).

VAR_OUTPUT

```
VAR_OUTPUT
    system      : ST_IEC870_5_104SystemInterface;
    eConnState  : E_SocketConnectionState      := eSOCKET_DISCONNECTED;  (* TCP/
IP connection state *)
    eDTState    : E_IEC870_5_104DataTransferState  := eIEC870_STOPDT;    (* IEC60870-5-104 data tran
sfer state *)
END_VAR
```

system: system interface [▶ 478]. This variable is used by other IEC application functions as a communication interface for the IEC device (here: substation).

- Member variable *system.device* is expected by the F_iecSetAOQuality [▶ 282] function as VAR_IN_OUT parameter, for example.
- Member variable *system.device.errors* is a device error FIFO. The PLC application can read and analyze registered errors.

eConnState: status of the TCP/IP connection with the master.

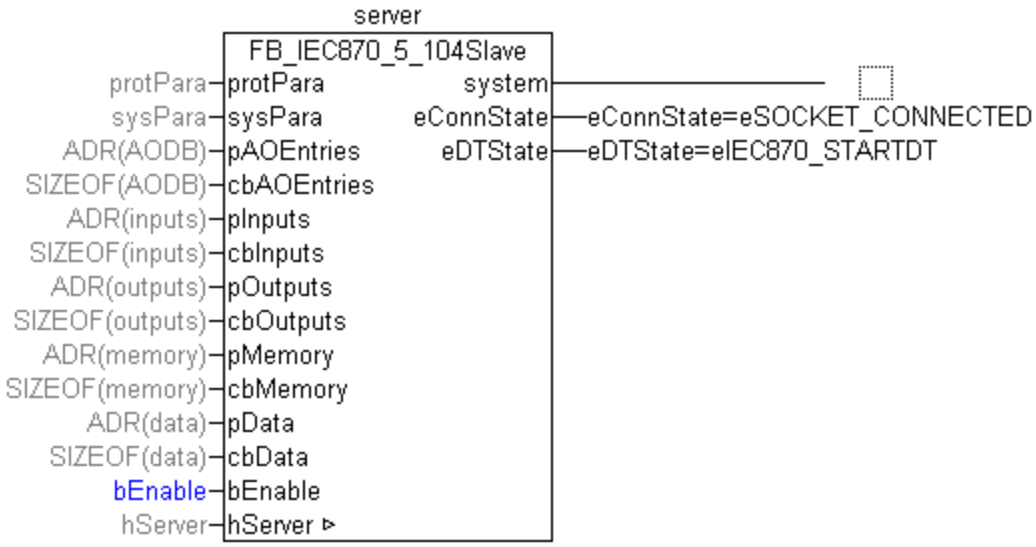
eDTState: status [▶ 449] of the IEC60870-5-104 data exchange (STARTDT, STOPDT)

Example:

Example projects: IEC60870-5-104 controlled station [▶ 479]

Call in FBD:

```
PROGRAM test
VAR
    AODB            : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    inputs AT%IB0   : ARRAY[0..999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..999] OF BYTE;
    memory AT%MB0   : ARRAY[0..999] OF BYTE;
    data        : ARRAY[0..999] OF BYTE;

    hServer         : T_HSERVER;
    server      : FB_IEC870_5_104Slave;


    bEnable         : BOOL := TRUE;
    protPara    : ST_IEC870_5_104ProtocolParams;
    sysPara         : ST_IEC870_5_101SystemParams := ( asduAddr := 7 );
    eConnState      : E_SocketConnectionState;
    eDTState    : E_IEC870_5_104DataTransferState;
    bError      : BOOL;
    iecError    : ST_IEC870_5_101ErrorFifoEntry;
END_VAR
```



In the following structured text example the device error FIFO is read cyclically, and the registered errors are written into the Windows Application Log.

```
REPEAT
    server.system.device.errors.RemoveError( getError=>iecError, bOk=>bError );
    IF bError THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG, 'IEC60870-5-101 slave error: 0x%s', D
WORD_TO_HEXSTR( iecError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT bError
END_REPEAT
```

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104Slave.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_104.Lib, TcIEC870_5_101.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

12.2.3 F_GetVersionTcIEC870_5_104Slave

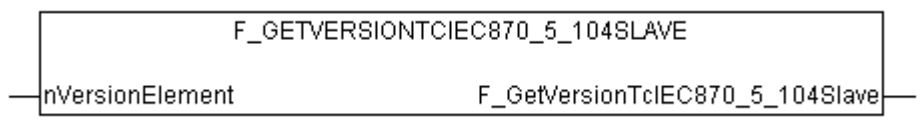


Fig. 9: F_GetVersionTcIEC870_5_104Slave

This function reads version information from the plc library.

FUNCTION F_GetVersionTcIEC870_5_104Slave: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104Slave.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_104.Lib, TcIEC870_5_101.Lib are included automatically)

12.2.4 ST_IEC870_5_104ServerConnection

Product Version: TwinCAT PLC Library: IEC60870-5-104 controlled station (slave) v2.0.2 and higher.

A variable of this type represents a IEC870-5-104 Server connection.

Example for a declaration for a triple connection:

```
connections : ARRAY[0..3] OF ST_IEC870_5_104ServerConnection;
```

Example for a declaration for a single connection:

```
connections : ARRAY[0..0] OF ST_IEC870_5_104ServerConnection;
```

NOTICE

The structure elements should not be written or changed.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104Slave.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib; TcIEC870_5_104.Lib are inculded automatically)

12.2.5 ST_IEC870_5_104GrpStatus

Product Version: TwinCAT PLC Library: IEC60870-5-104 controlled station (slave) v2.0.2 and higher.

Status of an IEC870-5-104 Slave Group.

```
TYPE ST_IEC870_5_104GrpStatus:
STRUCT
    nConnected : DWORD := 0;
    nSuspended : DWORD := 0;
    nDTStarted : DWORD := 0;
END_STRUCT
END_TYPE
```

nConnected: Number of connected TCP/IP Connections (ESTABLISHED).

nSuspended: Number of connections those status changes (CONNECTED->DISCONNECTED oder DISCONNECTED->CONNECTED).

nDTStarted: Number of connections with active data transfer.

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104Slave.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib; TcIEC870_5_104.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

12.2.6 ST_IEC870_5_104SystemInterface

```
TYPE ST_IEC870_5_104SystemInterface :
STRUCT
    device  : ST_IEC870_5_101DeviceInterface;
    service : ST_IEC870_5_104SlaveServices;
END_STRUCT
END_TYPE
```

device: Communication interface [▶ 319] of IEC device.

service: IEC device service;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_104Slave.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_104.Lib, TcIEC870_5_101.Lib are included automatically)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

12.3 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.).

2. Compare the interoperability check list of controlled station and control station.

3. Check the software installation hints described in this documentation (e.g. installation of CAB files on CE platform).

4. In the event of connection problems the PING command can be used to ascertain whether the external communication partner can be reached via the network connection. If this is not the case, check the network configuration and firewall settings.

5. Check the input parameters that are transferred to the F_CreateServerHnd() function (network address, port number etc.) for correctness.

6. Check whether the function block issues an error code. The documentation for the error codes can be found here: Overview of error codes.

7. Check the protocol parameters [▶ 446] that are transferred to the function block (iK, iW, t0, t1, t2, t3, APDULength etc.). Compare them to the parameters set in the controlling station.

8. Check the system parameter [▶ 317] that are transferred to the function block (ASDU address, ASDU address octet size, information object address octet size, cause of transfer octet size, etc.). Compare them to the the parameters set in the controlling station.

9. Check the configured data points (type, information object addresses etc.).

10. Check if the controlling station issues an error code.

11. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings.

12. Check the usage of FB_SocketCloseAll() function block and the usage of the LISTEN_MODE_CLOSEALL parameter in your PLC application.
If your application is working with more than one TCP/IP connections (server/clients) than you have to use one instance of FB_SocketCloseAll() function block to close old/opened connections. Activate this function block instance only once at program start and don't use the LISTEN_MODE_CLOSEALL parameter.

13. Sniffer tools such as Wireshark enable logging of the entire network communication. The log can then be analyzed by Beckhoff support staff.

12.4 Examples

Used controlled station configuration parameters:

- Local (server) host address: **127.0.0.1** (you have to adapt at least this parameter to your target system!!)
- Local (server) port addres: **2404**
- k: **12**
- w: **8**
- t0: **30s**
- t1: **15s**
- t2: **10s**
- t3: **20s**
- Cause of transfer size: **2 octets**
- Common ASDU address size: **2 octets**
- Information object address size: **3 octets**
- Originator address: **1**
- Common ASDU address: **7**
- Max. APDU length: **253**

Requirements

PLC project	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737010443/.zip	TwinCAT IEC60870-5-104 controlled station with one master connection. Very **small** PLC project with one single point in monitoring (IOA = 100) and one single command in control direction (IOA = 10).
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737009035/.zip	TwinCAT IEC60870-5-104 controlled station with one master connection. More complex PLC project with different data points in both directions.

PLC project	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11737011851/.zip	TwinCAT IEC60870-5-104 controlled station with two master connections. More complex PLC project with different data points in both directions.

13 TcIEC870_5_104Master: IEC 60870-5-104 Control Station (master)

Control stations (master) according to IEC60870-5-104 can be realised in TwinCAT PLC with PLC functions and function blocks

The PLC library has two software interfaces. The end application is imposed on one of these interfaces. The choice of interface depends on the requirements for the end application. The characteristics of both interfaces are described briefly below.

'High level' interface: IEC 60870-5-104 Control Station

This interface is a so-called ‘single-block solution’. All functions are encapsulated in one PLC block. The block implements the most important services and functions. This implementation is sufficient for over 90% of applications.

Pro: Very little PLC programming work is required in order to create an application; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. are already implemented in the block and are executed automatically; the mapping of the IEC<->PLC process data and that of the data points is configured via function calls; the PLC programmer does not need to be very well acquainted with the protocol standard;

Contra: The PLC application has only a small influence on the execution of the protocol; no influence on the execution of the services – these are automatically implemented internally; time stamps are automatically generated by the block and cannot be changed (handed over by externals); only the direct command execution, for example, is possible; poorer performance if there are many data points.

This interface is recommended if you:

- are not familiar with the protocol standard;
- are implementing a simple application with few data points (<1000);
- are not placing any great performance demands on the application;
- are not sending any special command execution such as Select/Execute or data + time stamp from external devices;
- do not require any functions that are not supported according to the compatibility list;

'Low level' interface: IEC 60870-5-104 Transport Interface

This interface starts lower down on the protocol stack and enables the sending and/or receipt of individual frames (ASDUs).

Pro: Very flexible; all properties in the ASDU frame can be changed (e.g. an own time stamp, Select/Execute or a special command execution etc.); high performance can be achieved because only the necessary services are implemented; high performance if there are many data points;

Contra: Larger amount of programming work; all services, such as general query, counter query, time synchronisation, command execution, spontaneous data transmission etc. must be implemented by the PLC programmer himself (programmed out); the PLC programmer must be familiar with the protocol standard.

This interface is recommended if you:

- are familiar with the protocol standard;
- are implementing a protocol converter application;
- are implementing virtually all available standard functions in the application;

- are using special functions, such as the relaying of the time stamp from a Modbus device or the gaining of control over the command execution;
- require functions that are not supported according to the compatibility list;
- have many data points (>1000) and need high performance;

Interoperability check list

for TwinCAT PLC Library: IEC 60870-5-104 control station (relating to the "high level" interface). Here you can https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11764581515/.zip

System requirements

If you use the TwinCAT PLC Library IEC60870-5-104 control station and the TwinCAT PLC Library IEC60870-5-104 or 101 substation together in one programming environment:

The Control station is only compatible with the substation v3.0.0 or higher. The reason: Some PLC libraries are used together by the control station and the substation.

Development environment:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT Installation Level:TwinCAT PLC or higher;
- TwinCAT System version 2.10.0 Build >= 1301 or higher;
- Target system:
- Industrial PC or Embedded PC/CX (x86, ARM);
- Operating system:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86) (image v1.75 or higher);
  - Windows CE (ARM) (image v2.13 or higher);
- TwinCAT PLC runtime system version 2.9.0 or higher;

Product components

- **TcIEC870_5_104Master.Lib** (implements the Beckhoff IEC60870-5-104 control station). This library have to be included in the PLC project. All other libraries are included automatically.
- TcIEC870_5_104.Lib (implements the transfer protocol according to IEC60870-5-104);
- TcIEC870_5_101.Lib (implements the connection functions and common data types);
- TcSocketHelper.Lib (TCP/IP help functions);
- TcpIp.Lib (TCP/IP base functions);
- TwinCAT TCP/IP Connection Server;

Installation on Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

The PLC libraries are copied into ..\TwinCAT\Plc\Lib folder. The TwinCAT TCP/IP Connection Server is entered in the TwinCAT Server list. The TCP/IP Connection Server is automatically started when TwinCAT is started and stopped when TwinCAT is stopped.

Installation on Windows CE

Product version for the runtime system under Windows CE is available as separate product. If you have CE version, please do the following steps:

- Install the product on your programming PC. The PLC libraries are copied into ..\TwinCAT\Plc\Lib folder.
- X86 CPU (CX1000, CX1020, IPC):
  - The folder: **...\TwinCAT\CE\TCPIP\Install** contains a Cabinet-File for the CE runtime system.
  - Copy the file: **TcTCPIPSvrCe.I586.CAB** in a folder of the CE runtime system.
- ARM CPU (CX9000):
  - The folder: **...\TwinCAT\CE\TCPIP\Install** contains a Cabinet-File for the CE runtime system.
  - Copy the file: **TcTCPIPSvrCe.ARMV4I.CAB** in a folder of the CE runtime system.
- On CE system: Install (double click to Cabinet-File) CE components.
- Please suspend the CE device once after installation via "Start-> Suspend". The TwinCAT TCP/IP Connection Server starts with CE operating system.

This document is not full product manual. Please install the full version of the Beckhoff Information System.

You will find it

- on the Beckhoff Product DVDs
- on our Web-Server: http://www.beckhoff.com under download.

Examples

Example projects are in the Beckhoff Information System documentation of TwinCAT PLC library.

Link to 'high level' example overview page: IEC 60870-5-104 Control Station [▶ 500];

Link to 'low level' example overview page: IEC 60870-5-104 Transport Interface [▶ 450];

Further Documentation

- Documentation of TwinCAT PLC Library ('low level' interface): IEC 60870-5-104 Transport Interface [▶ 443];
- Documentation of TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 271];
- Documentation: TwinCAT TCP/IP Connection Server;
- IEC 60870-5-1 Transmisstion frame formats;
- IEC 60870-5-2 Link transmission procedures;
- IEC 60870-5-3 General structure of application data;
- IEC 60870-5-4 Definition and coding of application information elements;
- IEC 60870-5-5 Basic application functions;
- IEC 60870-5-101 Companion Standard for basic telecontrol tasks;
- IEC 60870-5-101:1995/A1:2000 Companion Standard for basic telecontrol tasks. Amendment 1;
- IEC 60870-5-101:1995/A2:2001 Companion Standard for basic telecontrol tasks. Amendment 2;
- IEC 60870-5-104:2000 Network access for IEC 60870-5-101 using standard transport profiles;

13.1 Introduction (tutorial)

This introduction explains how to implement and configure an IEC60870-5-104 controlling station (master) within the TwinCAT PLC.

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

13.1.1 Creating a PLC project and integrating PLC libraries

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

- Start TwinCAT PLC Control.

- Create a new PLC project via File -> New. Select PC or CX (x86 or ARM) as the target system.

- A new MAIN program block will now be created automatically. Select ST (Structured Text) as the language for the block. Confirm.

- From the menu select Window -> Library Management and then Insert -> Further Library...

- Select **TcIEC870_5_104Master.Lib** from the list of TwinCAT libraries and confirm.

13.1.2 Defining and configuring an application object database of controlling station

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

Application objects = single points, double points, measured values, short floating point values, etc.

In this example the commands were configured such that the process data for the commands are located in the same memory area as the information data in monitoring direction, although in a different byte/bit offset. If required the commands may have the same byte/bit offset as the information in monitoring direction.

Example:
C_SC_NA_1 with IOA = 10 on the same byte/bit offset as M_SP_NA_1 with IOA = 100 (both byte offset = 100 and bit offset = 0). In this case a change in value of the single point in the substation (M_SP_NA_1 with object address 100) would trigger a data transfer with cause of transmission <3> (spontaneous) to the control station. In the control station the new value is copied to the bit/byte offset address of the command (C_SC_NA_1 with object address 10) and triggers a command transfer to the substation.

As an example we will configure the following application objects as part of the introductory project:

ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
M_SP_NA_1	100	General interrogation group global	0	Memory	100	0	1 bit
M_SP_NA_1	101	General interrogation group global	0	Memory	100	1	1 Bit
M_SP_TB_1	102	General interrogation group global	0	Memory	100	2	1 Bit
M_DP_NA_1	200	General interrogation group global	0	Memory	200	0	2 Bits
M_DP_NA_1	201	General interrogation group global	0	Memory	200	2	2 Bits

ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
M_DP_TB_1	202	General interrogation group global	0	Memory	200	4	2 Bits
M_ST_NA_1	300	General interrogation group global	0	Memory	300	0	1 Byte
M_ST_NA_1	301	General interrogation group global	0	Memory	301	0	1 Byte
M_ST_TB_1	302	General interrogation group global	0	Memory	302	0	1 Byte
M_BO_NA_1	400	General interrogation group global	0	Memory	400	0	4 Byte
M_BO_NA_1	401	General interrogation group global	0	Memory	404	0	4 Byte
M_BO_TB_1	402	General interrogation group global	0	Memory	408	0	4 Byte
M_ME_NA_1	500	General interrogation group global	0	Memory	500	0	2 Byte
M_ME_NA_1	501	General interrogation group global	0	Memory	502	0	2 Byte
M_ME_TD_1	502	General interrogation group global	0	Memory	504	0	2 Byte
M_ME_NB_1	600	General interrogation group global	0	Memory	600	0	2 Byte
M_ME_NB_1	601	General interrogation group global	0	Memory	602	0	2 Byte
M_ME_TE_1	602	General interrogation group global	0	Memory	604	0	2 Byte
M_ME_NC_1	700	General interrogation group global	0	Memory	700	0	4 Byte
M_ME_NC_1	701	General interrogation group global	0	Memory	704	0	4 Byte
M_ME_TF_1	702	General interrogation group global	0	Memory	708	0	4 Byte
M_IT_NA_1	800	General counter interrogation	0	Memory	800	0	4 Byte

ASDU identifier	Object address IOA	Group configuration parameter	Base time multiplier	PLC process data area	Byte offset	Bit offset	Process data width in the TwinCAT PLC
M_IT_NA_1	801	General counter interrogation	0	Memory	804	0	4 Byte
M_IT_TB_1	802	General counter interrogation	0	Memory	808	0	4 Byte
Commands							
C_SC_NA_1	10	-	0	Memory	2100	0	1 Bit
C_SC_NA_1	11	-	0	Memory	2100	1	1 Bit
C_SC_TA_1	12	-	0	Memory	2100	2	1 Bit
C_DC_NA_1	20	-	0	Memory	2200	0	2 Bit
C_DC_NA_1	21	-	0	Memory	2200	2	2 Bit
C_DC_TA_1	22	-	0	Memory	2200	4	2 Bit
C_BO_NA_1	40	-	0	Memory	2400	0	4 Byte
C_BO_NA_1	41	-	0	Memory	2404	0	4 Byte
C_BO_TA_1	42	-	0	Memory	2408	0	4 Byte
C_SE_NA_1	50	-	0	Memory	2500	0	2 Byte
C_SE_NA_1	51	-	0	Memory	2502	0	2 Byte
C_SE_TA_1	52	-	0	Memory	2504	0	2 Byte
C_SE_NB_1	60	-	0	Memory	2600	0	2 Byte
C_SE_NB_1	61	-	0	Memory	2602	0	2 Byte
C_SE_TB_1	62	-	0	Memory	2604	0	2 Byte
C_SE_NC_1	70	-	0	Memory	2700	0	4 Byte
C_SE_NC_1	71	-	0	Memory	2704	0	4 Byte
C_SE_TC_1	72	-	0	Memory	2708	0	4 Byte

Declaring a database variable

The application object database is an array variable of type ST_IEC870_5_101AODBEntry [▶ 313]. Each array element corresponds to an application object.

The array elements are not manipulated directly but through specially provided functions and a database handle (table handle). The database handle must be initialized via a F_iecCreateTableHnd [▶ 295] function call before it can be used. During this process the array elements are linked as a hash table. With a larger number of data points the hash table enables faster access to individual data points.

The maximum number of application objects is freely selectable and is only limited by the available memory. During PLC programming you have to specify a constant maximum number. The maximum number of application objects cannot be changed at runtime. In our example 50 application objects are declared. This number is sufficient for most applications. Please note that many application objects require adequate memory and runtime resources.

Define the following variable in MAIN:

```
PROGRAM MAIN
VAR
     AODB  : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable : T_HAODBTable;
END_VAR
```

Configuring application objects

The object type (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 etc.), the object address and further object parameters are specified during configuration of the individual application objects.

After initialization of the database handle is the application object database (database array) is empty and must be filled with the required data (data points). The data point configuration for the central station must match the data point configuration for the substation, i.e. in the central station data points of the same type and with the same common ASDU address and the same information object address must be configured as in the substation. Other parameters such as the mapping range and byte/bit offset can be configured as required.

The following functions are available for manipulating the application database:

Function	Description
F_iecCreateTableHnd [▶ 295]	Initializes the hash table handle
F_iecAddTableEntry [▶ 296]	Configures and adds a new hash table entry
F_iecRemoveTableEntry [▶ 301]	Removes a hash table entry
F_iecLookupTableEntry [▶ 300]	Checks if a special hash table entry exists
F_iecGetPosOfTableEntry [▶ 298]	Determines the linear position of a hash table entry

The required application objects are configured during program runtime. The data base handle is transferred to the function via VAR_IN_OUT. Configuration is usually carried out once during PLC program start-up via an Init routine.

To configure the application objects during program start-up, add the following PLC code in MAIN:

```
PROGRAM MAIN
VAR
    AODB  : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable: T_HAODBTable;

     init        : BOOL := TRUE;
    initError           : UDINT;
    asduAddr        : UDINT := 7;
END_VAR
```

```
IF init THEN
    init := FALSE;

    initError := F_iecCreateTableHnd( ADR( AODB ), SIZEOF( AODB ), hTable );
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
            'F_iecCreateTableHnd() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF


  (* Monitored Single Points *)
 initError := F_iecAddTableEntry( M_SP_NA_1, asduAddr, 100, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
100, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_SP_NA_1, asduAddr, 101, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 100, 1, 0, hTable );
    initError := F_iecAddTableEntry( M_SP_TB_1, asduAddr, 102, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 100, 2, 0, hTable );
  (* Double Points*)
 initError := F_iecAddTableEntry( M_DP_NA_1, asduAddr, 200, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
200, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_DP_NA_1, asduAddr, 201, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 200, 2, 0, hTable );
    initError := F_iecAddTableEntry( M_DP_TB_1, asduAddr, 202, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 200, 4, 0, hTable );
  (* Regulating step value *)
 initError := F_iecAddTableEntry( M_ST_NA_1, asduAddr, 300, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
300, 0, 0, hTable );
```

```
    initError := F_iecAddTableEntry( M_ST_NA_1, asduAddr, 301, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 301, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ST_TB_1, asduAddr, 302, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 302, 0, 0, hTable );
  (* 32 bit string *)
 initError := F_iecAddTableEntry( M_BO_NA_1, asduAddr, 400, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
400, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_BO_NA_1, asduAddr, 401, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 404, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_BO_TB_1, asduAddr, 402, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 408, 0, 0, hTable );
  (* Measured value, normalized value *)
 initError := F_iecAddTableEntry( M_ME_NA_1, asduAddr, 500, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
500, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_NA_1, asduAddr, 501, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 502, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_TD_1, asduAddr, 502, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 504, 0, 0, hTable );
  (* Mesured value, scaled value *)
 initError := F_iecAddTableEntry( M_ME_NB_1, asduAddr, 600, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
600, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_NB_1, asduAddr, 601, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 602, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_TE_1, asduAddr, 602, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 604, 0, 0, hTable );
  (* Measured value , short floating point value *)
 initError := F_iecAddTableEntry( M_ME_NC_1, asduAddr, 700, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMORY,
700, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_NC_1, asduAddr, 701, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 704, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_ME_TF_1, asduAddr, 702, IEC870_GRP_INROGEN, 0, MAP_AREA_MEMOR
Y, 708, 0, 0, hTable );
  (* Integrated totals *)
 initError := F_iecAddTableEntry( M_IT_NA_1, asduAddr, 800, IEC870_GRP_REQCOGEN, 0, MAP_AREA_MEMORY,
 800, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_IT_NA_1, asduAddr, 801, IEC870_GRP_REQCOGEN, 0, MAP_AREA_MEMO
RY, 804, 0, 0, hTable );
    initError := F_iecAddTableEntry( M_IT_TB_1, asduAddr, 802, IEC870_GRP_REQCOGEN, 0, MAP_AREA_MEMO
RY, 808, 0, 0, hTable );

  (* Single commands *)
 initError := F_iecAddTableEntry( C_SC_NA_1, asduAddr, 10, 0, 0, MAP_AREA_MEMORY, 2100, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SC_NA_1, asduAddr, 11, 0, 0, MAP_AREA_MEMORY, 2100, 1, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_SC_TA_1, asduAddr, 12, 0, 0, MAP_AREA_MEMORY, 2100, 2, 0, hTa
ble );
  (* Double commands *)
 initError := F_iecAddTableEntry( C_DC_NA_1, asduAddr, 20, 0, 0, MAP_AREA_MEMORY, 2200, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_DC_NA_1, asduAddr, 21, 0, 0, MAP_AREA_MEMORY, 2200, 2, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_DC_TA_1, asduAddr, 22, 0, 0, MAP_AREA_MEMORY, 2200, 4, 0, hTa
ble );
  (* 32 bit string commands *)
 initError := F_iecAddTableEntry( C_BO_NA_1, asduAddr, 40, 0, 0, MAP_AREA_MEMORY, 2400, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_BO_NA_1, asduAddr, 41, 0, 0, MAP_AREA_MEMORY, 2404, 0, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_BO_TA_1, asduAddr, 42, 0, 0, MAP_AREA_MEMORY, 2408, 0, 0, hTa
ble );
  (* Set point, normalized values*)
 initError := F_iecAddTableEntry( C_SE_NA_1, asduAddr, 50, 0, 0, MAP_AREA_MEMORY, 2500, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SE_NA_1, asduAddr, 51, 0, 0, MAP_AREA_MEMORY, 2502, 0, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_SE_TA_1, asduAddr, 52, 0, 0, MAP_AREA_MEMORY, 2504, 0, 0, hTa
ble );
  (* Set point, scaled valuess *)
 initError := F_iecAddTableEntry( C_SE_NB_1, asduAddr, 60, 0, 0, MAP_AREA_MEMORY, 2600, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SE_NB_1, asduAddr, 61, 0, 0, MAP_AREA_MEMORY, 2602, 0, 0, hTa
ble );
    initError := F_iecAddTableEntry( C_SE_TB_1, asduAddr, 62, 0, 0, MAP_AREA_MEMORY, 2604, 0, 0, hTa
ble );
  (* Set point, short floating point values *)
 initError := F_iecAddTableEntry( C_SE_NC_1, asduAddr, 70, 0, 0, MAP_AREA_MEMORY, 2700, 0, 0, hTable
 );
    initError := F_iecAddTableEntry( C_SE_NC_1, asduAddr, 71, 0, 0, MAP_AREA_MEMORY, 2704, 0, 0, hTa
ble );
```

```
    initError := F_iecAddTableEntry( C_SE_TC_1, asduAddr, 72, 0, 0, MAP_AREA_MEMORY, 2708, 0, 0, hTa
ble );

END_IF
```

13.1.3 Mapping of PLC and IEC process data

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

The TwinCAT PLC process data are cyclically mapped (copied) into the IEC process data (application objects) and vice versa at program runtime. For mapping of the IEC<->PLC process data up to 4 process data areas (IO inputs, IO outputs, memory area, data area) can be declared as buffer variables in the PLC program. The byte size of the buffers is freely selectable and may be different for each area. Its not necessary to declare unused buffer areas.

In our introductory example we declare 4 PLC process data areas with 3000 bytes each:

```
PROGRAM MAIN
VAR
    AODB    : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable  : T_HAODBTable;

    init    : BOOL := TRUE;
    initError       : UDINT;
    asduAddr    : UDINT := 7;


     inputs AT%IB0   : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0   : ARRAY[0..2999] OF BYTE;
    memory AT%MB0    : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;

END_VAR
```

How the process data are to be mapped at runtime is specified during configuration of the application objects via the F_iecAddTableEntry [▶ 296] function.
See also in: Declaration and configuration of application objects [▶ 484].

The buffer variables were now declared as byte arrays. To improve access to the required data we define the individual variables a second time and allocate them to the corresponding byte/bit offset addresses. In the event of a change in the byte array the corresponding variable is changed at the same time and vice versa, although this is not compulsory. The bytes/bits of the byte array buffer variables can be accessed directly.

```
VAR_GLOBAL
(* Memory offset 0..99 unused *)
  (* Single points *)
 msgSingle_0     AT%MX100.0 : BOOL;
    msgSingle_1     AT%MX100.1 : BOOL;
    msgSingle_2     AT%MX100.2 : BOOL;

  (* Double points *)
    (*      Bit 0..1 = first double point,
        Bit 2..3 = second double point,
        Bit 4..5 = third double point,
        Bit 6..7 = fourth double point *)
    msgDouble_0     AT%MB200    : BYTE;

  (* Regulating step values *)
    msgStep_0           AT%MB300    : BYTE;
    msgStep_1           AT%MB301    : BYTE;
    msgStep_2           AT%MB302    : BYTE;

  (* 32 bit strings *)
 msgBitStr_0     AT%MD400    : DWORD;
    msgBitStr_1     AT%MD404    : DWORD;
    msgBitStr_2     AT%MD408    : DWORD;

  (* Measured values, normalized values *)
 msgNormalized_0     AT%MW500    : WORD;
    msgNormalized_1     AT%MW502    : WORD;
    msgNormalized_2     AT%MW504    : WORD;
```

```
  (* Mesured values, scaled values *)
 msgScaled_0     AT%MW600    : INT;
    msgScaled_1     AT%MW602    : INT;
    msgScaled_2     AT%MW604    : INT;

  (* Measured values, short floating point values *)
 msgFloating_0   AT%MD700    : REAL;
    msgFloating_1   AT%MD704    : REAL;
    msgFloating_2   AT%MD708    : REAL;

  (* Integrated totals *)
 msgTotal_0      AT%MD800    : UDINT;
    msgTotal_1      AT%MD804    : UDINT;
    msgTotal_2      AT%MD808    : UDINT;

  (* Single commands *)
 cmdSingle_0     AT%MX2100.0 : BOOL;
    cmdSingle_1     AT%MX2100.1 : BOOL;
    cmdSingle_2     AT%MX2100.2 : BOOL;

  (* Double commands *)
    (*      Bit 0..1 = first double command,
        Bit 2..3 = second double command,
        Bit 4..5 = third double command,
        Bit 6..7 = fourth double command *)
 cmdDouble_0     AT%MB2200       : BYTE;


  (* 32 bit string commands *)
 cmdBitStr_0     AT%MD2400       : DWORD;
    cmdBitStr_1     AT%MD2404       : DWORD;
    cmdBitStr_2     AT%MD2408       : DWORD;

  (* Set point, normalized values *)
 cmdNormalized_0     AT%MW2500       : WORD;
    cmdNormalized_1     AT%MW2502       : WORD;
    cmdNormalized_2     AT%MW2504       : WORD;

  (* Set point, scaled values *)
 cmdScaled_0     AT%MW2600       : INT;
    cmdScaled_1     AT%MW2602       : INT;
    cmdScaled_2     AT%MW2604       : INT;

  (* Set point, short floating point values *)
 cmdFloating_0   AT%MD2700       : REAL;
    cmdFloating_1   AT%MD2704       : REAL;
    cmdFloating_2   AT%MD2708       : REAL;
END_VAR
```

Mapping of the IEC<->PLC process data in the controlling station

Process data in monitoring direction (slave->master information)

Example 1

Single point information (M_SP_NA_1) with the IOA = 100, PLC memory area, byte offset = 100, bit offset = 0.

Controlled station -> ... -> Controlling station FB -> memory[100].0 == msgSingle_0

Example 2

Measured value, short floating point value (M_ME_NC_1) with the IOA = 700, PLC memory area, byte offset = 700, bit offset = 0 (irrelevant).

Controlled station -> ... -> Controlling station FB -> memory[700..703] == msgFloating_0

Process data in control direction (master->slave commands)

Example 1

Single command state (C_SC_NA_1) with the IOA = 10, PLC memory area, byte offset = 2100, bit offset = 0.

cmdSingle_0 == memory[2100].0 -> Controlling station FB -> ... -> Controlled station

Example 2

Set point, short floating point value (C_SE_NC_1) with the IOA = 70, PLC memory area, byte offset = 2700, bit offset = 0 (irrelevant).

cmdFloating_0 == memory[2700..2703] -> Controlling station FB -> ... -> Controlled station

13.1.4 Declaring and calling an instance of the IEC60870-5-104 controlling station

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

The complete functionality of a controlling station is encapsulated in the FB_IEC870_5_104Master function block. An instance can be used to establish a connection to the controlled station. For establishing a further connection a further instance of the function block can be declared.

Add the following PLC code to the declaration part of MAIN:

```
PROGRAM MAIN
VAR
    AODB        : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable      : T_HAODBTable;

    init        : BOOL := TRUE;
    initError       : UDINT;
    asduAddr        : UDINT := 7;

    inputs AT%IB0   : ARRAY[0..2999] OF BYTE;
    outputs AT%QB0  : ARRAY[0..2999] OF BYTE;
    memory AT%MB0   : ARRAY[0..2999] OF BYTE;
    data        : ARRAY[0..2999] OF BYTE;


     bEnable          : BOOL := TRUE;
    client      : FB_IEC870_5_104Master;

ND_VAR
```

and the instance is called in the program part:

```
IF init THEN
    init := FALSE;
...
ELSE
...

    client(
        pAOEntries := ADR( AODB ),
        cbAOEntries := SIZEOF( AODB ),
        pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
        cbOutputs := SIZEOF( outputs ),
        pMemory := ADR( memory ),
        cbMemory := SIZEOF( memory ),
        pData := ADR( data ),
        cbData := SIZEOF( data ),
        bEnable := bEnable,
        hTable := hTable );

...
END_IF
```

13.1.5 IEC60870-5-104 protocol parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

The behaviour of the controlling station can be adapted to the requirements of the slave via the IEC60870-5-104 protocol parameters. Most parameters have preallocated default values that do not have to be changed.

In our example we change the values of the iK and iW parameters and configure the IP address and port number of the slave station.

```
IF init THEN
    init := FALSE;
...

    client.protPara.sRemoteHost := '127.0.0.1';
    client.protPara.nRemotePort := 2404;
    client.protPara.iK := 12;
    client.protPara.iW := 8;
    client.protPara.bThrottleMode := TRUE;
    client.protPara.bPackFrames := TRUE;


ELSE
    client( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all transfer protocol parameters can be found here: ST_IEC870_5_104PotocolParams [▶ 446].

13.1.6 System parameters

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

The common ASDU address and the user functions of the controlling station are configured via the system parameters.

In our introduction we configure the following system parameters:

- The common ASDU address is set to 7. (*asduAddr*)
- Logging of debugging messages in the application log is activated (*dbgMode*). Logging of device state changes is enabled.

Add the following PLC code to your PLC project:

```
IF init THEN
    init := FALSE;
...


    client.sysPara.asduAddr := 7;
    client.sysPara.dbgMode := IEC870_DEBUGMODE_DEVSTATE (* IEC870_DEBUGMODE_ASDU OR IEC870_DEBUGMODE
_LINKERROR OR IEC870_DEBUGMODE_LINKLAYER *);

...

ELSE
    client( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

The documentation for all system parameters can be found here: ST_IEC870_5_101SystemParams [▶ 317].

13.1.7 Initialisation sequence

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

```
client.acqPara.arrInitSeq[0] := eIEC870_ISTEP_TEST; (* Send test command *)
client.acqPara.arrInitSeq[1] := eIEC870_ISTEP_CLOCK; (* Send clock synchronization command *)
client.acqPara.arrInitSeq[2] := eIEC870_ISTEP_GENRO; (* Send general interrogation command *)
client.acqPara.arrInitSeq[3] := eIEC870_ISTEP_CORO; (* Send counter interrogation command *)
client.acqPara.arrInitSeq[5] := eIEC870_ISTEP_UNUSED; (* not used *)
```

13.1.8 Station interrogation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

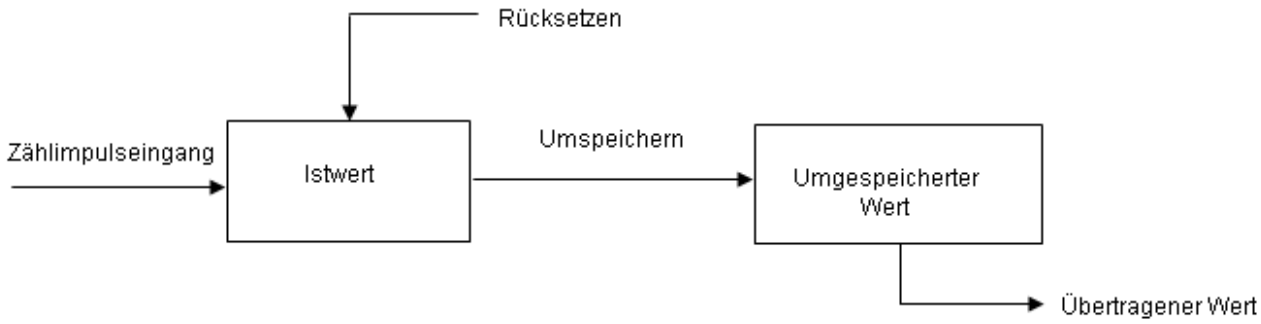
The station interrogation command is initiated by the central station. The ID field of the command also contains the group (1 to 16 or general). The substation transfers the application objects associated with this group to the central station with cause of transmission <*20*> to <*36*>. Application objects with time tags are transferred without time tags.

```
client.acqPara.arrGenro[0].tPollCycle := T#60s;
client.acqPara.arrGenro[0].eQOI := eIEC870_QOI_INROGEN;
client.acqPara.arrGenro[0].bEnable := TRUE;
```

13.1.9 Transfer of integrated totals (counter interrogation)

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

General count transfer model:



The actual values are added by counters. Via a re-store command that is either received by the central station or generated locally (in the substation), the actual values can be re-stored (copied) periodically into re-stored values. After freezing, the recorded value is either reset to zero (logging of incremental values), or the counter continues adding up (logging of counter readings).

Application objects with counts are assigned to groups. The groups are frozen individually, reset, or transferred. The central station sends count query commands to the substation. The task to be carried out (FRZ) and the group (RQT) are specified in an ID field of the command (QCC).

The allocation of the application objects to the individual groups (1 to 4 or general) is specified via the group flag parameter during configuration. There are four operating modes for recording counter readings and incremental values. Each mode includes notes about the configuration of the system parameters or the application objects.

Mode A: local freeze with spontaneous transfer

The substation internally initiates freeze or freeze with reset. The frozen counts are transferred spontaneously, once the freeze or freeze with reset function was executed. In this mode the central station does not issue any count query commands.

Configuration of the system parameters:

Configuration of the application objects:

Mode B: Local freeze with counter interrogation

The substation internally initiates freeze or freeze with reset. The central station queries the frozen counts via count query commands. In this case the central station must not use freeze or freeze with reset in the command ID field (FRZ=0). The counts are queried generally or in groups 1 to 4.

Configuration of the system parameters:

Configuration of the application objects:

Mode C: The central station initiates freeze, freeze with reset, or reset

The central station periodically issues a count query command to the substation for controlling the freeze or (and) reset process. This command does not result in a count transfer. The central station sends a subsequent count query command for collecting the frozen counts. This is similar to operating mode B.

Configuration of the system parameters:

Configuration of the application objects:

Mode D: The central station initiates freeze and (or) reset, and the frozen values are transferred spontaneously

This mode is a combination of the count command from the central station as in mode C and spontaneous transfer of the counts as in mode A.

Configuration of the system parameters:

Configuration of the application objects:

13.1.10 Clock (time) synchronisation

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

Under construction...

13.1.11 Testing the communication

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

You can simulate the command transmission. Setting the *bExecuteCmd* variable to TRUE enables and to FALSE disables the simulation. In our example single command (C_SC_NA_1, IOA = 10) is send cyclicaly every 10 seconds to the controlled station.

```
PROGRAM MAIN
VAR
    ...

    bExecuteCmd : BOOL;
    timer : TON;

    ...
END_VAR
```

```
...

(* Simple command simulation *)
timer( IN := bExecuteCmd, PT := T#10s ); (* Send cyclic command *)
IF timer.Q THEN
    timer( IN := FALSE );
    cmdSingle_0 := NOT cmdSingle_0; (* toggle single command ON<->OFF *)

(*      cmdDouble_0 := SEL( cmdDouble_0 = 1, 1, 2 );

    cmdBitStr_0 := cmdBitStr_0 + 1;

    cmdNormalized_0 := cmdNormalized_0 + 2;

    cmdScaled_0 := cmdScaled_0 + 4;

    cmdFloating_0 := cmdFloating_0 + 1.2; *)
END_IF

...
```

13.1.12 Protocol and data transmission errors

Here you can unpack the complete PLC sources: https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip

The station error messages are placed into a FIFO. Up to 10 error messages can be stored. At fatal communication errors (e.g. error of the linking layer, the check sum of the frame doesn't fit) the connection is cut and has to be build on new. Errors within the application layer (e.g. the ADSU send buffer is overflowed by to many frames) are only logged and doesn't cut the connection.

For this errors it is also possible to cut the connection out of the application. In addition to the error code also the error source is notified in the error message. This simplifies the localization of an error.

Example

The occurring error messages of an IEC60870-5-104 master station station can be read out via the request:

```
PROGRAM MAIN
VAR
...

    client : FB_IEC870_5_104Master;

...
END_VAR
```

```
...

REPEAT
    client.system.device.errors.RemoveError( );
    IF client.system.device.errors.bOk THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'IEC60870-5-104 master error: 0x%s',
            DWORD_TO_HEXSTR( client.system.device.errors.getError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT client.system.device.errors.bOk
END_REPEAT

...
```

13.2 Quick start

Simple projects with complete sources can be found here: IEC60870-5-104 central station [▶ 500].

Interoperability check list can be found here: Interoperability check list

Communication- and/or device error code overview can be found here: Error codes

Detailed introduction to implementation of controlled station in TwinCAT PLC can be found here: TUTORIAL [▶ 483]

Short guide

Application object database

The application object database of the central station must be configured as hash table with the function F_iecCreateTableHnd [▶ 295]. The individual array elements are linked with each other in the form of a hash table. This enables faster access to the individual data points, but also has certain disadvantages that must be taken into account:

- The size of the application database (array size) must not modify at runtime (e.g. through online change). The central station stops execution immediately and reports an error. The reason: The hash table links no longer match. When the program is modified it is best to load the complete project into the runtime system.
- The individual array elements must not be accessed via the index but via the special functions (e.g. F_iecAddTableEntry [▶ 296] etc.).

- With indexed table element access the internal configuration parameters must not be overwritten or modified. If the type, the ASDU address or the object address is changed the data point can no longer be found. To reconfigure a data point it should first be removed from the table via function call F_iecRemoveTableEntry [▶ 301]. The new data point can then be added.

An implementation in the form of linear table would mean that for each received ASDU (data unit) the central station would have to search the complete array for a suitable element. With many data points this would lead to long execution times.

Protocol parameters

The TCP/IP transport layer characteristics are configured via the protocol parameters. Most protocol parameters are preconfigured with default values and do not have to be set explicitly. However, the PLC application has to configure the IP address (*sRemoteHost*) and the port address (*nRemotePort*) of the substation as a one-off operation.

System parameters

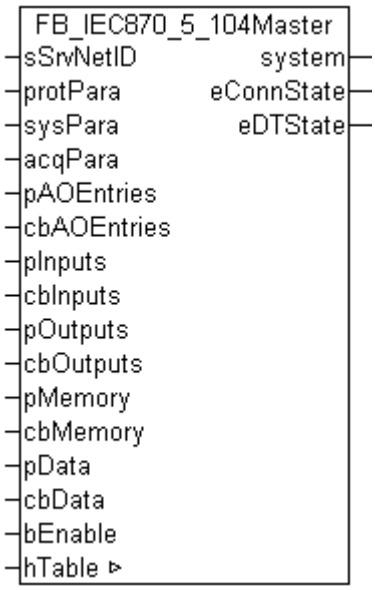
The system parameters are also preconfigured with default values. During commissioning it is useful to activate debugging output (*dbgMode*) in order to be able to locate any errors.

Parameters for cyclic data acquisition

The following parameters are preconfigured with default values:

- Initialisation sequence (consisting of a test command, time synchronisation, station query and counter query);
- Cyclic commands:
  - Test command every 60 s;
  - Time synchronization every 60 s;
  - Group station query: generally, every 60 s;
  - Group counter query: generally, every 60 s;

13.2.1 FB_IEC870_5_104Master



An instance of the FB_IEC870_5_104Master function block can be used to implement an IEC60870-5-104 central station (master) in the TwinCAT PLC. A connection to the slave is established for each instance of the function block.

The function block features the following tasks:

- **STARTDT** (starts the data exchange);
- **STOPDT** (stops the data exchange);

Normally the data exchange is started automatically once the connection is established. This is the default configuration of the function block. If required the data exchange can be stopped or started by calling the actions.

VAR_IN_OUT

```
VAR_IN_OUT
    hTable : T_HAODBTable;
END_VAR
```

hTable: Application object database handle [▶ 343] (hash table handle). The table handle must be initialized once with the function F_iecCreateTableHnd [▶ 295] before it can be used.

VAR_INPUT

```
VAR_INPUT
    sSrvNetID       : T_AmsNetID := '';
    protPara        : ST_IEC870_5_104ProtocolParams := ( bControlDT := TRUE,
                          bDTControlled := FALSE,
                          sRemoteHost := '',
                          nRemotePort := 2404 );
    sysPara         : ST_IEC870_5_101SystemParams := ( bEndOfInit := FALSE,
                          asduAddr := 11,
                          tSyncTimeut := T#0s );
    acqPara         : ST_IEC870_5_101AcquisitionParams;
    pAOEntries      : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF ST_IEC870_5_101AODBEntry := 0;
    cbAOEntries     : UDINT := 0;
    pInputs         : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbInputs        : UDINT := 0;
    pOutputs        : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbOutputs       : UDINT := 0;
    pMemory         : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbMemory        : UDINT := 0;
    pData           : POINTER TO ARRAY[0..IEC870_ANYSIZE_ARRAY] OF BYTE := 0;
    cbData          : UDINT := 0;
    bEnable         : BOOL := TRUE;
END_VAR
```

sSrvNetID: String containing the network address of the TwinCAT TCP/IP Connection Server. For the local computer (default) an empty string may be specified.

protPara: IEC60870-5-104 protocol parameter [▶ 446].

sysPara: System parameter [▶ 317].

acqPara: Parameter for cyclic data acquisition.

pAOEntries: Address of the application object database [▶ 313] variables.

cbAOEntries: Byte size of the application object database variables.

pInputs: Address of the PLC process data area for the inputs.

cbInputs: Byte size of the PLC process data area for the inputs.

pOutputs: Address of the PLC process data area for the outputs.

cbOutputs: Byte size of the PLC process data area for the outputs.

pMamory: Address of the PLC process data area for the flags.

cbMamory: Byte size of the PLC process data area for the flags.

pData: Address of the PLC data area.

cbData: Byte size of the PLC data area.

bEnable : Activates/deactivates the function block (communication and connections).

The addresses can be determined with the ADR operator and the byte sizes with the SIZEOF operator.

VAR_OUTPUT

```
VAR_OUTPUT
    system      : ST_IEC870_5_104ExSystemInterface;
    eConnState  : E_SocketConnectionState := eSOCKET_DISCONNECTED;
    eDTState    : E_IEC870_5_104DataTransferState := eIEC870_STOPDT;
END_VAR
```

system: system interface [▶ 499]. This variable is used by other IEC application functions as a communication interface for the IEC device (here: central station).

- Member variable *system.device* is expected by the F_iecSetAOQuality [▶ 282] function as VAR_IN_OUT parameter, for example.
- Member variable *system.device.errors* is a device error FIFO. The PLC application can read and analyse registered errors.

eConnState: Status of the TCP/IP connection with the slave..

eDTState: Status [▶ 449] of the IEC60870-5-104 data exchange (STARTDT, STOPDT)

Example in ST:IEC60870-5-104 central station [▶ 500]

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1301	PC or CX (X86) CX (ARM)	TcIEC870_5_104Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcpIp.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib; TcIEC870_5_104.Lib; are included automatically)

13.2.2 F_GetVersionTcIEC870_5_104Master

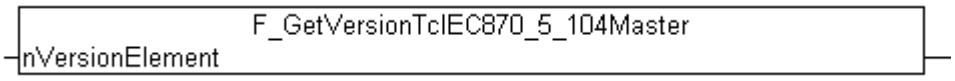


Fig. 10: F_GetVersionTcIEC870_5_104Master

This function reads version information from the plc library.

FUNCTION F_GetVersionTcIEC870_5_104Master: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element, that is to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_104Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib;

Development Environment	Target System	PLC libraries to include
		TcIEC870_5_104.Lib; TcIEC870_5_101.Lib are included automatically)

13.2.3 ST_IEC870_5_104ExSystemInterface

```
TYPE ST_IEC870_5_104ExSystemInterface :
STRUCT
    device   : ST_IEC870_5_101DeviceInterface;
    service  : ST_IEC870_5_101SystemServices;
    hSOTable : T_HAODBTABLE;
END_STRUCT
END_TYPE
```

device: Communication interface [▶ 319] of IEC device;

service: IEC device service;

hSOTable : System object database handle [▶ 343];

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_104Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_104.Lib; TcIEC870_5_101.Lib are included automatically)

13.3 Troubleshooting/diagnostics

1. Check the hardware and software requirements described in this documentation (TwinCAT version, CE image version etc.).
2. Compare the interoperability check list of controlled station and control station.
3. Check the software installation hints described in this documentation (e.g. installation of CAB files on CE plattform).
4. In the event of connection problems the PING command can be used to ascertain whether the external communication partner can be reached via the network connection. If this is not the case, check the network configuration and firewall settings.
5. Check the network address and port number parameter that are transferred to the function block for correctness.
6. Check whether the function block issues an error code/source [▶ 495]. The documentation for the error codes can be found here: Overview of error codes.
7. Check the protocol parameters [▶ 446] that are transferred to the function block (iK, iW, t0, t1, t2, t3, APDULength etc.). Compare them to the parameters set in the controlled station.
8. Check the system parameters [▶ 317] that are transferred to the function block (ASDU address, ASDU address octet size, information object address octet size, cause of transfer octet size, etc.). Compare them to the the parameters set in the controlled station.
9. Check the acquisition parameters [▶ 320] that are transferred to the function block (initialization sequence, cyclic general interrogation, cyclic counter interrogation, cyclic test commands, etc.).
10. Check the data point configuration (type, information object addresses etc.).
11. Check if the controlled station issues an error code.
12. Activate the debug output during connection establishment and/or of ASDU data. Open the TwinCAT System Manager and activate the LogView window. Analyze/check the debug output strings.

13. Check the usage of FB_SocketCloseAll() function block and the usage of the LISTEN_MODE_CLOSEALL parameter in your PLC application.
    If your application is working with more than one TCP/IP connections (server/clients) than you have to use one instance of FB_SocketCloseAll() function block to close old/opened connections. Activate this function block instance only once at program start and don't use the LISTEN_MODE_CLOSEALL parameter.
14. Sniffer tools such as Wireshark enable logging of the entire network communication. The log can then be analysed by Beckhoff support staff.

13.4 Examples

Used controlling station configuration parameters:

- Remote (server) host address: **127.0.0.1**
- Remote (server) port addres: **2404**
- k: **12**
- w: **8**
- t0: **30s**
- t1: **15s**
- t2: **10s**
- t3: **20s**
- Cause of transfer size: **2 octets**
- Common ASDU address size: **2 octets**
- Information object address size: **3 octets**
- Originator address: **1**
- Common ASDU address: **7**
- Max. APDU length: **253**

Requirements

PLC project	Description
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751839627/.zip	Simple TwinCAT IEC60870-5-104 control station (master). Very **small** PLC project with one single point in monitoring (IOA = 100) and one single command in control direction (IOA = 10).
https://infosys.beckhoff.com/content/1033/TS650x_tcplc_iec60870-5-10x/Resources/11751838219/.zip	TwinCAT IEC60870-5-104 control station (master)

14 Error codes

Overview

Codes (hex)	Codes (dec)	Error source	Description
0x00000000-0x00002000	0-8192	Serial communication errors	Serial communication error (only if error source [▶ 332]=eIEC870_ESRC_IEC60870_5_101LINK).
0x00000000-0x00007800	0-30720	TwinCAT System error	TwinCAT System error (ADS error codes inclusive only if error error source <>eIEC870_ESRC_IEC60870_5_101LINK).
0x80070000-0x8007FFFF	2147942400-2148007935	Error source = Code - 0x80070000 = Win32 system error code	0x80070000-0x8007FFFF
0x00008100-0x000081FF	32768-33023	IEC 60870-5-10x error	Internal IEC 60870-5-10x error

IEC 60870-5-10x errors

Code (hex)	Code (dec)	Symbolic constant	Description
0x00008101	33025	IEC870_COMMERR_INVALIDSTARTBYTE	Invalid frame start character <> 0x68
0x00008102	33026	IEC870_COMMERR_RXBUFFOVERFLOW	Receive buffer overflow
0x00008103	33027	IEC870_COMMERR_TXBUFFOVERFLOW	Send buffer overflow
0x00008104	33028	IEC870_COMMERR_INVALIDUFFMT	Invalid U-Frame format, more than one function (STARTDT, STOPDT, TESTFR) activated
0x00008105	33029	IEC870_COMMERR_INVALIDSFFMT	Invalid S-Frame format, invalid length parameter
0x00008106	33030	IEC870_COMMERR_T1RESPONSE	t1 (response timeout) expired
0x00008107	33031	IEC870_COMMERR_SENDSEQ	Send sequence error
0x00008108	33032	IEC870_COMMERR_KOVERFLOW	k reached
0x00008109	33033	IEC870_COMMERR_FATALERR	Fatal internal error
0x0000810A	33034	IEC870_COMMERR_INVALIDSTATE	Device is in invalid state (disconnected?)
0x0000810B	33035	IEC870_COMMERR_INVALIDSIZE	Invalid parameter size
0x0000810C	33036	IEC870_COMMERR_INVALIDVALUE	Invalid parameter value
0x0000810D	33037	IEC870_COMMERR_INVALIDTYPE	Invalid asdu (object) type
0x0000810F	33039	IEC870_COMMERR_TIMEOUT	Communication timeout
0x00008110	33040	IEC870_COMMERR_LENGTH1	Invalid length field value
0x00008111	33041	IEC870_COMMERR_LENGTH2	Length field and length field copy differs
0x00008112	33042	IEC870_COMMERR_STARTCHAR2	Invalid second stat character

Code (hex)	Code (dec)	Symbolic constant	Description
0x00008113	33043	IEC870_COMMERR_CHECKSUM	Invalid checksum
0x00008114	33044	IEC870_COMMERR_ENDCHAR	Invalid end character
0x00008115	33045	IEC870_COMMERR_LINKADDR	Invalid link address size
0x00008116	33046	IEC870_COMMERR_SRVFUNCODE	Invalid link service function code
0x00008117	33047	IEC870_COMMERR_FRAMETYPE	Invalid frame type
0x00008118	33048	IEC870_COMMERR_UNSUPMODE	Unsupported comunication mode (balanced mode)
0x00008119	33049	IEC870_COMMERR_T2KOVERFLOW	k reached and t2 (response timeout) expired
0x0000811A	33050	IEC870_COMMERR_INVALIDCONFIG	Invalid object configuration/initialization
0x0000811B	33051	IEC870_COMMERR_UNKNOWNTYPE	Unknown asdu type
0x0000811C	33052	IEC870_COMMERR_UNKNOWNCOT	Unknown cause of transfer
0x0000811D	33053	IEC870_COMMERR_UNKNOWNASDUADDR	Unknown asdu address
0x0000811E	33054	IEC870_COMMERR_UNKNOWNOBJADDR	Unknown object address
0x0000811F	33055	IEC870_COMMERR_NEGACTCON	Negative activation confirmation
0x00008120	33056	IEC870_COMMERR_NEGACTTERM	Negative activation termination
0x00008121	33057	IEC870_COMMERR_NEGDEACTCON	Negative deactivation confirmation
0x00008122	33058	IEC870_COMMERR_BUSY	Allready in busy state
0x00008123	33059	IEC870_COMMERR_AODBOVERFLOW	Application object database overflow
0x00008124	33060	IEC870_COMMERR_AODBNOTFOUND	Application object not in database
0x00008125	33061	IEC870_COMMERR_ACTCONTIMEOUT	Activation confirmation timeout error
0x00008126	33062	IEC870_COMMERR_ACTTERMTIMEOUT	Activation termination timeout error
0x00008127	33063	IEC870_COMMERR_DEACTCONTIMEOUT	Deactivation termination error
0x00008128	33064	IEC870_COMMERR_SELEXECTIMEOUT	Command select/execute timeout error
0x00008129	33065	IEC870_COMMERR_READRESTIMEOUT	Read command response timeout error
0x00008130	33072	IEC870_COMMERR_LIBNOTCOMPAT	Product libraries are incompatible
0x00008131	33073	IEC870_COMMERR_DIR	Invalid DIR bit value
0x00008132	33074	IEC870_COMMERR_PRM	Invalid PRM bit value
0x00008133	33075	IEC870_COMMERR_FCV	Invalid FCV bit value
0x00008134	33076	IEC870_COMMERR_SCANTIMEOUT	Station scan cycle timeout error

15 Glossary

Term	Description
substation, slave, server, controlled station	Synonyms for a subordinate station (which is monitored)
central station, master, client, controlling station	Synonyms for a higher-level station (control station, monitors other stations)
control direction	Data transfer direction from the central station to the substation
monitoring direction	Data transfer direction from the substation to the central station
application objects	IEC information objects in the TwinCAT PLC application (single points, double points, measured values, short floating point values etc.)
APDU	Application protocol data unit
APCI	Application protocol control information
ASDU	Application service data unit
IOA, information object address	Address of the single point, double point, measured value, short floating point value etc.
primary station	Primary station sends commands/requests to the secondary station and controls the data transmission of the secondary station.
secondary station	Secondary station answers to the requests of the primary station.
combined station	Combined stations can act as primary and secondary station (balanced mode).
balanced mode	In balanced mode both stations act as primary and secondary stations and can initialize the data transmission.
unbalanced mode	In unbalanced mode only the primary station is able to initialize the data transmissions. The central station acts as primary station and the substation acts as secondary station.

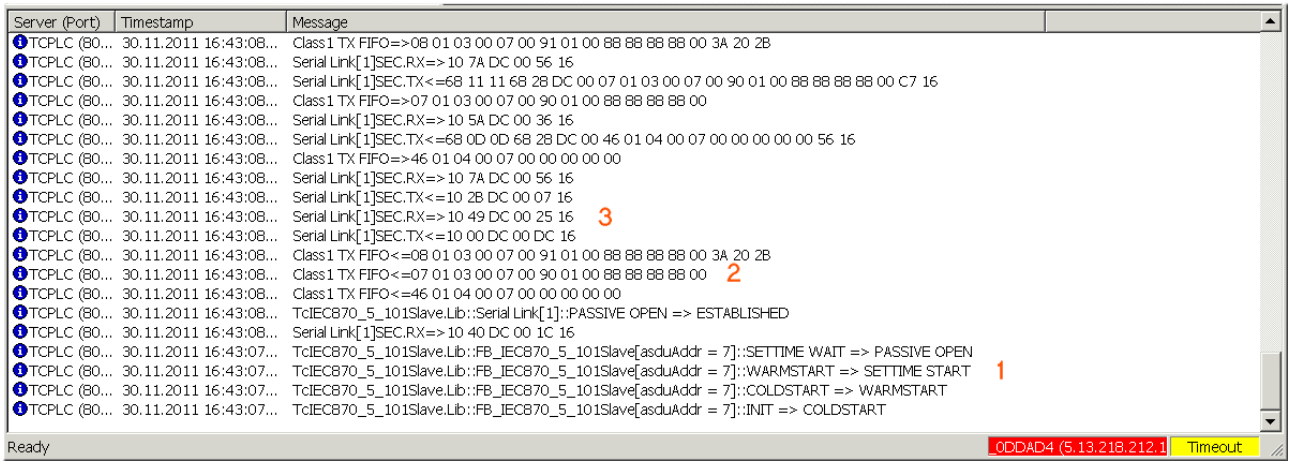
16 Appendix

16.1 Troubleshooting and debugging

Debugging messages written to the application log facilitate troubleshooting of the system. Currently three stages of debugging messages can be activated in an IEC application. These messages can be activated via the dbgMode system parameter of the control station (ST_IEC870_5_101SystemParams [▶ 317]).

1. Station status messages (dbgMode: IEC870_DEBUGMODE_DEVSTATE);
2. Hexadecimal output of the ASDUs (frames without link layer control header, dbgMode: IEC870_DEBUGMODE_ASDU). 32 ASDU data bytes per row are output as hexadecimal numbers. Bigger frames are split into multiple lines;
3. Hexadecimal output of the APDUs (serial frames, dbgMode: IEC870_DEBUGMODE_LINKLAYER). 32 APDU data bytes per row are output as hexadecimal numbers. Bigger frames are split into multiple lines;

Using additional option it is also possible to output the link layer errors (dbgMode: IEC870_DEBUGMODE_LINKERROR). In order to view the debugging messages start the TwinCAT System Manager and activate log view. A debugging output is shown below. The three different message types are identified with the respective numbers.



Further diagnostic tools:

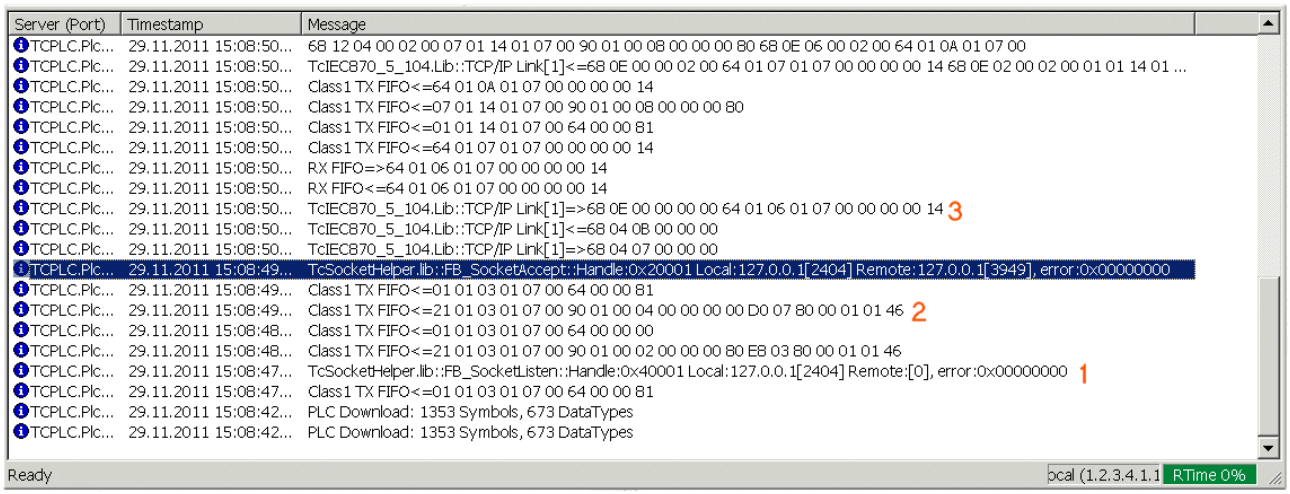
- Portmon for Windows (v3.02, Windows Sysinternals);
- Diverse protocoll test suite products;

16.2 Troubleshooting and debugging

Debugging messages written to the application log facilitate troubleshooting of the system. If the "Low level" interface is used then three stages of debugging messages can be activated in an PLC application:

1. Debugging messages that are logged when the TCP/IP connection is established or released (TcSocketHelper.Lib messages). These messages can be activated via the nMode parameter when the F_CreateServerHnd() function (controlled station) or FB_ClientServerConnection() function block (controlling station) is called;
2. Hexadecimal output of the ASDUs (without link layer control header). 32 ASDU data bytes per row are output as hexadecimal numbers. Bigger frames are split into multiple lines. These messages can be activated via the ST_IEC870_5_101TBuffer member variable: eDbg;
3. Hexadecimal output of the APDUs (TCP/IP frames). 32 APDU data bytes per row are output as hexadecimal numbers. Bigger frames are split into multiple lines. These messages can be activated via the FB_IEC870_5_104TProtocol [▶ 444] input variable: bOutDbg;

In order to view the activated debug messages, start the TwinCAT System Manager and activate log view. A debug output is shown below. The three different message types are identified with the respective numbers.



Further diagnostic tools:

- TwinCAT ADS monitor;
- Network monitor;
- Wireshark;
- Ethereal;
- Diverse protocol test suite products;

16.3 Configuration of serial interface

- Configuration of standard PC COMx port [▶ 505]
- Configuration of serial KL6xxx bus terminal [▶ 507]
- Configuration of serial EL6xxx bus terminal [▶ 507]

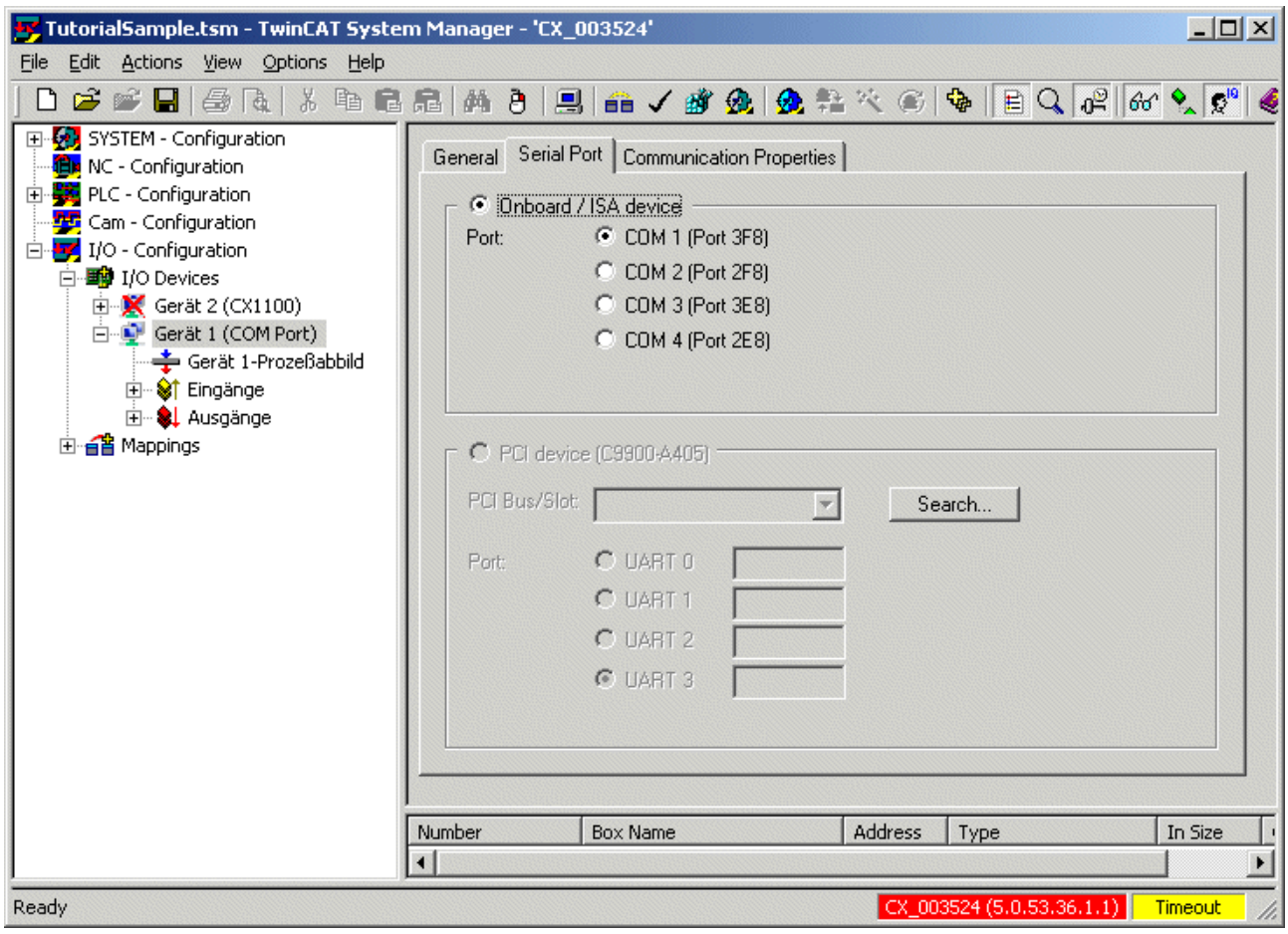
1. Configuration of standard PC COMx port

The serial interface can only be added in PC/CX systems.

Right-click on "I/O devices". Select "Append Device". Select "Serial Interface" under "Miscellaneous". Then set the following settings.

1.1. Serial Port tab settings

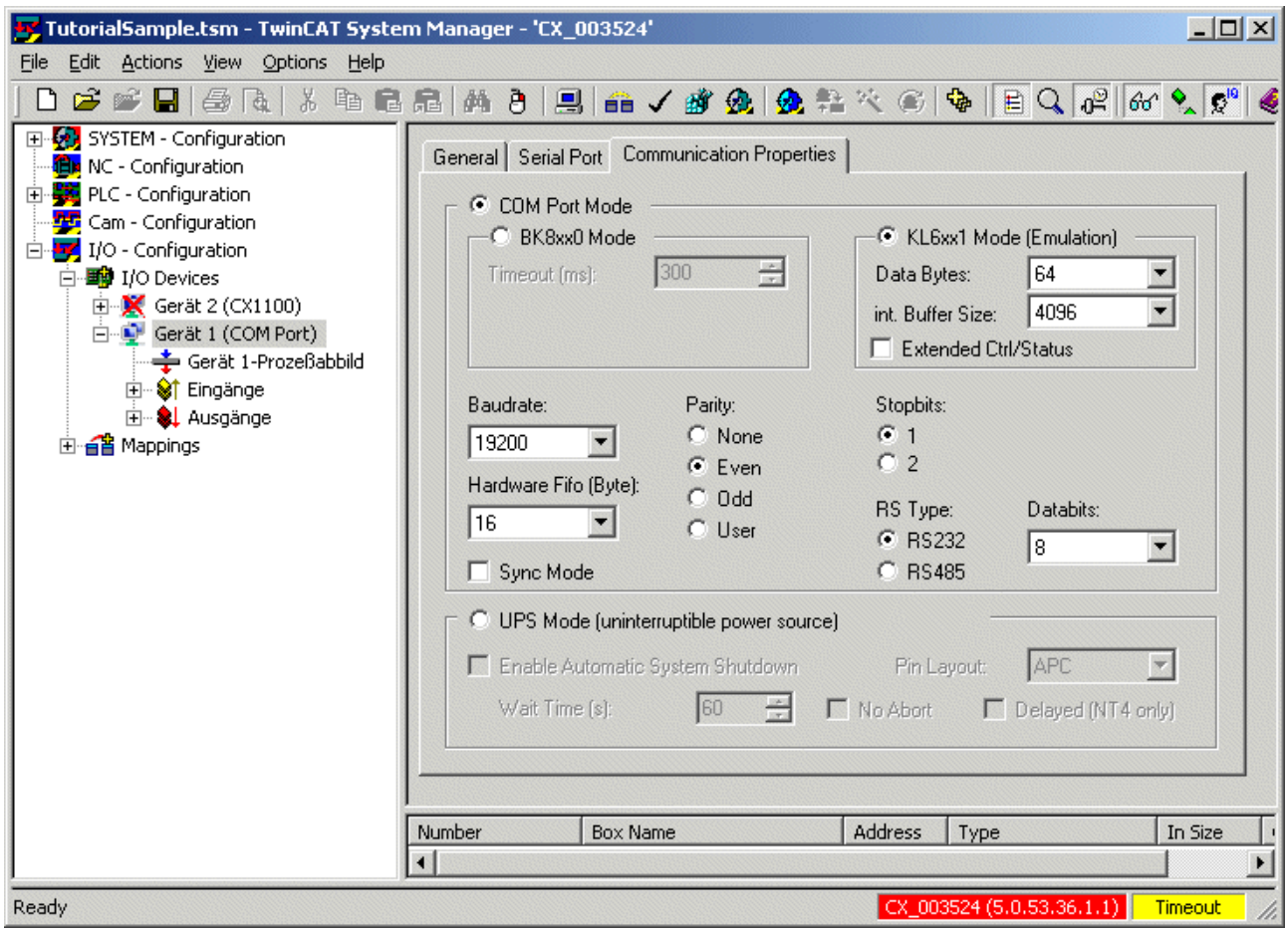
Serial interface COM1



1.2. Communication Properties tab settings

Select the option KL6xxx1 Mode (Emulation) and set the communication parameter:

Baud rate, here 19200 Baud, 8 data bits, Parity = even, 1 stop bit



2. Configuration of serial KL6xxx bus terminal

The interface is configured within the TwinCAT PLC by the instance of the FB_IEC870_SerialLineCtrlfunction block. Communication parameters like baud rate, parity etc. have to be set via this function block.

2.1 RS485 operating mode

In the RS485 operating mode (e.g. KL6041), the data is exchanged by means of half duplex transfer.The transmit and receive lines are connected to each other in RS485 operating mode. As a result, the terminal receives not only data from other devices, but also its own transmitted data. The own data may disturb the communication. For half-duplex mode, the *handshake* input of the FB_IEC870_SerialLineCtrl function block must be set to the value: RS485_HALFDUPLEX.

3. Configuration of serial EL6xxx bus terminal

The settings for the serial interface can be found in the CoE Online tab (CoE=**C**anOpen **o**ver **E**therCAT). The tab is only available for online access to the terminal, i.e. the settings cannot be modified if the System Manager is not linked to the hardware. The settings are stored in the terminal in a non-volatile manner, i.e. the settings are retained even if the voltage supply fails. These data do not have to be re-entered after a terminal scan. If the terminal is replaced, the data from the new terminal become active and have to be adjusted. To avoid this, the modified data can be entered in the *Startup* tab. During system startup, the data from the *Startup* tab are transferred to the terminal. Terminals can then be replaced without problem. After a terminal scan, the data have to be re-entered in the *Startup* tab.

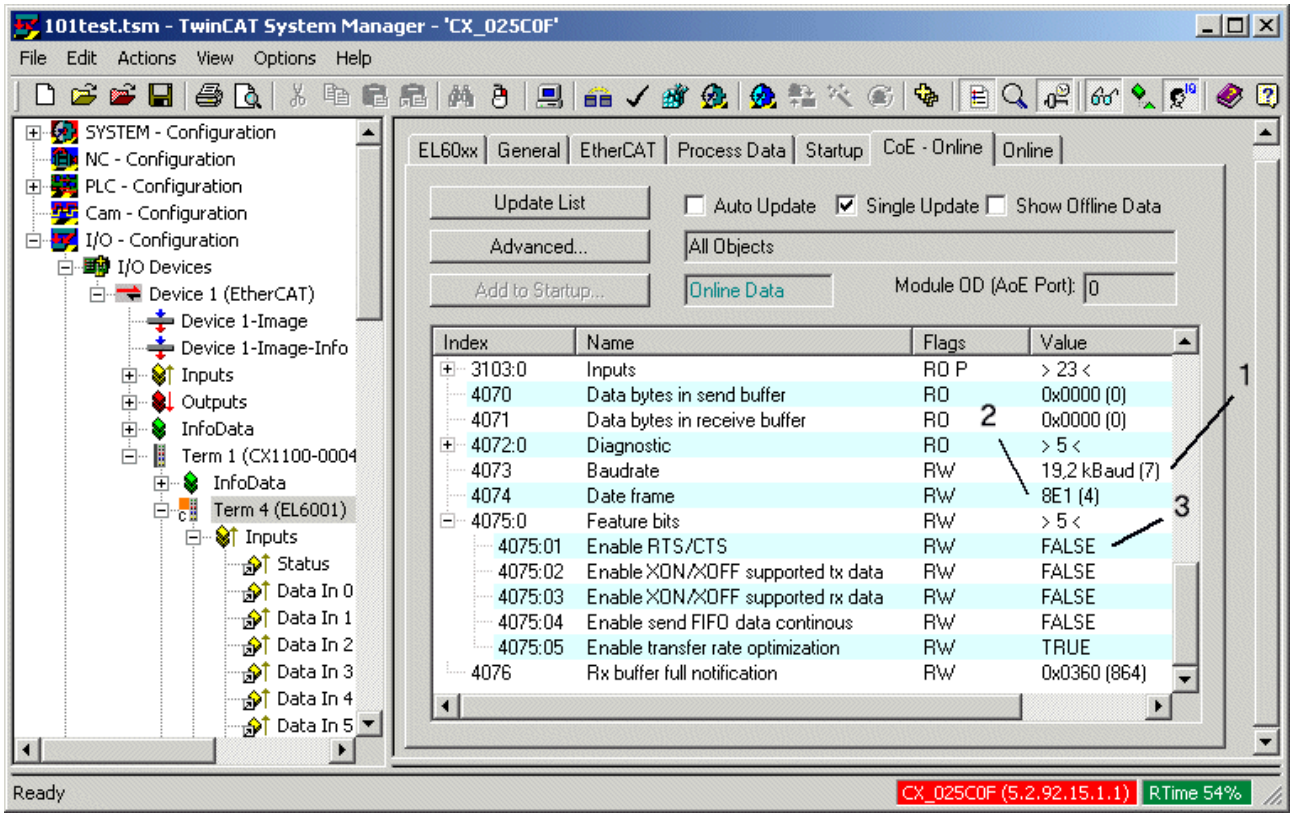
3.1. CoE tab settings

(1) Baud rate, here 19200 Baud

(2) Date frame, 8E1 corresponds to 8 data bits, Parity = even, 1 stop bit

(3) Feature bits -> Enable RTS/CTS = *FALSE* for EL6001

Double-clicking on the corresponding row brings up a menu in which the settings can be modified.



3.1.1 RS485 operating mode

In the RS485 mode, the data is exchanged by means of half duplex transfer. The transmit and receive lines are connected to one another in RS485 operating mode. As a result, the terminal receives not only the data from other devices, but also its own transmitted data. The own data may disturb the communication. This can be suppressed with the option "Enable half duplex" in the "COM Settings" object (index 8000:06).

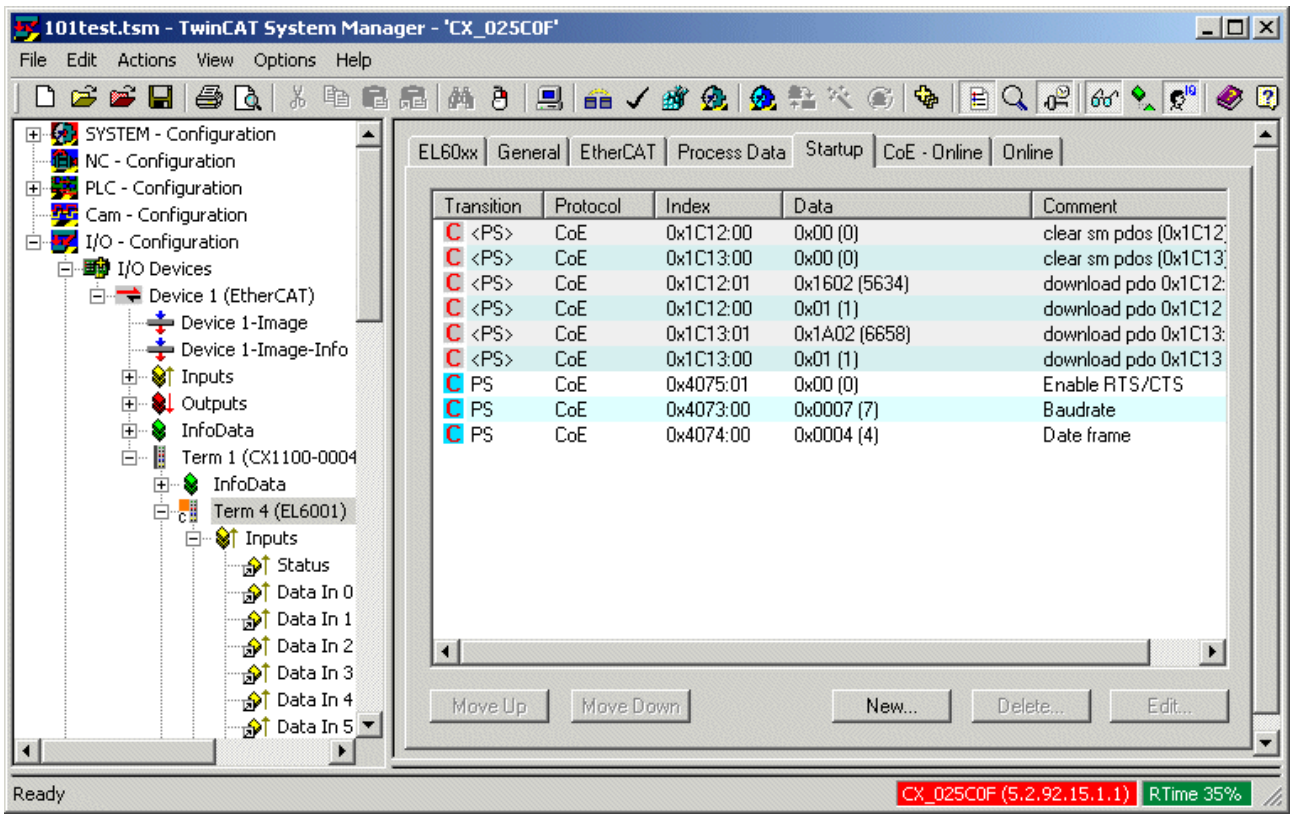
3.2. Startup tab settings

All non-standard settings should be entered in the Startup list. During system startup, the data from the *Startup* tab are transferred to the terminal. Terminals can then be replaced without problem. After a terminal scan, the data have to be re-entered in the *Startup* tab.

Baud rate, here 19200 Baud

Date frame, 8E1 corresponds to 8 data bits, Parity = even, 1 stop bit

Feature bits -> Enable RTS/CTS = *FALSE* for EL6001



16.4 Firewall settings

The TwinCAT PLC library IEC 60870-5-104 sub station (slave/server) uses the TCP/IP protocol for communication. It is therefore essential to make sure that the corresponding TCP port will be activated during the using of a firewall. The table below lists standard ports, that must be considered by using a firewall.

Description	Type	Protocol	Port
IEC 60870-5-104 telecontrol protocol	Protocol	TCP	2404

The configuration of the Windows Firewall is done via the corresponding dialog in the control panel. For further information please see the Windows resp. Firewall documentation.

NOTICE

Remember if you use an Embedded Controller without display and USB: Activate the port for Remote Display (Windows CE) or Remote Desktop (Windows XP / Windows Vista) in the firewall. Otherwise, you have no way to manage the computer via the network.

More Information:
www.beckhoff.com/ts650x

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com