BECKHOFF New Automation Technology

Handbuch | DE

TS650x

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

Supplement | Communication

05.07.2023 | Version: 1.3

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Technische Referenz

In der Fernwirktechnik müssen Geräte verschiedener Hersteller miteinander kommunizieren. Die IEC 60870-5-Familie definiert auf der Basis der fünf Grundnormen IEC 60870-5-1 (Telegrammformate), -2 (Übertragungsprozeduren/Verbindungsschicht), -3 (Strukturen/Anwendungsdaten), -4 (Informationselemente) und -5 (grundlegende Anwendungsfunktionen) die anwendungsbezogenen Normen IEC 60870-5-101, -102, -103 und 104. Durch die international genormten Fernwirkprotokolle IEC 60870-5-101/-102/-103 kann eine serielle bzw. über das Fernwirkprotokoll IEC 60870-5-104 eine TCP/IP-basierte Datenübertragung realisiert werden.

Die Hersteller sind nicht verpflichtet die komplette Norm in den Geräten zu implementieren. Aus diesem Grund kann es zu Inkompatibilitäten zwischen den Geräten bei der Inbetriebnahme kommen. Um dem vorzubeugen, bieten Hersteller zu jedem Gerät eine passende Kompatibilitätsliste an. In dieser Liste sind die implementierten Funktionen aufgelistet bzw. markiert. Mit der Hilfe der Kompatibilitätsliste können die benötigten Funktionalitäten zweier Geräte im Vorfeld verglichen werden. Bitte beachten Sie auch die Kompatibilitätslisten zu den TwinCAT IEC 60870-5-10x-Bibliotheken.

Die SPS-Bibliotheken verfügen teilweise über zwei Software-Schnittstellen ("Low level"- und "High level"-Schnittstelle). Die Endapplikation setzt auf einer dieser Schnittstellen auf. Bei zwei verfügbaren Schnittstellen hängt die Wahl der Schnittstelle von den Anforderungen an die Endapplikation ab. Bitte lesen Sie dazu auch die Hinweise in der jeweiligen Produktinformation. Im Folgenden werden die wichtigsten Eigenschaften der Produkte kurz aufgelistet.

TwinCAT PLC IEC 60870-5-101 Master

- Datenübertragung: Seriell;
- Produktinformation [▶ 18]:
  - Systemvoraussetzungen [▶ 350];
  - Produktkomponenten [▶ 350];
  - Installation [▶ 350];
- Verfügbare Schnittstellen:
  - "High level"-Schnittstelle: IEC 60870-5-101 Zentralstation (implementiert in TcIEC870_5_101Master.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
  - "Low level"-Schnittstelle: IEC 60870-5-101 Serial Link Interface (implementiert in TcIEC870_5_101Link.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
- Realisierbare Endapplikationen:
  - "High level": nur Zentralstation (master);
  - "Low level": Zentralstation und/oder Unterstation (master/slave);
- Schnittstellendokumentation:
  - "High level": TwinCAT PLC Library: IEC 60870-5-101 Zentralstation [▶ 349];
  - "Low level": TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 396];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268] (gemeinsame Datentypen);
  - TwinCAT PLC Library: Serielle Kommunikation;

- Beispiele:
  - "High level": IEC 60870-5-101 Zentralstation [▶ 368];
  - "Low level": IEC 60870-5-101 Serial Link Interface [▶ 413];

TwinCAT PLC IEC 60870-5-101 Slave

- Datenübertragung: Seriell;
- Produktinformation [▶ 19];
  - Systemvoraussetzungen [▶ 371];
  - Produktkomponenten [▶ 371];
  - Installation [▶ 371];
- Verfügbare Schnittstellen:
  - "High level"-Schnittstelle: IEC 60870-5-101 Unterstation (implementiert in TcIEC870_5_101Slave.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
  - "Low level"-Schnittstelle: IEC 60870-5-101 Serial Link Interface (implementiert in TcIEC870_5_101Link.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
- Realisierbare Endapplikationen:
  - "High level": nur Unterstation (slave);
  - "Low level": Zentralstation und/oder Unterstation (master/slave);
- Schnittstellendokumentation:
  - "High level": TwinCAT PLC Library: IEC 60870-5-101 Unterstation [▶ 370];
  - "Low level": TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 396];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268] (gemeinsame Datentypen);
  - TwinCAT PLC Library: Serielle Kommunikation;
- Beispiele:
  - "High level": IEC 60870-5-101 Unterstation [▶ 394];
  - "Low level": IEC 60870-5-101 Serial Link Interface [▶ 413];

TwinCAT PLC IEC 60870-5-102 Master

- Datenübertragung: Seriell;
- Produktinformation [▶ 19];
  - Systemvoraussetzungen [▶ 416];
  - Produktkomponenten [▶ 416];
  - Installation [▶ 417];
- Verfügbare Schnittstellen:
  - "Low level"-Schnittstelle: IEC 60870-5-102 Serial Link Interface (implementiert in TcIEC870_5_102Link.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
- Realisierbare Endapplikationen:
  - "Low level": nur Zentralstation (master);
- Schnittstellendokumentation:
  - "Low level": TwinCAT PLC Library: IEC 60870-5-102 Serial Link Interface [▶ 416];
  - TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 396];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268] (gemeinsame Datentypen);
  - TwinCAT PLC Library: Serielle Kommunikation;

- Beispiele:
  - "Low level": IEC 60870-5-102 Serial Link Interface [▶ 429];

TwinCAT PLC IEC 60870-5-103 Master

- Datenübertragung: Seriell;
- Produktinformation [▶ 20];
  - Systemvoraussetzungen [▶ 431];
  - Produktkomponenten [▶ 431];
  - Installation [▶ 432];
- Verfügbare Schnittstellen:
  - "Low level"-Schnittstelle: IEC 60870-5-103 Serial Link Interface (implementiert in TcIEC870_5_103Link.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
- Realisierbare Endapplikationen:
  - "Low level": nur Zentralstation (master);
- Schnittstellendokumentation:
  - "Low level": TwinCAT PLC Library: IEC 60870-5-103 Serial Link Interface [▶ 431];
  - TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 396];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268] (gemeinsame Datentypen);
  - TwinCAT PLC Library: Serielle Kommunikation;
- Beispiele:
  - "Low level": IEC 60870-5-103 Serial Link Interface [▶ 441];

TwinCAT PLC IEC 60870-5-104 Master

- Datenübertragung: TCP/IP;
- Produktinformation [▶ 20];
  - Systemvoraussetzungen [▶ 482];
  - Produktkomponenten [▶ 482];
  - Installation [▶ 482];
- Verfügbare Schnittstellen:
  - "High level"-Schnittstelle: IEC 60870-5-104 Zentralstation (implementiert in TcIEC870_5_104Master.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
  - "Low level"-Schnittstelle: IEC 60870-5-104 Transport Interface (implementiert in TcIEC870_5_104.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
- Realisierbare Endapplikationen:
  - "High level": nur Zentralstation (master);
  - "Low level": Zentralstation und/oder Unterstation (master/slave);
- Schnittstellendokumentation:
  - "High level": TwinCAT PLC Library: IEC 60870-5-104 Zentralstation [▶ 481];
  - "Low level": TwinCAT PLC Library: IEC 60870-5-104 Transport Interface [▶ 442];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268] (gemeinsame Datentypen);
  - TwinCAT TCP/IP Connection Server;
- Beispiele:
  - "High level": IEC 60870-5-104 Zentralstation [▶ 500];

- "Low level": IEC 60870-5-104 Transport Interface [▶ 449];

TwinCAT PLC IEC 60870-5-104 Slave

- Datenübertragung: TCP/IP;
- Produktinformation [▶ 21];
  - Systemvoraussetzungen [▶ 452];
  - Produktkomponenten [▶ 452];
  - Installation [▶ 452];
- Verfügbare Schnittstellen:
  - "High level"-Schnittstelle: IEC 60870-5-104 Unterstation (implementiert in TcIEC870_5_104Slave.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
  - "Low level"-Schnittstelle: IEC 60870-5-104 Transport Interface (implementiert in TcIEC870_5_104.Lib, diese Bibliothek muss in das SPS-Projekt eingebunden werden);
- Realisierbare Endapplikationen:
  - "High level": nur Unterstation (slave);
  - "Low level": Zentralstation und/oder Unterstation (master/slave);
- Schnittstellendokumentation:
  - "High level": TwinCAT PLC Library: IEC 60870-5-104 Unterstation [▶ 451];
  - "Low level": TwinCAT PLC Library: IEC 60870-5-104 Transport Interface [▶ 442];
  - TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268] (gemeinsame Datentypen);
  - TwinCAT TCP/IP Connection Server;
- Beispiele:
  - "High level": IEC 60870-5-104 Unterstation [▶ 479];
  - "Low level": IEC 60870-5-104 Transport Interface [▶ 449];

3 Telegrammstrukturen

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.
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

LPDU bytes	Explanation
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
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 [▶ 425]								DATA UNIT IDENTIFIER		ASDU		
	SQ	Number of object											
	T	P/N	Cause of transmission [▶ 427]										
	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
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 [▶ 438], control-direction [▶ 439])								DATA UNIT IDENTIFIER		ASDU		
7	SQ	Number of object											
8	Cause of transmission (monitoring-direction [▶ 439], control-direction [▶ 440])												
9	ASDU address												
10	Function type [▶ 266]												
11	Information number [▶ 266]								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
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
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-Objektbeschreibung

4.1 Standard IEC 60870-5-101 Datentypen

Type	Dez	Hex	Beschreibung
ASDU_TYPEUNDEF	0	0x00	Wird nicht verwendet
M_SP_NA_1 [▶ 37]	1	0x01	Einzelmeldung
M_SP_TA_1 [▶ 38]	2	0x02	Einzelmeldung mit einem Zeitstempel
M_DP_NA_1 [▶ 40]	3	0x03	Doppelmeldung
M_DP_TA_1 [▶ 42]	4	0x04	Doppelmeldung mit einem Zeitstempel
M_ST_NA_1 [▶ 46]	5	0x05	Stufenstellungsmeldung
M_ST_TA_1 [▶ 47]	6	0x06	Stufenstellungsmeldung mit einem Zeitstempel
M_BO_NA_1 [▶ 51]	7	0x07	Bitmuster von 32 bit
M_BO_TA_1 [▶ 53]	8	0x08	Bitmuster von 32 bit mit einem Zeitstempel
M_ME_NA_1 [▶ 57]	9	0x09	Messwert, normierter Wert
M_ME_TA_1 [▶ 58]	10	0x0A	Messwert, normierter Wert mit einem Zeitstempel
M_ME_NB_1 [▶ 64]	11	0x0B	Messwert, skalierter Wert
M_ME_TB_1 [▶ 66]	12	0x0C	Messwert, skalierter Wert mit einem Zeitstempel
M_ME_NC_1 [▶ 70]	13	0x0D	Messwert, Gleitkommazahl mit einfacher Genauigkeit
M_ME_TC_1 [▶ 71]	14	0x0E	Messwert, Gleitkommazahl mit einfacher Genauigkeit und einem Zeitstempel
M_IT_NA_1 [▶ 75]	15	0x0F	Zählwerte
M_IT_TA_1 [▶ 77]	16	0x10	Zählwerte mit einem Zeitstempel
M_EP_TA_1 [▶ 188]	17	0x11	Schutzereignis mit einem Zeitstempel
M_EP_TB_1 [▶ 190]	18	0x12	Gepackte Anregungen des Schutzes mit einem Zeitstempel
M_EP_TC_1 [▶ 192]	19	0x13	Gepackte Auslösungen des Schutzes mit einem Zeitstempel
M_PS_NA_1 [▶ 201]	20	0x14	Gepackte Einzelmeldungen mit Zustandsanzeige
M_ME_ND_1 [▶ 62]	21	0x15	Messwert, normierter Wert ohne Qualitätskennung
ASDU_TYPE_22..29	22..29	0x16..0x1D	Reserviert (Standardbereich)
M_SP_TB_1 [▶ 39]	30	0x1E	Einzelmeldung mit einem Zeitstempel CP56Time2a
M_DP_TB_1 [▶ 44]	31	0x1F	Doppelmeldung mit einem Zeitstempel CP56Time2a
M_ST_TB_1 [▶ 49]	32	0x20	Stufenstellungsmeldung mit einem Zeitstempel CP56Time2a
M_BO_TB_1 [▶ 55]	33	0x21	Bitmuster von 32 bit mit einem Zeitstempel CP56Time2a
M_ME_TD_1 [▶ 60]	34	0x22	Messwert, normierter Wert mit einem Zeitstempel CP56Time2a
M_ME_TE_1 [▶ 68]	35	0x23	Messwert, skalierter Wert mit einem Zeitstempel CP56Time2a
M_ME_TF_1 [▶ 73]	36	0x24	Messwert, Gleitkommazahl mit einfacher Genauigkeit und einem Zeitstempel CP56Time2a
M_IT_TB_1 [▶ 79]	37	0x25	Zählwerte mit einem Zeitstempel CP56Time2a
M_EP_TD_1 [▶ 195]	38	0x26	Schutzereignis mit einem Zeitstempel CP56Time2a
M_EP_TE_1 [▶ 197]	39	0x27	Gepackte Anregungen des Schutzes mit einem Zeitstempel CP56Time2a
M_EP_TF_1 [▶ 199]	40	0x28	Gepackte Auslösungen des Schutzes mit einem Zeitstempel CP56Time2a
ASDU_TYPE_41..44	41..44	0x29..0x2C	Reserviert (Standardbereich)
C_SC_NA_1 [▶ 81]	45	0x2D	Einzelbefehl

Type	Dez	Hex	Beschreibung
C_DC_NA_1 [▶ 85]	46	0x2E	Doppelbefehl
C_RC_NA_1 [▶ 88]	47	0x2F	Stufenstellbefehl
C_SE_NA_1 [▶ 92]	48	0x30	Sollwert-Stellbefehl, normierter Wert
C_SE_NB_1 [▶ 96]	49	0x31	Sollwert-Stellbefehl, skalierter Wert
C_SE_NC_1 [▶ 99]	50	0x32	Sollwert-Stellbefehl, Gleitkommazahl mit einfacher Genauigkeit
C_BO_NA_1 [▶ 103]	51	0x33	Bitmusterbefehl von 32 bit
ASDU_TYPE_52..57	52..57	0x34..0x39	Reserviert (Standardbereich)
C_SC_TA_1 [▶ 83]	58	0x3A	Einzelbefehl mit einem Zeitstempel CP56Time2a
C_DC_TA_1 [▶ 86]	59	0x3B	Doppelbefehl mit einem Zeitstempel CP56Time2a
C_RC_TA_1 [▶ 90]	60	0x3C	Stufenstellbefehl mit einem Zeitstempel CP56Time2a
C_SE_TA_1 [▶ 94]	61	0x3D	Sollwert-Stellbefehl, normierter Wert mit einem Zeitstempel CP56Time2a
C_SE_TB_1 [▶ 97]	62	0x3E	Sollwert-Stellbefehl, skalierter Wert mit einem Zeitstempel CP56Time2a
C_SE_TC_1 [▶ 101]	63	0x3F	Sollwert-Stellbefehl, Gleitkommazahl mit einfacher Genauigkeit und einem Zeitstempel CP56Time2a
C_BO_TA_1 [▶ 105]	64	0x40	Bitmusterbefehl von 32 bit mit einem Zeitstempel CP56Time2a
ASDU_TYPE_65..69	65..69	0x41..0x45	Reserviert (Standardbereich)
M_EI_NA_1 [▶ 110]	70	0x46	Initialisierungsende
ASDU_TYPE_71..99	71..99	0x47..0x63	Reserviert (Standardbereich)
C_IC_NA_1 [▶ 122]	100	0x64	(General-, Stations-) Abfragebefehl
C_CI_NA_1 [▶ 124]	101	0x65	Zähler-Abfragebefehl
C_RD_NA_1 [▶ 128]	102	0x66	Abfragebefehl
C_CS_NA_1 [▶ 121]	103	0x67	Uhrzeit-Synchronisationsbefehl
C_TS_NA_1 [▶ 107]	104	0x68	Prüfbefehl
C_RP_NA_1 [▶ 126]	105	0x69	Prozess-Rücksetzbefehl
C_CD_NA_1	106	0x6A	Befehl zur Telegrammlaufzeit-Erfassung
C_TS_TA_1	107	0x6B	Prüfbefehl mit einem Zeitstempel CP56Time2a
ASDU_TYPE_108..109	108..109	0x6C..0x6D	Reserviert (Standardbereich)
P_ME_NA_1 [▶ 203]	110	0x6E	Parameter für Messwerte, normierter Wert
P_ME_NB_1 [▶ 205]	111	0x6F	Parameter für Messwerte, skalierter Wert
P_ME_NC_1 [▶ 207]	112	0x70	Parameter für Messwerte, Gleitkommazahl mit einfacher Genauigkeit
P_AC_NA_1 [▶ 209]	113	0x71	Parameter für Aktivierung
ASDU_TYPE_114..119	114..119	0x72..0x77	Reserviert (Standardbereich)
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	Verzeichniss

Type	Dez	Hex	Beschreibung
ASDU_TYPE_127..255	127..255	0x7F..0xFF	Reserviert (benutzerdefinierter Bereich)

4.2 Standard IEC 60870-5-102 Datentypen

Type	Dez	Hex	Beschreibung
ASDU_TYPEUNDEF_2	0	0x00	Wird nicht verwendet
M_SP_TA_2 [▶ 211]	1	0x01	Einzelmeldung mit Zeitstempel
M_IT_TA_2 [▶ 212]	2	0x02	Abrechnungszählerstände, je 4 Oktette
M_IT_TB_2 [▶ 214]	3	0x03	Abrechnungszählerstände, je 3 Oktette
M_IT_TC_2 [▶ 215]	4	0x04	Abrechnungszählerstände, je 2 Oktette
M_IT_TD_2 [▶ 217]	5	0x05	Periodisch rückgesetzte Abrechnungszählerstände, je 4 Oktette
M_IT_TE_2 [▶ 218]	6	0x06	Periodisch rückgesetzte Abrechnungszählerstände, je 3 Oktette
M_IT_TF_2 [▶ 220]	7	0x07	Periodisch rückgesetzte Abrechnungszählerstände, je 2 Oktette
M_IT_TG_2 [▶ 222]	8	0x08	Betriebszählerstände, je 4 Oktette
M_IT_TH_2 [▶ 223]	9	0x09	Betriebszählerstände, je 3 Oktette
M_IT_TI_2 [▶ 225]	10	0x0A	Betriebszählerstände, je 2 Oktette
M_IT_TK_2 [▶ 226]	11	0x0B	Periodisch rückgesetzte Betriebszählerstände, je 4 Oktette
M_IT_TL_2 [▶ 228]	12	0x0C	Periodisch rückgesetzte Betriebszählerstände, je 3 Oktette
M_IT_TM_2 [▶ 230]	13	0x0D	Periodisch rückgesetzte Betriebszählerstände, je 2 Oktette
	14..69	0x0E,,0x45	Reserviert (Standardbereich)
M_EI_NA_2	70	0x46	Initialisierungsende
P_MP_NA_2	71	0x47	Hersteller- und Produktspezifikation der Zähler-DEE
M_TI_TA_2	72	0x48	Aktuelle Systemzeit der Zähler-DEE
	73..99	0x49..0x63	Reserviert (Standardbereich)
C_RD_NA_2 [▶ 129]	100	0x64	Abruf der Hersteller und Produktspezifikation
C_SP_NA_2 [▶ 131]	101	0x65	Abruf einer Liste von Einzelmeldungen mit Zeitstempel
C_SP_NB_2 [▶ 133]	102	0x66	Abruf einer Liste von Einzelmeldungen mit Zeitstempel eines ausgewählten Zeitbereichs
C_TI_NA_2 [▶ 135]	103	0x67	Abruf der aktuellen Systemzeit der Zähler-DEE
C_CI_NA_2 [▶ 136]	104	0x68	Abruf der Abrechnungszählerstände der ältesten Messperiode
C_CI_NB_2 [▶ 138]	105	0x69	Abruf der Abrechnungszählerstände der ältesten Messperiode und eines ausgewählten Adressbereichs
C_CI_NC_2 [▶ 140]	106	0x6A	Abruf der Abrechnungszählerstände einer bestimmten vergangenen Messperiode
C_CI_ND_2 [▶ 142]	107	0x6B	Abruf der Abrechnungszählerstände einer bestimmten vergangenen Messperiode und eines ausgewählten Adressbereichs
C_CI_NE_2 [▶ 144]	108	0x6C	Abruf periodisch rückgesetzter Abrechnungszählerstände der ältesten Messperiode

Type	Dez	Hex	Beschreibung
C_CI_NF_2 [▶ 146]	109	0x6D	Abruf periodisch rückgesetzter Abrechnungszählerstände der ältesten Messperiode und eines ausgewählten Adressbereichs
C_CI_NG_2 [▶ 148]	110	0x6E	Abruf periodisch rückgesetzter Abrechnungszählerstände einer bestimmten vergangenen Messperiode
C_CI_NH_2 [▶ 150]	111	0x6F	Abruf periodisch rückgesetzter Abrechnungszählerstände einer bestimmten vergangenen Messperiode und eines ausgewählten Adressbereichs
C_CI_NI_2 [▶ 152]	112	0x70	Abruf der Betriebszählerstände der ältesten Messperiode
C_CI_NK_2 [▶ 153]	113	0x71	Abruf der Betriebszählerstände der ältesten Messperiode und eines ausgewählten Adressbereichs
C_CI_NL_2 [▶ 155]	114	0x72	Abruf der Betriebszählerstände einer bestimmten vergangenen Messperiode
C_CI_NM_2 [▶ 157]	115	0x73	Abruf der Betriebszählerstände einer bestimmten vergangenen Messperiode und eines ausgewählten Adressbereichs
C_CI_NN_2 [▶ 159]	116	0x74	Abruf periodisch rückgesetzter Betriebszählerstände der ältesten Messperiode
C_CI_NO_2 [▶ 161]	117	0x75	Abruf periodisch rückgesetzter Betriebszählerstände der ältesten Messperiode und eines ausgewählten Adressbereichs
C_CI_NP_2 [▶ 163]	118	0x76	Abruf periodisch rückgesetzter Betriebszählerstände einer bestimmten vergangenen Messperiode
C_CI_NQ_2 [▶ 165]	119	0x77	Abruf periodisch rückgesetzter Betriebszählerstände einer bestimmten vergangenen Messperiode und eines ausgewählten Adressbereichs
C_CI_NR_2 [▶ 167]	120	0x78	Abruf der Abrechnungszählerstände eines ausgewählten Zeitbereichs und eines ausgewählten Adressbereichs
C_CI_NS_2 [▶ 169]	121	0x79	Abruf periodisch rückgesetzter Abrechnungszählerstände eines ausgewählten Zeitbereichs und eines ausgewählten Adressbereichs
C_CI_NT_2 [▶ 172]	122	0x7A	Abruf der Betriebszählerstände eines ausgewählten Zeitbereichs und eines ausgewählten Adressbereichs
C_CI_NU_2 [▶ 174]	123	0x7B	Abruf periodisch rückgesetzter Betriebszählerstände eines ausgewählten Zeitbereichs und eines ausgewählten Adressbereichs
	124..127	0x7C..0x7F	Reserviert (Standardbereich)
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)

Type	Dez	Hex	Beschreibung
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	Reserviert
C_DS_TA_2	180	0xB4	-
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 time period and a selected address
C_CB_UN_2	190	0xBE	Read a block of periodical reset operational integrated totals of a time period and a selected address
	191..255	0xBF..0xFF	Reserviert

4.3 Standard IEC 60870-5-103 Datentypen

In Überwachungsrichtung	In Steuerungsrichtung	Dez	Hex	Beschreibung
M_TYPEUNDEF_3	C_TYPEUNDEF_3	0	0x00	Wird nicht verwendet
M_TTM_TA_3 [▶ 231]	-	1	0x01	Meldung mit Zeitstempel
M_TMR_TA_3 [▶ 233]	-	2	0x02	Meldung mit Zeitstempel und Relativzeit
M_MEI_NA_3 [▶ 235]	-	3	0x03	Messwerte I
M_TME_TA_3 [▶ 238]	-	4	0x04	Echtzeitmesswerte mit Relativzeit
M_IRC_NA_3 [▶ 112]	-	5	0x05	Identifikationsmeldung
M_SYN_TA_3 [▶ 116]	C_SYN_TA_3 [▶ 176]	6	0x06	Zeitsynchronisierung
-	C_IGI_NA_3 [▶ 178]	7	0x07	Generalabfrage-Initialisierung

In Überwachungsrichtung	In Steuerungsrichtung	Dez	Hex	Beschreibung
M_TGI_NA_3 [▶ 114]	-	8	0x08	Generalabfrage-Ende
M_MEII_NA_3 [▶ 240]	-	9	0x09	Messwerte II
M_GD_XA_3 [▶ 119]	C_GD_NA_3 [▶ 179]	10	0x0A	Generische Daten
M_GI_XA_3 [▶ 117]	-	11	0x0B	Generische Identifikation
-	-	12..19	0x0C..0x13	Reserviert (Standardbereich)
-	C_GRC_NA_3 [▶ 181]	20	0x14	Allgemeiner Befehl
-	C_GC_NA_3 [▶ 183]	21	0x15	Generischer Befehl
-	-	22	0x16	Reserviert (Standardbereich)
M_LRD_TA_3 [▶ 243]	-	23	0x17	Störfallübersicht
-	C_ODT_NA_3 [▶ 184]	24	0x18	Auftrag zur Übertragung der Stördaten
-	C_ADT_NA_3 [▶ 186]	25	0x19	Quittung für Stördatenübertragung
M_RTD_TA_3 [▶ 245]	-	26	0x1A	Bereit zur Übertragung von Stördaten
M_RTC_NA_3 [▶ 247]	-	27	0x1B	Bereit zur Übertragung eines Kanals
M_RTT_NA_3 [▶ 249]	-	28	0x1C	Bereit zur Übertragung von Marken
M_TOT_NA_3 [▶ 251]	-	29	0x1D	Übertragung von Marken
M_TOV_NA_3 [▶ 252]	-	30	0x1E	Übertragung von Störwerten
M_EOT_NA_3 [▶ 254]	-	31	0x1F	Ende der Übertragung
-	-	32..255	0x1F..0xFF	Reserviert (benutzerdefinierter Bereich)

4.4 Standard IEC 60870-5-104 Datentypen

Type	Dez	Hex	Beschreibung
ASDU_TYPEUNDEF	0	0x00	Wird nicht verwendet
M_SP_NA_1	1	0x01	Einzelmeldung
M_SP_TA_1	2	0x02	Einzelmeldung mit einem Zeitstempel
M_DP_NA_1	3	0x03	Doppelmeldung
M_DP_TA_1	4	0x04	Doppelmeldung mit einem Zeitstempel
M_ST_NA_1 [▶ 262]	5	0x05	Stufenstellungsmeldung
M_ST_TA_1	6	0x06	Stufenstellungsmeldung mit einem Zeitstempel
M_BO_NA_1	7	0x07	Bitmuster von 32 bit
M_BO_TA_1	8	0x08	Bitmuster von 32 bit mit einem Zeitstempel

Type	Dez	Hex	Beschreibung
M_ME_NA_1	9	0x09	Messwert, normierter Wert
M_ME_TA_1	10	0x0A	Messwert, normierter Wert mit einem Zeitstempel
M_ME_NB_1	11	0x0B	Messwert, skalierter Wert
M_ME_TB_1	12	0x0C	Messwert, skalierter Wert mit einem Zeitstempel
M_ME_NC_1	13	0x0D	Messwert, Gleitkommazahl mit einfacher Genauigkeit
M_ME_TC_1	14	0x0E	Messwert, Gleitkommazahl mit einfacher Genauigkeit und einem Zeitstempel
M_IT_NA_1	15	0x0F	Zählwerte
M_IT_TA_1	16	0x10	Zählwerte mit einem Zeitstempel
M_EP_TA_1	17	0x11	Schutzereignis mit einem Zeitstempel
M_EP_TB_1	18	0x12	Gepackte Anregungen des Schützes mit einem Zeitstempel
M_EP_TC_1	19	0x13	Gepackte Auslösungen des Schutzes mit einem Zeitstempel
M_PS_NA_1	20	0x14	Gepackte Einzelmeldungen mit Zustandsanzeige
M_ME_ND_1	21	0x15	Messwert, normierter Wert ohne Qualitätskennung
ASDU_TYPE_22..29	22..29	0x16..0x1D	Reserviert (Standardbereich)
M_SP_TB_1	30	0x1E	Einzelmeldung mit einem Zeitstempel CP56Time2a
M_DP_TB_1	31	0x1F	Doppelmeldung mit einem Zeitstempel CP56Time2a
M_ST_TB_1 [▶ 264]	32	0x20	Stufenstellungsmeldung mit einem Zeitstempel CP56Time2a
M_BO_TB_1	33	0x21	Bitmuster von 32 bit mit einem Zeitstempel CP56Time2a
M_ME_TD_1	34	0x22	Messwert, normierter Wert mit einem Zeitstempel CP56Time2a
M_ME_TE_1	35	0x23	Messwert, skalierter Wert mit einem Zeitstempel CP56Time2a
M_ME_TF_1	36	0x24	Messwert, Gleitkommazahl mit einfacher Genauigkeit und einem Zeitstempel CP56Time2a
M_IT_TB_1	37	0x25	Zählwerte mit einem Zeitstempel CP56Time2a
M_EP_TD_1	38	0x26	Schutzereignis mit einem Zeitstempel CP56Time2a
M_EP_TE_1	39	0x27	Gepackte Anregungen des Schutzes mit einem Zeitstempel CP56Time2a
M_EP_TF_1	40	0x28	Gepackte Auslösungen des Schutzes mit einem Zeitstempel CP56Time2a
ASDU_TYPE_41..44	41..44	0x29..0x2C	Reserviert (Standardbereich)
C_SC_NA_1	45	0x2D	Einzelbefehl
C_DC_NA_1	46	0x2E	Doppelbefehl
C_RC_NA_1	47	0x2F	Stufenstellbefehl
C_SE_NA_1	48	0x30	Sollwert-Stellbefehl, normierter Wert
C_SE_NB_1	49	0x31	Sollwert-Stellbefehl, skalierter Wert
C_SE_NC_1	50	0x32	Sollwert-Stellbefehl, Gleitkommazahl mit einfacher Genauigkeit
C_BO_NA_1	51	0x33	Bitmusterbefehl von 32 bit
ASDU_TYPE_52..57	52..57	0x34..0x39	Reserviert (Standardbereich)
C_SC_TA_1	58	0x3A	Einzelbefehl mit einem Zeitstempel CP56Time2a
C_DC_TA_1	59	0x3B	Doppelbefehl mit einem Zeitstempel CP56Time2a
C_RC_TA_1	60	0x3C	Stufenstellbefehl mit einem Zeitstempel CP56Time2a
C_SE_TA_1	61	0x3D	Sollwert-Stellbefehl, normierter Wert mit einem Zeitstempel CP56Time2a
C_SE_TB_1	62	0x3E	Sollwert-Stellbefehl, skalierter Wert mit einem Zeitstempel CP56Time2a

Type	Dez	Hex	Beschreibung
C_SE_TC_1	63	0x3F	Sollwert-Stellbefehl, Gleitkommazahl mit einfacher Genauigkeit und einem Zeitstempel CP56Time2a
C_BO_TA_1	64	0x40	Bitmusterbefehl von 32 bit mit einem Zeitstempel CP56Time2a
ASDU_TYPE_65..69	65..69	0x41..0x45	Reserviert (Standardbereich)
M_EI_NA_1	70	0x46	Initialisierungsende
ASDU_TYPE_71..99	71..99	0x47..0x63	Reserviert (Standardbereich)
C_IC_NA_1	100	0x64	(General-, Stations-) Abfragebefehl
C_CI_NA_1	101	0x65	Zähler-Abfragebefehl
C_RD_NA_1	102	0x66	Abfragebefehl
C_CS_NA_1	103	0x67	Uhrzeit-Synchronisierungsbefehl
C_TS_NA_1	104	0x68	Prüfbefehl
C_RP_NA_1	105	0x69	Prozess-Rücksetzbefehl
C_CD_NA_1	106	0x6A	Befehl zur Telegrammlaufzeit-Erfassung
C_TS_TA_1 [▶ 109]	107	0x6B	Prüfbefehl mit einem Zeitstempel CP56Time2a
ASDU_TYPE_108..109	108..109	0x6C..0x6D	Reserviert (Standardbereich)
P_ME_NA_1	110	0x6E	Parameter für Messwerte, normierter Wert
P_ME_NB_1	111	0x6F	Parameter für Messwerte, skalierter Wert
P_ME_NC_1	112	0x70	Parameter für messwerte, Gleitkommazahl mit einfacher Genauigkeit
P_AC_NA_1	113	0x71	Parameter für Aktivierung
ASDU_TYPE_114..119	114..119	0x72..0x77	Reserviert (Standardbereich)
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	Verzeichniss
ASDU_TYPE_127..255	127..255	0x7F..0xFF	Reserviert (benutzerdefinierter Bereich)

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 [▶ 260]	NT [▶ 260]	SB [▶ 260]	BL [▶ 260]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	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		
	|--- data[0] =	7 6 5 4 3 2 1 0 IV [▶ 260] NT [▶ 260] SB [▶ 260] BL [▶ 260] 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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV	QDS = Quality descriptor
			|---	data[2..8] =	CP56Time2a [▶ 37]								Seven octets binary time tag
			|---	data[9..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

Sehen Sie dazu auch

Weitere Informationselemente [▶ 261]

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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	QDS [▶ 261]
			|---	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	QDS [▶ 261]
			|---	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	QDS [▶ 261]
			|---	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	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 Systeminformation in Überwachungsrichtung

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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x5 (5)	= M_IRC_NA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects

|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/ type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 13									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	COL [▶ 257]	Compatibility level
			|---	data[1..8] =	ASC [▶ 257]	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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x8 (8)	= M_TGI_NA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc		= GLB									Function code/ type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	SCN [▶ 259]								Scan number

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

4.21.4 M_SYN_TA_3

Time synchronisation.

- obj					ASDU object [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x6 (6)	= M_SYN_TA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]
|	|---	asduAddr			Common address of asdu

|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc		= GLB									Function code/ type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 7									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..6] =	CP56Time2a [▶ 343]								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 [▶ 437]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0xB (11)	= M_GI_XA_3									Type identification [▶ 438]
|	|---	bSQ		= TRUE									Sequence of information objects
|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc		GEN									Function code/type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= variable									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	RII [▶ 259]								Return information identifier
			|---	data[1] =	GROUP [▶ 258]								GIN = Generic identification number
			|---	data[2] =	ENTRY [▶ 258]								
			|---	data[3] =	CONT [▶ 257]	COUNT [▶ 257]	NO [▶ 259]						NDE = Number of descriptive elements

			|---	data[4] =	KOD [▶ 258]	Kind of description	Element 1
			|---	data[5] =	DATATYPE [▶ 257]	GDD = Generic data description	
			|---	data[6] =	DATASIZE [▶ 257]		
			|---	data[7] =	CONT [▶ 257] 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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0xA (10)	= M_GD_XA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 332]

|---	- info												INFORMATION OBJECT [▶ 437]	
	|---	fc		GEN									Function code/type [▶ 266]	
	|---	n											Information number [▶ 266]	
	|---	stream											Information element/ object data [▶ 314]	
		|---	length	= variable										
		|---	data		7	6	5	4	3	2	1	0		
			|---	data[0] =	RII [▶ 259]								Return information identifier	
			|---	data[1] =	CONT [▶ 257]	COUNT [▶ 257]	NO [▶ 259]						NGD = Number of generic data sets	
			|---	data[2] =	GROUP [▶ 258]								GIN = Generic identification number	Dataset 1
			|---	data[3] =	ENTRY [▶ 258]									
			|---	data[4] =	KOD [▶ 258]								Kind of description	
			|---	data[5] =	DATATYPE [▶ 257]								GDD = Generic data description	
			|---	data[6] =	DATASIZE [▶ 257]									
			|---	data[7] =	CONT [▶ 257]	NUMBER								
			|---	data[8..8+(DATASIZE*NUMBER)] =	GID								Generic identification data	

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

4.22 Systeminformation in Steuerungsrichtung

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 [▶ 423]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x64 (100)	= C_RD_NA_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr		= 0									Record address
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x65 (101)	= C_SP_NA_2	Type identification [▶ 425]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]
	|---	rcdAddr			Record address

	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x66 (102)	= C_SP_NB_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address
	|---	stream											Information element/ object data [▶ 212]
		|---	length	= 10									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 428]								Five octets binary time tag (from)

			|---	data[5..9] =	CP40Time2a [▶ 428]	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x67 (103)	= C_TI_NA_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr		= 0									Record address
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x68 (104)	= C_CI_NA_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]

|	|---	eType	= 0x69 (105)	= C_CI_NB_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]

|	|---	eType	= 0x6A (106)	= C_CI_NC_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x6B (107)	= C_CI_ND_2	Type identification [▶ 425]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]
	|---	rcdAddr			Record address of accounting period

	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]

|	|---	eType	= 0x6C (108)	= C_CI_NE_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x6D (109)	= C_CI_NF_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= 1									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x6E (110)	= C_CI_NG_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test

|	|---	bPN											Positive/ negative confirmation/ activation
|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x6F (111)	= C_CI_NH_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 428]	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x70 (112)	= C_CI_NI_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x71 (113)	= C_CI_NK_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x72 (114)	= C_CI_NL_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]

|	|---	eType	= 0x73 (115)	= C_CI_NM_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 428]								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 [▶ 423]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x74 (116)	= C_CI_NN_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 0	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]

|	|---	eType	= 0x75 (117)	= C_CI_NO_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]

|	|---	eType	= 0x76 (118)	= C_CI_NP_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..4] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x77 (119)	= C_CI_NQ_2	Type identification [▶ 425]

|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]
	|---	rcdAddr			Record address of accounting period

	|---	stream										Information element/ object data [▶ 212]
		|---	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 [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]

|	|---	eType	= 0x78 (120)	= C_CI_NR_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]

	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 428]								Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 428]								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 [▶ 423]

|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x79 (121)	= C_CI_NS_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 428]								Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x7A (122)	= C_CI_NT_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu

|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 428]								Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x7B (123)	= C_CI_NU_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation/ activation

|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address of accounting period
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 428]	Five octets binary time tag (from)
			|---	data[7..11] =	CP40Time2a [▶ 428]	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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x6 (6)	= C_SYN_TA_3	Type identification [▶ 438]
|	|---	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 [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc		:= GLB									Function code/ type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 7									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..6] =	CP56Time2a [▶ 343]								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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x7 (7)	= C_IGI_NA_3	Type identification [▶ 438]
|	|---	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 [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc		:= GLB									Function code/ type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 1									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	SCN [▶ 259]								Scan number
			|---	data[1..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.32 C_GD_NA_3

Generic data.

- obj					ASDU object [▶ 437]

|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x0A (10)	= C_GD_NA_3									Type identification [▶ 438]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 440]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc		:= GEN									Function code/type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= variable									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	RII [▶ 259]								Return information identifier
			|---	data[1] =	CONT [▶ 257]	COUNT [▶ 257]	NO [▶ 259]						NGD = Number of generic data sets

			|---	data[2] =	GROUP [▶ 258]	GIN = Generic identification number	Dataset 1
			|---	data[3] =	ENTRY [▶ 258]		
			|---	data[4] =	KOD [▶ 258]	Kind of description	
			|---	data[5] =	DATATYPE [▶ 257]	GDD = Generic data description	
			|---	data[6] =	DATASIZE [▶ 257]		
			|---	data[7] =	CONT [▶ 257] 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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x14 (20)	= C_GRC_NA_3	Type identification [▶ 438]

|	|---	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 [▶ 332]
|---	- info				INFORMATION OBJECT [▶ 437]
	|---	fc		:= GLB	Function code/ type [▶ 266]
	|---	n			Information number [▶ 266]

	|---	stream										Information element/ object data [▶ 314]
		|---	length	= 2								
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =							DCS [▶ 336]		DCO = Double command
			|---	data[1] =	RII [▶ 259]								Return information identifier
			|---	data[2..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.22.34 C_GC_NA_3

Generic command.

- obj					ASDU object [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x15 (21)	= C_GC_NA_3	Type identification [▶ 438]
|	|---	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 [▶ 332]
|---	- info											INFORMATION OBJECT [▶ 437]
	|---	fc	:= GEN									Function code/type [▶ 266]
	|---	n										Information number [▶ 266]
	|---	stream										Information element/ object data [▶ 314]
		|---	length	= 2 + (i * 3)								
		|---	data		7	6	5	4	3	2	1	0
			|---	data[0] =				RII [▶ 259]				Return information identification
			|---	data[1] =				NOG [▶ 259]				Number of generic data sets
			|---	data[2] =				GROUP [▶ 258]				GIN = Generic identification number
			|---	data[3] =				ENTRY [▶ 258]				Dataset 1
			|---	data[4] =				KOD [▶ 258]				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 [▶ 437]
|---	+ head			Reserved
|---	- ident			DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x18 (24)	= C_ODT_NA_3	Type identification [▶ 438]
|	|---	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 [▶ 332]
|---	- info				INFORMATION OBJECT [▶ 437]

|---	fc											Function code/ type [▶ 266]
|---	n	= 0										not used
|---	stream											Information element/ object data [▶ 314]
	|---	length	= 5									
	|---	data		7	6	5	4	3	2	1	0	
		|---	data[0] =				TOO [▶ 260]					Type of order
		|---	data[1] =				TOV [▶ 260]					Type of disturbance value
		|---	data[2..3] =				FAN [▶ 258]					Fault number
		|---	data[4] =				ACC [▶ 256]					Actual channel
		|---	data[5..IEC870_MAX_ASDU_DATA_BYTE] =									

4.22.36 C_ADT_NA_3

Acknowledgement for disturbance data transmission.

- obj												ASDU object [▶ 437]
|---	+ head											Reserved

|---	- ident			DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x19 (25)	= C_ADT_NA_3	Type identification [▶ 438]
|	|---	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 [▶ 332]
|---	- info				INFORMATION OBJECT [▶ 437]
	|---	fc			Function code/ type [▶ 266]
	|---	n		= 0	not used

|---	stream											Information element/ object data [▶ 314]
	|---	length	= 5									
	|---	data		7	6	5	4	3	2	1	0	
		|---	data[0] =				TOO [▶ 260]					Type of order
		|---	data[1] =				TOV [▶ 260]					Type of disturbance value
		|---	data[2..3] =				FAN [▶ 258]					Fault number
		|---	data[4] =				ACC [▶ 256]					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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	EI [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	EI [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	EI [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	EI [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	EI [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	EI [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]				OV [▶ 261]	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 Prozessinformation in Überwachungsrichtung

4.25.1 M_SP_TA_2

Single-point information with time tag.

- obj													ASDU object [▶ 423]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x1 (1)	= M_SP_TA_2									Type identification [▶ 425]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= i									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation /activation
|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address
	|---	stream											Information element/ object data [▶ 212]
		|---	length	= i * 9									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	SPA [▶ 256]		Single-point address	Information object 1
			|---	data[1] =	SPQ [▶ 256]	SPI [▶ 212]	SPQ = Single-point qualifier SPI = Single-point information	
			|---	data[2..8] =	CP56Time2b [▶ 428]		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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x2 (2)	= M_IT_TA_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= i									Number of objects	
|	|---	bT											Test	
|	|---	bPN											Positive/ negative confirmation /activation	
|	|---	eCOT											Cause of transmission [▶ 427]	
|	|---	asdu Addr											Common address of asdu	
|	|---	eClass											Fifo priority class [▶ 212]	
|---	- info												INFORMATION OBJECT [▶ 423]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[6] =	Signature								Signature (optional)	
			|---	data[7..length - 6] =										Information object 2..i

			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x3 (3)	= M_IT_TB_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]
	|---	rcdAddr			Record address

	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[5] =	Signature								Signature (optional)	
			|---	data[6..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved

|---	- ident												DATA UNIT IDENTIFIER [▶ 424]	
|	|---	eType	= 0x4 (4)	= M_IT_TC_2								Type identification [▶ 425]	
|	|---	bSQ		= FALSE								Sequence of information objects	
|	|---	nObj		= i								Number of objects	
|	|---	bT										Test	
|	|---	bPN										Positive/ negative confirmation /activation	
|	|---	eCOT										Cause of transmission [▶ 427]	
|	|---	asduAddr										Common address of asdu	
|	|---	eClass										Fifo priority class [▶ 212]	
|---	- info											INFORMATION OBJECT [▶ 423]	
	|---	rcdAddr										Record address	
	|---	stream										Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	

			|---	data[5..length - 6] =			Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x5 (5)	= M_IT_TD_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[6] =	Signature								Signature (optional)	
			|---	data[7..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x6 (6)	= M_IT_TE_2									Type identification [▶ 425]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= i									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation /activation
|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence	Quality descriptor	
			|---	data[5] =	Signature				Signature (optional)	
			|---	data[6..length - 6] =						Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]				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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x7 (7)	= M_IT_TF_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= i									Number of objects	
|	|---	bT											Test	
|	|---	bPN											Positive/ negative confirmation /activation	
|	|---	eCOT											Cause of transmission [▶ 427]	
|	|---	asdu Addr											Common address of asdu	
|	|---	eClass											Fifo priority class [▶ 212]	
|---	- info												INFORMATION OBJECT [▶ 423]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	
			|---	data[5..length - 6] =										Information object 2..i

			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0x8 (8)	= M_IT_TG_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]
	|---	rcdAddr			Record address

	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[6] =	Signature								Signature (optional)	
			|---	data[7..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head				Reserved

|---	- ident												DATA UNIT IDENTIFIER [▶ 424]	
|	|---	eType	= 0x9 (9)	= M_IT_TH_2								Type identification [▶ 425]	
|	|---	bSQ		= FALSE								Sequence of information objects	
|	|---	nObj		= i								Number of objects	
|	|---	bT										Test	
|	|---	bPN										Positive/ negative confirmation /activation	
|	|---	eCOT										Cause of transmission [▶ 427]	
|	|---	asduAddr										Common address of asdu	
|	|---	eClass										Fifo priority class [▶ 212]	
|---	- info											INFORMATION OBJECT [▶ 423]	
	|---	rcdAddr										Record address	
	|---	stream										Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[5] =	Signature								Signature (optional)	

			|---	data[6..length - 6] =			Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0xA (10)	= M_IT_TI_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]

|---	- info												INFORMATION OBJECT [▶ 423]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	
			|---	data[5..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]								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 [▶ 423]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0xB (11)	= M_IT_TK_2									Type identification [▶ 425]
|	|---	bSQ		= FALSE									Sequence of information objects
|	|---	nObj		= i									Number of objects
|	|---	bT											Test
|	|---	bPN											Positive/ negative confirmation /activation
|	|---	eCOT											Cause of transmission [▶ 427]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 212]
|---	- info												INFORMATION OBJECT [▶ 423]
	|---	rcdAddr											Record address
	|---	stream											Information element/ object data [▶ 212]
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence	Quality descriptor	
			|---	data[6] =	Signature				Signature (optional)	
			|---	data[7..length - 6] =						Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]				Five octets binary time tag	
			|---	data[length..IEC870_MAX_ASDU_DATA_BYTE] =					Reserved	

4.25.12 M_IT_TL_2

Periodical reset operational integrated totals, 3 octets each.

- obj					ASDU object [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0xC (12)	= M_IT_TL_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects

|	|---	nObj		= i									Number of objects	
|	|---	bT											Test	
|	|---	bPN											Positive/ negative confirmation /activation	
|	|---	eCOT											Cause of transmission [▶ 427]	
|	|---	asdu Addr											Common address of asdu	
|	|---	eClass											Fifo priority class [▶ 212]	
|---	- info												INFORMATION OBJECT [▶ 423]	
	|---	rcdAddr											Record address	
	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[5] =	Signature								Signature (optional)	
			|---	data[6..length - 6] =										Information object 2..i

			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]	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 [▶ 423]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 424]
|	|---	eType	= 0xD (13)	= M_IT_TM_2	Type identification [▶ 425]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	bT			Test
|	|---	bPN			Positive/ negative confirmation /activation
|	|---	eCOT			Cause of transmission [▶ 427]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 212]
|---	- info				INFORMATION OBJECT [▶ 423]
	|---	rcdAddr			Record address

	|---	stream											Information element/ object data [▶ 212]	
		|---	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 [▶ 261]	CA [▶ 260]	CY [▶ 260]	Sequence					Quality descriptor	
			|---	data[4] =	Signature								Signature (optional)	
			|---	data[5..length - 6] =										Information object 2..i
			|---	data[length - 5..length - 1] =	CP40Time2a [▶ 428]								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 [▶ 437]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x1 (1)	= M_TTM_TA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 332]
|---	- info				INFORMATION OBJECT [▶ 437]
	|---	fc			Function code/ type [▶ 266]

	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 6									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0	0	0	0	0	0	DPI [▶ 341]		DPI = Double-point information
			|---	data[1..4] =	CP32Time2a [▶ 343]								Four octets binary time tag
			|---	data[5] =	SIN [▶ 260]								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 [▶ 437]
|---	+ head				Reserved

|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x2 (2)	= M_TMR_TA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 332]
|---	- info				INFORMATION OBJECT [▶ 437]
	|---	fc			Function code/ type [▶ 266]

	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 10									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0	0	0	0	0	0	DPI [▶ 341]		DPI = Double-point information
			|---	data[1..2] =	RET [▶ 259]								Relative time
			|---	data[3..4] =	FAN [▶ 258]								Fault number
			|---	data[5..8] =	CP32Time2a [▶ 343]								Four octets binary time tag
			|---	data[9] =	SIN [▶ 260]								Supplementary information
			|---	data[10..IEC870_MAX_ASDU_DATA_BYTE] =									Reserved

4.25.16 M_MEI_NA_3

Measurands I.

- obj					ASDU object [▶ 437]

|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x33	= M_MEI_NA_3	Type identification [▶ 438]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 332]
|---	- info				INFORMATION OBJECT [▶ 437]
	|---	fc			Function code/ type [▶ 266]

	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	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 [▶ 259]								Current L2
			|---	data[2] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[3] =	MVAL [▶ 259]								Voltage L1-L2
			|---	data[4] =						RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[5] =	MVAL [▶ 259]								Active power P

			|---	data[6] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[7] =	MVAL [▶ 259]				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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x4 (4)	= M_TME_TA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/ type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 12									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..3] =	R32 [▶ 262]								Short circuit location

			|---	data[4..5] =	RET [▶ 259]	Relative time
			|---	data[6..7] =	FAN [▶ 258]	Fault number
			|---	data[8..11] =	CP32Time2a [▶ 343]	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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x9 (9)	= M_MEII_NA_3	Type identification [▶ 438]
|	|---	bSQ		= FALSE	Sequence of information objects
|	|---	nObj		= i	Number of objects

|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/ type [▶ 266]
	|---	n											Information number [▶ 266]
	|---	stream											Information element/ object data [▶ 314]
		|---	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 [▶ 259]				Current L1
			|---	data[2] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[3] =	MVAL [▶ 259]				Current L2
			|---	data[4] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[5] =	MVAL [▶ 259]				Current L3
			|---	data[6] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[7] =	MVAL [▶ 259]				Voltage L1-E
			|---	data[8] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[9] =	MVAL [▶ 259]				Voltage L2-E
			|---	data[10] =		RES	ER	OV	MEA = Measurand with quality descriptor

			|---	data[11] =	MVAL [▶ 259]				Voltage L3-E
			|---	data[12] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[13] =	MVAL [▶ 259]				Active power P
			|---	data[14] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[15] =	MVAL [▶ 259]				Reactive power Q
			|---	data[16] =		RES	ER	OV	MEA = Measurand with quality descriptor
			|---	data[17] =	MVAL [▶ 259]				Frequency f
			|---	data[18..IEC870_MAX_ASDU_DATA_BYTE] =					Reserved

4.25.19 M_LRD_TA_3

List of recorded disturbances.

- obj					ASDU object [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]

|	|---	eType	= 0x17 (23)	= M_LRD_TA_3									Type identification [▶ 438]		
|	|---	bSQ		= FALSE									Sequence of information objects		
|	|---	nObj		= i									Number of objects		
|	|---	eCOT											Cause of transmission [▶ 439]		
|	|---	asduAddr											Common address of asdu		
|	|---	eClass											Fifo priority class [▶ 332]		
|---	- info												INFORMATION OBJECT [▶ 437]		
	|---	fc											Function code/type [▶ 266]		
	|---	n		= 0									not used		
	|---	stream											Information element/ object data [▶ 314]		
		|---	length	= i * 10											
		|---	data		7	6	5	4	3	2	1	0			
			|---	data[0..1] =	FAN [▶ 258]								Fault number	Dataset 1	
			|---	data[2] =	RES				OTEV	TEST	TM	TP	SOF = Status of fault		
			|---	data[3..9] =	CP56Time2a [▶ 343]								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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	0x1A (26)	= M_RTD_TA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 332]

|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/ type [▶ 266]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 15									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0								not used
			|---	data[1] =	TOV [▶ 260]								Type of disturbance values
			|---	data[2..3] =	FAN [▶ 258]								Fault number
			|---	data[4..5] =	NOF [▶ 259]								Number of grid faults
			|---	data[6] =	NOC [▶ 259]								Number of channels
			|---	data[7..8] =	NOE [▶ 259]								Number of information elements of a channel
			|---	data[9..10] =	INT [▶ 258]								Interval

			|---	data[11..14] =	CP32Time2a [▶ 343]	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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x1B (27)	= M_RTC_NA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]

|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/ type [▶ 266]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 17									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	0								not used
			|---	data[1] =	TOV [▶ 260]								Type of disturbance value
			|---	data[2..3] =	FAN [▶ 258]								Fault number
			|---	data[4] =	ACC [▶ 256]								Actual channel
			|---	data[5..8] =	R32 [▶ 262]								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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	0x1C (28)	= M_RTT_NA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects

|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/ type [▶ 266]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 314]
		|---	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 [▶ 258]								Fault number

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

4.25.23 M_TOT_NA_3

Transmission of tags.

- obj													ASDU object [▶ 437]
|---	+ head												Reserved
|---	- ident												DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x1D (29)	= M_TOT_NA_3									Type identification [▶ 438]
|	|---	bSQ		= TRUE									Sequence of information objects
|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/type [▶ 266]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 5 + (i * 3)									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0..1] =	FAN [▶ 258]								Fault number

			|---	data[2] =	NOT [▶ 259]		Number of tags	
			|---	data[3..4] =	TAP [▶ 260]		Tag position	
			|---	data[5] =			Function code/ type	Tag 1
			|---	data[6] =			Information number	
			|---	data[7] =	0	DPI [▶ 341]	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 [▶ 437]
|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x1E (30)	= M_TOV_NA_3	Type identification [▶ 438]

|	|---	bSQ		= TRUE									Sequence of information objects
|	|---	nObj		= 1									Number of objects
|	|---	eCOT											Cause of transmission [▶ 439]
|	|---	asduAddr											Common address of asdu
|	|---	eClass											Fifo priority class [▶ 332]
|---	- info												INFORMATION OBJECT [▶ 437]
	|---	fc											Function code/ type [▶ 266]
	|---	n		= 0									not used
	|---	stream											Information element/ object data [▶ 314]
		|---	length	= 8 + (i *2)									
		|---	data		7	6	5	4	3	2	1	0	

			|---	data[0] =	0	not used
			|---	data[1] =	TOV [▶ 260]	Type of disturbance values
			|---	data[2..3] =	FAN [▶ 258]	Fault number
			|---	data[4] =	ACC [▶ 256]	Actual channel
			|---	data[5] =	NDV [▶ 259]	Number of relevant disturbance values per ASDU
			|---	data[6..7] =	NFE [▶ 259]	Number of the ASDU's first information element
			|---	data[8..9] =	SDV [▶ 260]	Single disturbance value 1
			|---	data[10..11] =	SDV [▶ 260]	Single disturbance value 2
			|---	data[12..IEC870_MAX_ASDU_DATA_BYTE] =	SDV [▶ 260]	Single disturbance value 3..i

4.25.25 M_EOT_NA_3

End of transmission.

- obj					ASDU object [▶ 437]

|---	+ head				Reserved
|---	- ident				DATA UNIT IDENTIFIER [▶ 438]
|	|---	eType	= 0x1F (31)	= M_EOT_NA_3	Type identification [▶ 438]
|	|---	bSQ		= TRUE	Sequence of information objects
|	|---	nObj		= 1	Number of objects
|	|---	eCOT			Cause of transmission [▶ 439]
|	|---	asduAddr			Common address of asdu
|	|---	eClass			Fifo priority class [▶ 332]
|---	- info				INFORMATION OBJECT [▶ 437]
	|---	fc			Function code/ type [▶ 266]

	|---	n		= 0									not used
	|---	strea m											Infor matio n eleme nt/ object data [▶ 314]
		|---	length	= 5									
		|---	data		7	6	5	4	3	2	1	0	
			|---	data[0] =	TOO [▶ 260]								Type of order
			|---	data[1] =	TOV [▶ 260]								Type of distur bance values
			|---	data[2 ..3] =	FAN [▶ 258]								Fault numb er
			|---	data[4] =	ACC [▶ 256]								Actual chann el
			|---	data[5 ..IEC8 70_M AX_A SDU_ DATA _BYT E] =									Reser ved

4.26 Informationselemente

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 Weitere Informationselemente

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> = Ausführen (execute);
- <1> = Anwählen (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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	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 [▶ 261]	NT [▶ 261]	SB [▶ 261]	BL [▶ 260]	0	0	0	OV [▶ 261]	QDS = Quality descriptor

			|---	data[2..8] =	CP56Time2a	Seven octets binary time tag
			|---	data[9..IEC870_MAX_ASDU_DATA_BYTE] =		Reserved

4.28 Funktionstyp (code)

Typ	Beschreibung
0..127	Reserviert (privater Bereich)
128	Distanzschutz
129	Nicht benutzt (kompatibler Bereich)
130..143	Reserviert (privater Bereich)
144..145	Nicht benutzt (kompatibler Bereich)
146..159	Reserviert (privater Bereich)
160	Überstromzeitschutz
161	Nicht benutzt (kompatibler Bereich)
162..175	Reserviert (privater Bereich)
176	Transformator-Differentialschutz
177	Nicht benutzt (kompatibler Bereich)
178..191	Reserviert (privater Bereich)
192	Leitungsdifferentialschutz
193	Nicht benutzt (kompatibler Bereich)
194..207	Reserviert (privater Bereich)
208	Nicht benutzt (kompatibler Bereich)
209	Nicht benutzt (kompatibler Bereich)
210..223	Reserviert (privater Bereich)
224	Nicht benutzt (kompatibler Bereich)
225	Nicht benutzt (kompatibler Bereich)
226..239	Reserviert (privater Bereich)
240	Nicht benutzt (kompatibler Bereich)
241	Nicht benutzt (kompatibler Bereich)
242..253	Reserviert (privater Bereich)
254	Generische Funktion (GEN)
255	Globale Funktion (GLB)

4.29 Informationsnummer

Nummer	Überwachungsrichtung	Steuerungsrichtung
0..15	Systemfunktionen	Systemfunktionen
16..31	Zustand	Allgemeine Befehle
32..47	Überwachung	Nicht benutzt
48..63	Erdschlüsse	Nicht benutzt
64..127	Kurzschlüsse	Nicht benutzt

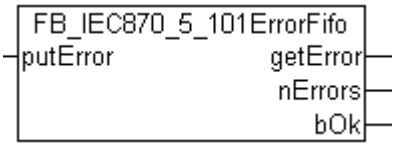
Nummer	Überwachungsrichtung	Steuerungsrichtung
128..143	Automatische Wiedereinschaltung	Nicht benutzt
144..159	Betriebsmesswerte	Nicht benutzt
160..239	Nicht benutzt	Nicht benutzt
240..255	Generische Funktionen	Generische Funktionen

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

Die TwinCAT SPS-Bibliothek **TcIEC870_5_101.Lib** implementiert die Verbindungsfunktionen und Datenstrukturen nach der IEC60870-5-101 Norm.

5.1 Funktionsbausteine

5.1.1 FB_IEC870_5_101ErrorFifo



IEC60870-5-10x Fehler-Fifo. Der älteste Eintrag wird immer überschrieben. Der Fifo hat eine konstante Größe. Die Größe wird durch die Konstante: *IEC870_MAX_ERROR_FIFO_SIZE* bestimmt (default: 10 Elemente).

Der Funktionsbaustein besitzt drei Aktionen:

- **AddError** (fügt eine neue Fehlermeldung dem Fifo hinzu);
- **RemoveError** (entfernt die älteste Fehlermeldung aus dem Fifo);
- **Reset** (löscht alle Fehlermeldungen, setzt den Fifo zurück);

Im Normalfall werden die Fehlermeldungen durch die internen IEC60870-5-10x Gerätefunktionen dem Fifo hinzugefügt. Die SPS-Applikation kann diese Fehlermeldungen durch den Aufruf der Aktion: **RemoveError** auslesen und auswerten.

VAR_INPUT

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

putError: Fehlermeldung, die dem Fifo [▶ 319] hinzugefügt werden soll.

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: Fehlermeldung, die aus dem Fifo [▶ 319] entfernt wurde.

nErrors: Liefert die aktuelle Anzahl Fifo-Einträge (Fehlermeldungen im Fifo).

bOk: Diese Variable wird TRUE, wenn ein neuer Eintrag erfolgreich hinzugefügt oder aus dem Fifo entfernt wurde.

Beispiel in ST:

Im folgenden ST-Beispiel wird der Gerätefehler-Fifo ausgelesen und die registrierten Fehler ins Windows Application Log geschrieben.

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

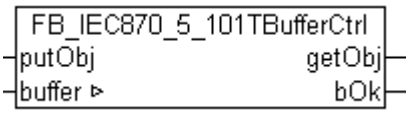
```
REPEAT
    fbErrors.RemoveError();
    IF fbErrors.bOk THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'IEC60870-5-10x device error: 0x%s',
            DWORD_TO_HEXSTR( fbErrors.getError.nErrId, 8, FALSE) );
    END_IF
UNTIL NOT fbErrors.bOk
END_REPEAT
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.1.2 FB_IEC870_5_101TBufferCtrl

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 / IEC60870-5-101 Unterstation v2.0.2 und höher.



Mit diesem Funktionsbaustein kann der Inhalt des TX/RX-Datenpuffers manipuliert werden, der bei der Kommunikation über das IEC60870-5-104/101 Transport Interface benutzt wird.

Der Funktionsbaustein besitzt folgende Aktionen:

- **RxRemoveObj** (entfernt den ältesten Fifoeintrag aus dem RX-Fifo);
- **RxReset** (löscht alle RX-Fifoeinträge, setzt den RX-Fifo zurück);
- **TxAddObj** (fügt einen neuen Fifoeintrag in den TX-Fifo);
- **TxReset** (löscht alle TX-Fifoeinträge, setzt den TX-Fifo zurück)

Durch den Aufruf der oben aufgelisteten Aktionen kann der Inhalt des TX/RX-Datenpuffers verändert werden.

VAR_IN_OUT

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

buffer: TX/RX-Datenpuffer [▶ 310]. Die TX/RX-Pufferparameter (wie z.B. asduSize) müssen vor der Benutzung konfiguriert werden.

VAR_INPUT

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

putObj: Dateneinheit [▶ 312] (ASDU), die gesendet werden soll.

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: Empfangene Dateneinheit [▶ 312] (ASDU).

bOk: Diese Variable wird TRUE, wenn ein neuer Eintrag erfolgreich hinzugefügt oder aus dem Fifo entfernt wurde. Diese Variable wird FALSE beim Pufferüberlauf und wenn kein Eintrag entfernt werden konnte, weil der Fifo bereits leer ist.

Beispiel in ST:

Folgendes Beispielcode (Ausschnitt) demonstriert die Benutzung der Baustein-Aktionen. Alle ~100ms (tCycle) wird eine neue ASDU (M_BO_TB_1) mit der Übertragungsursache: Spontan und Zeitstempel generiert und in den TX-Fifo abgelegt.
Die empfangenen Testkommandos (C_TS_TA_1) und Uhrzeitsynchronisationskommandos (C_CS_NA_1) werden aus dem RX-Fifo entfernt und mit gespiegelten ASDU's beantwortet.

```
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
```

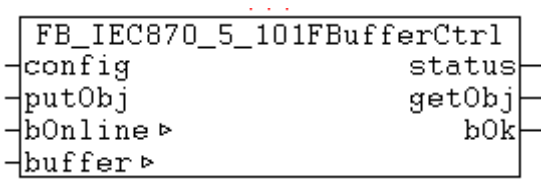
```
END_IF



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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

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.



Mit diesem Funktionsbaustein kann der Inhalt des TX/RX-Datenpuffers manipuliert werden, der bei der Kommunikation über das IEC60870-5-104/101 (Low Level) Transport Interface benutzt wird. Zusätzlich werden die zu verschickenden ASDUs (nur TX-Richtung) in die Datei gepuffert wenn die Verbindung zur Zentralstation unterbrochen wurde (im Offline-Mode). Die Funktionalität ähnelt der Funktionalität des FB_IEC870_5_101TBufferCtrl-Funktionsbausteins.

Der Funktionsbaustein besitzt folgende Aktionen:

- **RxRemoveObj** (entfernt den ältesten Fifoeintrag aus dem RX-Fifo);
- **RxReset** (löscht alle RX-Fifoeinträge, setzt den RX-Fifo zurück);
- **TxAddObj** (fügt einen neuen Fifoeintrag in den TX-Fifo);
- **TxReset** (löscht alle TX-Fifoeinträge, setzt den TX-Fifo zurück)

Durch den Aufruf der oben aufgelisteten Aktionen kann der Inhalt des TX/RX-Datenpuffers verändert werden.

VAR_IN_OUT

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

bOnline: Über diesen Eingang wird dem Funktionsbaustein mitgeteilt, ob die Verbindung sich im Offline oder Online-Mode befindet. TRUE = Online, FALSE = Offline. Im Offline-Mode werden die zu versendenden ASDUs in einer Datei gepuffert. Im Online-Mode werden dann die in der Datei gepufferten ASDUs aus der Datei entfernt und an die Zentralstation verschickt.

buffer: TX/RX-Datenpuffer [▶ 310]. Die TX/RX-Pufferparameter (wie z.B. asduSize) müssen vor der Benutzung konfiguriert werden.

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-Dateipuffer-Konfigurationseinstellungen [▶ 324].

putObj: Dateneinheit [▶ 312] (ASDU), die gesendet werden soll.

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-Datenpuffer-Statusinformationen [▶ 325].

getObj: Empfangene Dateneinheit [▶ 312] (ASDU).

bOk: Diese Variable wird TRUE, wenn ein neuer Eintrag erfolgreich hinzugefügt oder aus dem Fifo entfernt wurde. Diese Variable wird FALSE beim Pufferüberlauf und wenn kein Eintrag entfernt werden konnte, weil der Fifo bereits leer ist.

Beispiel in ST:

Folgender Beispielcode (Ausschnitt) demonstriert die Benutzung der Baustein-Aktionen. Alle ~1s (tCycle) wird eine neue ASDU (M_BO_TB_1) mit der Übertragungsursache: Spontan und Zeitstempel generiert und in den TX-Fifo abgelegt.
Die empfangenen Testkommandos (C_TS_TA_1) und Uhrzeitsynchronisationskommandos (C_CS_NA_1) werden aus dem RX-Fifo entfernt und mit gespiegelten ASDUs beantwortet.

Über die VAR_IN_OUT-Variable *bOnline* wird das Speichern oder Laden der ASDUs in die Datei gesteuert. Bei *bOnline* = FALSE wird die Datei geöffnet und die anfallenden TX-ASDUs im Hintergrund in die Datei geschrieben. Bei bOnline = TRUE werden die ASDUs aus der Datei im Hintergrund geladen und verschickt. Damit das Speichern und Laden der Pufferdatei im Hintergrund durchgeführt werden kann muss der

FB_IEC870_5_101FBufferCtrl-Funktionsbaustein zyklisch aufgerufen werden. Das Hinzufügen neuer TX-ASDUs oder das Bearbeiten alter RX-ASDUs wird von dem Speichern/Laden in/aus der Datei nicht beeinflusst.

```
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
```

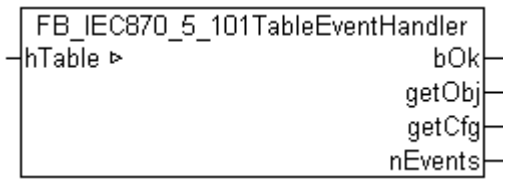
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1313	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.1.4 FB_IEC870_5_101TableEventHandler

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.

Das Auslesen der Ereignisse ist optional und nicht zwingend notwendig. D.h. die SPS-Applikation muss diesen Baustein nicht unbedingt instanziieren. Zurzeit wird diese Funktionalität nur vom IEC 60870-5-104 Master unterstützt!



Mit diesem Funktionsbaustein kann die SPS-Applikation einige Änderungen in der IEC-Applikationsobjektdatenbank erkennen und, wenn nötig, darauf entsprechend reagieren. Die Änderungen werden als Ereignisse bezeichnet. Jeder Ereignis-Typ wird in einer separaten internen Liste verwaltet. Die SPS-Applikation kann durch den Aufruf der Bausteinaktionen die anstehenden Ereignisse aus einer der Listen auslesen. Da auch mehrere Ereignisse pro SPS-Zyklus auftreten können werden die Ereignisse intern gezählt. Der Zähler wird beim Auftreten eines Ereignisses immer inkrementiert. An dem Bausteinausgang wird nur der letzte Wert und der Zählerstand ausgegeben.

Folgende Ereignisse werden von dem Funktionsbaustein registriert:

- **OnCreate-Ereignisse** werden immer dann gemeldet, wenn ein neues Applikationsobjekt (Single Point, Double Point, Measured Value...) der Applikationsdatenbank hinzugefügt wurde.
- **OnChange-Ereignisse** werden dann gemeldet, wenn ein Applikationsobjekt von der unteren Transportschicht empfangen wurde (Rx-Frames) oder verschickt wird (Tx-Frames), unabhängig davon ob sich der Wert des Informationsobjektes tatsächlich geändert hat oder nicht. Bei einem direkten Kommando z.B. C_SC_NA_1 in Steuerungsrichtung (Master->Slave) werden im Normalfall beifolgenden Übertragungsursachen Ereignisse gemeldet: eIEC870_COT_ACT (Aktivierung), eIEC870_COT_ACT_CON (Bestätigung der Aktivierung) und bei eIEC870_COT_TERM (Beendigung der Aktivierung). Bei einem Datenpunkt in Überwachungsrichtung (Slave->Master) z.B. M_SP_NA_1 können beifolgenden Übertragungsursachen Ereignisse gemeldet werden: eIEC870_COT_SPONTAN, eIEC870_COT_INROGEN, eIEC870_COT_BACKGROUND usw.

Der Funktionsbaustein besitzt zwei Aktionen:

- **RemoveOnCreateEvent** (Liest einen Eintrag aus der OnCreate-Ereignisliste aus);
- **RemoveOnChangeEvent** (Liest einen Eintrag aus der OnChange-Ereignisliste aus);

VAR_IN_OUT

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

hTable: Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

VAR_OUTPUT

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

bOk: Diese Variable wird TRUE, wenn ein Ereignis erfolgreich ausgelesen wurde. Bei FALSE ist die zuletzt gelesene Ereignisliste leer.

getObj: Der aktuelle Wert der Dateneinheit [▶ 312] (ASDU).

getCfg: Die aktuellen Konfigurationsparameter [▶ 313] der Dateneinheit (ASDU).

nEvents: Ereignis-Zähler (Multiplikator). Wertebereich: (0 bis 16#FFFFFFFF). Beim Erreichen des Maximalwertes wird nicht mehr inkrementiert.

Beispiel in ST:

Im folgenden Programmausschnitt werden die anstehenden Ereignisse in REPEAT-Schleifen ausgelesen und in Windows Application Log geschrieben. Die gesuchten Datenpunkte sind bereits als Hash-Tabelleneinträge konfiguriert worden. Siehe in der Beschreibung der Funktion: F_iecAddTableEntry [▶ 294].

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

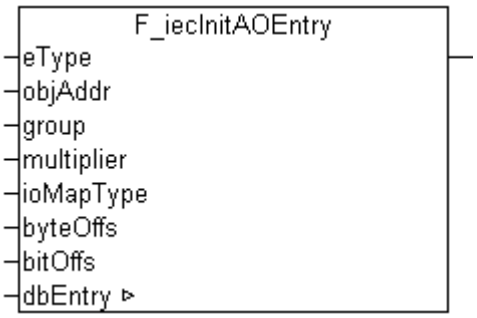
```
REPEAT
    fbHandler.RemoveOnChangeEvent( hTable := hTable );
    IF fbHandler.bOk THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_LOG,
            'RemoveOnChangeEvent(), IOA: %s',
            DWORD_TO_STRING( fbHandler.getObj.info.objAddr ) );
    END_IF
UNTIL NOT fbHandler.bOk
END_REPEAT


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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2 Funktionen

5.2.1 F_iecInitAOEntry



Die Funktion F_iecInitAOEntry konfiguriert die Applikationsobjekte (Single Points, Double Points, Measured Values...) in der Applikationsdatenbank als Linear-Tabelleneinträge. Das zu konfigurierende Tabellenelement (Arrayelement) muss als VAR_IN_OUT-Funktionsparameter an die Funktion übergeben werden.

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: Applikationsobjekt Typ [▶ 326], ASDU identifier (z.B.: M_SP_NA_1 für Single-Point oder M_DP_NA_1 für Double-Point).

objAddr : Objektadresse, frei wählbar.

group: Object-Group-Konfigurationsflags. Hier finden sie die Beschreibung aller Group-Flags [▶ 347]. Die Flags können mit OR-Verknüpfung kombiniert werden. Nicht alle Kombinationen sind aber sinnvoll!

multiplier: Basis-Zykluszeit-Multiplikator für zyklische/periodische Datenübertragung. 0 = Deaktiviert. Die Basis-Zykluszeit kann über den *tPerCyclicBase*-Parameter in den Systemparametern [▶ 315] konfiguriert werden.

ioMapType: TwinCAT SPS-Prozessdatenbereich. Dieser Parameter [▶ 329] legt fest wie die TwinCAT SPS und IEC-Applikationsobjekt Prozessdaten gemappt werden sollen.

byteOffs: TwinCAT SPS-Prozessdaten-Byte-Offset.

bitOffs: TwinCAT SPS-Prozessdaten-Bit-Offset.

dbEntry: Das zu konfigurierende Tabellenelement [▶ 312] (Arrayelement).

Rückgabeparameter	Beschreibung
0	Kein Fehler.
<> 0	Fehler: IEC60870-5-10x Fehlercode

Beispiel in ST:

Im folgenden Beispiel werden drei Datenpunkte als Linear-Tabelleneinträge konfiguriert.

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
```

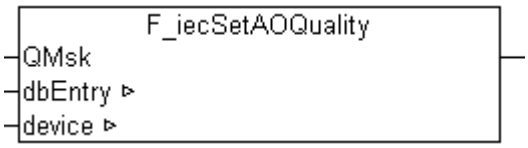
```
IF init THEN
    init := FALSE;
    initError := F_iecInitAOEntry(  eType       := M_SP_NA_1,
                    objAddr     := 100,
                    group       := IEC870_GRP_INRO1,
                    multiplier  := 0,
                    ioMapType   := MAP_AREA_MEMORY,
                    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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.2 F_iecSetAOQuality



Mit dieser Funktion können Quality-Flags eines Applikationsobjekts auf einen bestimmten Wert gesetzt/ zurückgesetzt werden.

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. Die Quality-Flags können mit OR-Verknüpfung kombiniert werden. Bei QMsk = Null werden keine Flags gesetzt/zurückgesetzt. Eine Liste der verfügbaren Quality-Flags finden Sie hier: Quality-Flags [▶ 348].

dbEntry: Applikationsobjekt [▶ 312] dessen Status der Quality-Flags gesetzt werden soll.

device: Kommunikationsschnittstelle [▶ 318] des IEC-Geräts.

Rückgabeparameter	Bedeutung
0	Kein Fehler.
<> 0	Fehler: IEC60870-5-10x Fehlercode

Beispiel für einen Aufruf 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
```

Programmcode:

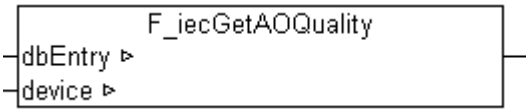
```
IF bBlock THEN
    bBlock := FALSE;
    F_iecSetAOQuality( IECQ_BL_ON, slave1AODB[1], server1.system.device );
```

```
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 );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.3 F_iecGetAOQuality



Mit der Funktion kann der Status der Quality-Flags eines Applikationsobjekts gelesen werden.

FUNCTION F_iecGetAOQuality: DWORD

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

dbEntry : Applikationsobjekt [▶ 312] dessen Quality-Flags gelesen werden sollen.

device: Kommunikationsschnittstelle [▶ 318] zum IEC-Gerät.

Rückgabeparameter	Bedeutung
0	Fehler, keine Quality-Flags für dieses Applikationsobjekt verfügbar.
<> 0	Kein Fehler. Der Rückgabeparameter liefert den Status der Quality-Flags. Eine Liste der verfügbaren Quality-Flags finden Sie hier: Quality-Flags [▶ 348].

Beispiel für einen Aufruf 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
```

Programmcode:

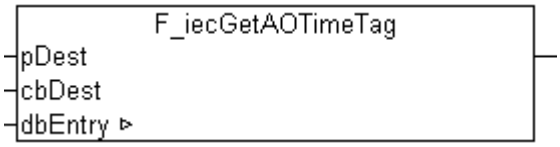
```
IF bBlock THEN
    bBlock := FALSE;
    F_iecSetAOQuality( IECQ_BL_ON, slave1AODB[1], server1.system.device );
END_IF
```

```
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 );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.4 F_iecGetAOTimeTag



Mit dieser Funktion kann der aktuelle Zeitstempel eines Applikationsobjekts in einen Bytepuffer gelesen werden.

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 : Pufferadresse.

cbDest: Bytegröße des Puffers.

dbEntry: Applikationsobjekt [▶ 312] dessen Zeitstempel gelesen werden soll.

Rückgabeparameter	Bedeutung
0	Fehler, das Applikationsobjekt besitzt keinen Zeitstempel.
<> 0	Anzahl der erfolgreich kopierten Zeitstempeldatenbytes. Bei einem CP24Time2a-Zeitstempelformat sind es z.B. 3 Bytes und bei einem CP56Time2a-Zeitstempelformat sind es 7 Bytes.

Beispiel für einen Aufruf in ST:

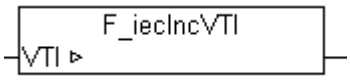
```
PROGRAM MAIN
VAR
(*...*)
    TT1     : T_CP56Time2a;
    TTSize      : UDINT;
    bGetTT      : BOOL;
END_VAR
```

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.5 F_iecIncVTI



Diese Funktion inkrementiert den INT7 regelnden Schrittwert. Das transiente Bit wird nicht verändert.

FUNCTION F_iecIncVTI: BOOL

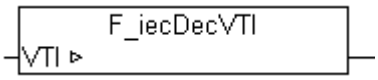
```
VAR_IN_OUT
    VTI : BYTE;
END_VAR
```

VTI: Das zu inkrementierende Byte.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.6 F_iecDecVTI



Diese Funktion dekrementiert den INT7 regelnden Schrittwert. Das transiente Bit wird nicht verändert.

FUNCTION F_iecDecVTI: BOOL

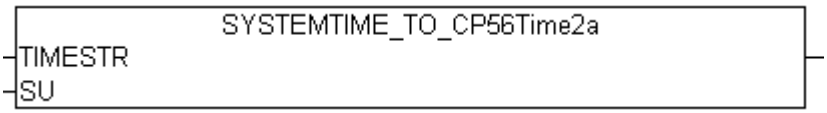
```
VAR_IN_OUT
    VTI     : BYTE;
END_VAR
```

VTI: Das zu dekrementierende Byte.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.7 SYSTEMTIME_TO_CP56Time2a



Die Funktion konvertiert das Windows-Systemtime Format in das CP56Time2a-Format. Alle reservierten Bits sind Null.

FUNCTION SYSTEMTIME_TO_CP56Time2a: T_CP56Time2a

T_CP56Time2a [▶ 343]

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

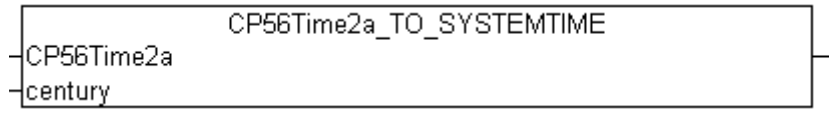
TIMESTR : Die zu konvertierende Systemzeit.

SU: Sommer-/Winterzeitformat. Diese Information ist in dem TIMESTR-Format nicht vorhanden und muss zusätzlich angegeben werden. TRUE = Sommerzeit, FALSE = Winterzeit.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.2.8 CP56Time2a_TO_SYSTEMTIME



Die Funktion konvertiert das CP56Time2a-Zeitformat in Windows-Systemzeit-Format. Das SU-Flag wird nicht benutzt.

FUNCTION CP56Time2a_TO_SYSTEMTIME: TIMESTRUCT

TIMESTRUCT

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

CP56Time2a : Die zu konvertierende Zeit im CP56Time2a-Format [▶ 343].

century: Das Jahrhundert (z.B. 20 für das Jahr 2005). Diese Information ist im CP56Time2a-Format nicht enthalten und aus diesem Grund muss sie zusätzlich angegeben werden.

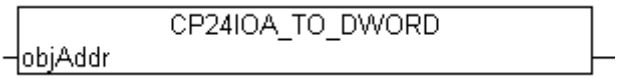
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.9 CP24IOA_TO_DWORD

Ab der Produktversion: TwinCAT PLC Library: IEC60870-5-104 Unterstation v3.0.0 / IEC60870-5-104 Zentralstation v1.0.0 und höher.



Die Funktion generiert eine strukturierte TwinCAT-Objektadresse (3 oktete). Siehe auch: DWORD_TO_CP24IOA [▶ 284].

FUNCTION CP24IOA_TO_DWORD: DWORD

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

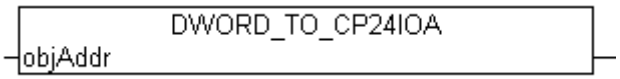
objAddr : Parameter [▶ 344] der strukturierten TwinCAT-Objektadresse.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.10 DWORD_TO_CP24IOA

Ab der Produktversion: TwinCAT PLC Library: IEC60870-5-104 Unterstation v3.0.0 / IEC60870-5-104 Zentralstation v1.0.0 und höher.



Die Funktion konvertiert eine strukturierte TwinCAT-Objektadresse in einzelne Adressparameter. Siehe auch: CP24IOA_TO_DWORD [▶ 284].

FUNCTION DWORD_TO_CP24IOA: T_CP24IOA

T_CP24IOA [▶ 344]

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

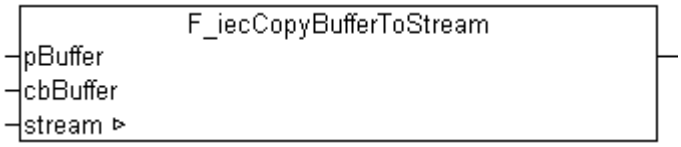
objAddr : Strukturierte TwinCAT-Objektadresse (3 oktete).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.11 F_iecCopyBufferToStream

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 und höher.



Diese Funktion kopiert Datenbytes von einer externen Puffervariablen in die *stream*-Variable. Der Speicherinhalt der *stream*-Variablen wird vergrößert. Der Rückgabeparameter der Funktion liefert die Anzahl der erfolgreich kopierten Datenbytes.

FUNCTION F_iecCopyBufferToStream: UDINT

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

pBuffer: Pointer (Adresse) einer externen Puffervariablen.

cbBuffer: Anzahl der Datenbytes die von der externen Puffervariablen in die *stream*-Variable kopiert werden sollen.

stream: Datenpuffer [▶ 314].

Beispiel in ST:

Bei einer steigenden Flanke an der *bTx* werden 4 Datenbytes der *txBuffer*-Variablen in die *stream*-Variable hineinkopiert.

```
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
```

Speicherdarstellung der *stream*-Variablen vor dem ersten Funktionsaufruf:

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

Speicherdarstellung der *stream*-Variablen nach dem ersten Funktionsaufruf:

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

Speicherdarstellung der *stream*-Variablen nach dem zweiten Funktionsaufruf:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.12 F_iecCopyStreamToBuffer

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 und höher.

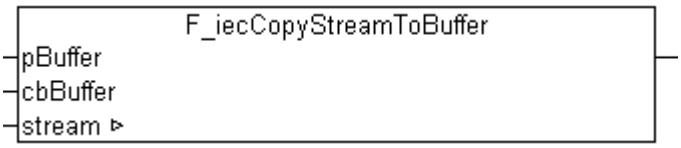


Abb. 1: F_iecCopyStreamToBuffer

Diese Funktion kopiert Datenbytes von der *stream*-Variablen in eine externe Puffervariable. Der Speicherinhalt der *stream*-Variablen bleibt unverändert. Der Rückgabeparameter der Funktion liefert die Anzahl der erfolgreich kopierten Datenbytes.

FUNCTION F_iecCopyStreamToBuffer: UDINT

```
VAR_INPUT
    pBuffer     : DWORD;
    cbBuffer    : UDINT;
```

```
END_VAR
VAR_IN_OUT
    stream  : ST_IEC870_5_101Stream;
END_VAR
```

pBuffer: Pointer (Adresse) einer externen Puffervariablen.

cbBuffer: Maximale Anzahl der Datenbytes die aus der *stream*-Variablen in den externen Puffer kopiert werden sollen.

stream: Datenpuffer [▶ 314].

Beispiel in ST:

Nach dem Programmstart werden 4 Datenbytes in die *stream*-Variable kopiert. Bei einer steigenden Flanke am bRx werden die ersten vier Datenbytes der *stream*-Variablen in die rxBuffer-Variable kopiert.

```
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
```

Speicherdarstellung der *stream*-Variablen nach dem Programmstart:

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

Speicherdarstellung der *stream*-Variablen nach dem ersten und jedem weiteren F_CopyStreamToBuffer-Funktionsaufruf:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.13 F_iecCopyStreamToStream

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 und höher.

Abb. 2: F_iecCopyStreamToStream

Diese Funktion kopiert Datenbytes von der *source*-Variablen in die *target*-Variable. Der Speicherinhalt der *source*-Variablen bleibt unverändert. Der Speicherinhalt der *target*-Variablen wird vergrößert. Der Rückgabeparameter der Funktion liefert die Anzahl der erfolgreich kopierten Datenbytes.

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: Anzahl der Bytes die von der *source*-Variablen in die *target*-Variable kopiert werden sollen.

target: Ziel-Datenpuffer [▶ 314].

source: Quell-Datenpuffer.

Beispiel in ST:

Bei einer steigenden Flanke am *bCopy* wird zuerst *srcValue* inkrementiert und in *srcStream* kopiert. Danach werden die ersten 4 Datenbytes von *srcStream* nach *dstStream* kopiert. Zum Schluss werden die ersten 4 Datenbytes von *dstStream* in die *dstValue*-Variable kopiert.

```
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
```

Speicherdarstellung der *srcStream*- und *dstStream*-Variablen nach dem ersten F_iecCopyStreamToStream-Funktionsaufruf:

Tab. 1: srcStream:

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

Tab. 2: dstStream:

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

Speicherdarstellung der *srcStream*- und *dstStream*-Variablen nach dem zweiten F_iecCopyStreamToStream-Funktionsbaufruf:

Tab. 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

Tab. 4: dstStream:

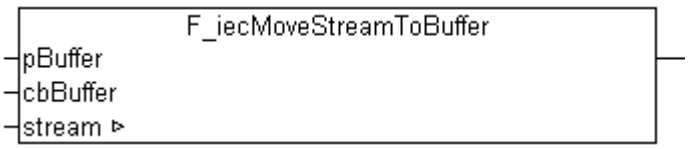
length	data											
8	16#01	16#00	16#00	16#00	16#01	16#00	16#00	16#00	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.14 F_iecMoveStreamToBuffer

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 und höher.



Diese Funktion kopiert Datenbytes von der *stream*-Variablen in eine externe Puffervariable und löscht anschließend die kopierten Datenbytes aus der *stream*-Variablen. Der Speicherinhalt der *stream*-Variablen wird verkleinert. Der Rückgabeparameter der Funktion liefert die Anzahl der erfolgreich kopierten Datenbytes.

FUNCTION F_iecMoveStreamToBuffer : UDINT

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

pBuffer: Pointer (Adresse) einer externen Puffervariablen.

cbBuffer: Maximale Anzahl der Datenbytes die aus der *stream*-Variablen in den externen Puffer kopiert werden sollen.

stream: Datenpuffer [▶ 314].

Beispiel in ST:

Nach dem Programmstart werden zwei DWORD-Werte in die *stream*-Variable kopiert. Bei einer steigenden Flanke an *bRx* werden jedes Mal 4 Datenbytes aus der *stream*-Variablen in die *rxBuffer*-Variable kopiert.

```
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
```

Speicherdarstellung der stream-Variablen nach dem Programmstart:

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

Speicherdarstellung der stream-Variablen nach dem ersten F_iecMoveStreamToBuffer-Funktionsaufruf:

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

Speicherdarstellung der stream-Variablen nach dem zweiten F_iecMoveStreamToBuffer-Funktionsaufruf:

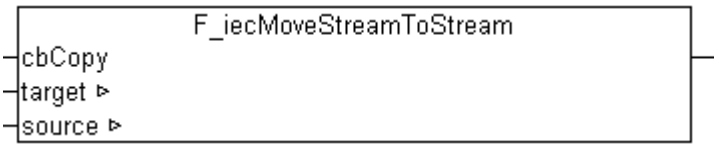
length	data											
0	16#01	16#EF	16#CD	16#AB	16#01	16#EF	16#CD	16#AB	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.15 F_iecMoveStreamToStream

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 und höher.



Diese Funktion kopiert Datenbytes von der *source*-Variablen zur *target*-Variablen und löscht anschließend die kopieren Datenbytes aus der *source*-Variablen. Der Speicherinhalt der *source*-Variablen wird verkleinert und der, der *target*-Variablen vergrößert.

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: Anzahl der Datenbytes die von der *source*-Variablen in die *target*-Variablen kopiert werden sollen.

target: Ziel-Datenpuffer [▶ 314].

source: Quell-Datenpuffer.

Beispiel in ST:

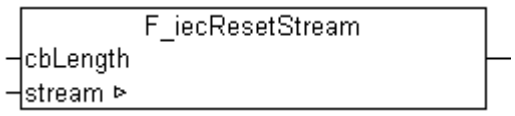
In Vorbereitung.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.16 F_iecResetStream

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 und höher.



Die Funktion initialisiert und setzt die *stream*-Variable zurück. Optional kann der interne Datenpuffer der *stream*-Variablen mit einer bestimmten Anzahl an Null-Bytes initialisiert werden. Der Rückgabewert der Funktion liefert die Anzahl der erfolgreich initialisierten Null-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: Anzahl der zu initialisierten Null-Bytes.

stream: Variable [▶ 314], die initialisiert werden soll.

Beispiel in ST:

Nach dem Programmstart wird der interne Puffer der *stream*-Variablen zurückgesetzt und mit 5 Null-Bytes initialisiert.

```
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
```

Speicherinhalt der stream-Variablen nach dem Programmstart:

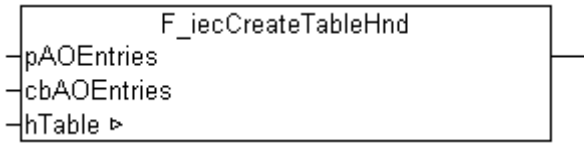
length	data											
5	16#00	16#00	16#00	16#00	16#00	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.17 F_iecCreateTableHnd

Ab Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.



Die Funktion F_iecCreateTableHnd initialisiert das Applikationsobjekt-Datenbankhandle (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig initialisiert werden.

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: Adresse der Applikationsobjekt-Datenbankvariablen.

cbAOEntries: Bytegröße der Applikationsobjekt-Datenbankvariablen.

hTable: Das zu initialisierende Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle).

Rückgabeparameter	Beschreibung
0	Kein Fehler.
<> 0	Fehler: IEC60870-5-10x Fehlercode

Beispiel in ST:

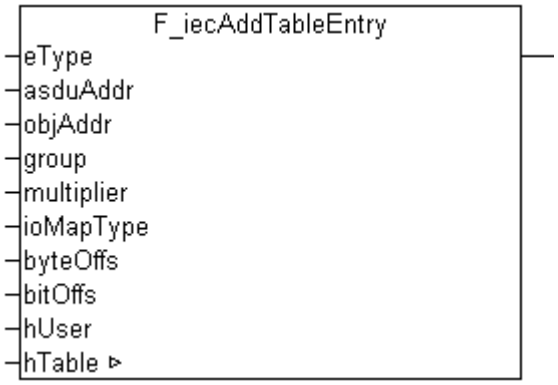
Siehe in der Beschreibung der F_iecAddTableEntry [▶ 294]-Funktion.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.18 F_iecAddTableEntry

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.



Die Funktion F_iecAddTableEntry konfiguriert die Applikationsobjekte (Single Points, Double Points, Measured Values...) in der Applikationsdatenbank als Hash-Tabelleneinträge. Die Funktion sucht automatisch nach einem freien, noch nicht belegten Tabellenelement (Arrayelement) und setzt dessen Konfigurationsparameter.

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: Applikationsobjekt Typ, ASDU identifier [▶ 326] (z.B.: M_SP_NA_1 für Single-Point oder M_DP_NA_1 für Double-Point).

asduAddr: Gemeinsame ASDU-Adresse.

objAddr: Objektadresse, frei wählbar.

group: Object-Group-Konfigurationsflags. Hier finden sie die Beschreibung aller Group-Flags [▶ 347]. Die Flags können mit OR-Verknüpfung kombiniert werden. Nicht alle Kombinationen sind aber sinnvoll!

multiplier: Basis-Zykluszeit-Multiplikator für zyklische/periodische Datenübertragung. 0 = Deaktiviert. Die Basis-Zykluszeit kann über den *tPerCyclicBase*-Parameter in den Systemparametern [▶ 315] konfiguriert werden.

ioMapType: TwinCAT SPS-Prozessdatenbereich. Dieser Parameter [▶ 329] legt fest wie die TwinCAT SPS und IEC-Applikationsobjekt Prozessdaten gemappt werden sollen.

byteOffs: TwinCAT SPS-Prozessdaten-Byte-Offset.

bitOffs: TwinCAT SPS-Prozessdaten-Bit-Offset.

hUser: Frei definierbarer 32-Bit-Wert. Dieser Wert wird in den Konfigurationsdaten des Applikationsobjekts abgelegt.

hTable: Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

Rückgabeparameter	Beschreibung
0	Kein Fehler.
<> 0	Fehler: IEC60870-5-10x Fehlercode

Beispiel in ST:

Im folgenden Beispiel werden drei Datenpunkte in die Applikationsdatenbank als Hash-Tabelleneinträge hinzugefügt:

eType	asduAddr	objAddr	group	multiplier	ioMapTy-pe	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 );
```

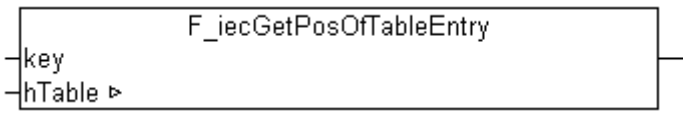
```
    IF initError <> 0 THEN
        ADSLOGSTR( ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_LOG,
            'F_iecAddTableEntry() error: %s',
            DWORD_TO_HEXSTR( initError, 8, FALSE ) );
        RETURN;
    END_IF

    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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.19 F_iecGetPosOfTableEntry

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.



Die Funktion F_iecGetPosOfTableEntry liefert den zum Lookup-Schlüssel passenden Tabellenindex (Arrayindex) eines Hash-Tabelleneintrags. Das erste Arrayelement hat die Positionsnummer eins (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-Schlüssel [▶ 323].

hTable: Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

Rückgabeparameter	Beschreibung
0	Es wurde kein zum Schlüssel passender Tabelleneintrag gefunden.
<> 0	Kein Fehler. Der Rückgabeparameter liefert den gesuchten Tabellenindex (non-zero-based array position).

Beispiel in ST:
Es wird nach dem Linearen-Tabellenindex von drei Datenpunkten gesucht. Die gesuchten Datenpunkte sind bereits als Hash-Tabelleneinträge konfiguriert worden. Siehe in der Beschreibung der Funktion: F_iecAddTableEntry [▶ 294].

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
```

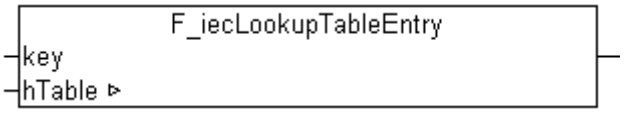
```
IF bGetPos      THEN
    bGetPos         := FALSE;

    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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.20 F_iecLookupTableEntry

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.



Die Funktion F_iecLookupTableEntry prüft ob ein zum Schlüssel passender Tabelleneintrag bereits vorhanden ist.

FUNCTION F_iecLookupTableEntry: UDINT

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

key: Lookup-Schlüssel [▶ 323].

hTable: Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

Rückgabeparameter	Beschreibung
0	Kein Fehler. Es existiert ein zum Schlüssel passender Tabelleneintrag.
<> 0	Es wurde kein Tabelleneintrag gefunden. Fehler: IEC60870-5-10x Fehlercode

Beispiel in ST:

Es wird die Existenz von drei Datenpunkten in der Applikationsdatenbank überprüft. Die gesuchten Datenpunkte sind bereits als Hash-Tabelleneinträge konfiguriert worden. Siehe in der Beschreibung der Funktion: F_iecAddTableEntry [▶ 294].

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
```

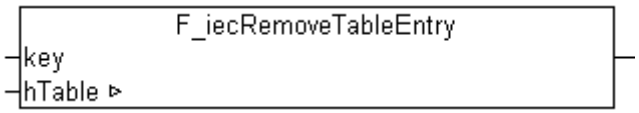
```
IF bLookup THEN
    bLookup := FALSE;

    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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.21 F_iecRemoveTableEntry

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.



Die Funktion F_iecRemoveTableEntry entfernt einen zum Schlüssel passenden Hash-Tabelleneintrag.

FUNCTION F_iecRemoveTableEntry: UDINT

```
VAR_INPUT
    key : ST_IEC870_5_101HashTableKey;
END_VAR
```

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

key: Lookup-Schlüssel [▶ 323].

hTable: Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

Rückgabeparameter	Beschreibung
0	Kein Fehler, der Tabelleneintrag wurde erfolgreich entfernt.
<> 0	Fehler: IEC60870-5-10x Fehlercode

Beispiel in ST:

Es werden drei Hash-Tabelleneinträge aus der Applikationsdatenbank entfernt. Die gesuchten Datenpunkte sind bereits als Hash-Tabelleneinträge konfiguriert worden. Siehe in der Beschreibung der Funktion: F_iecAddTableEntry [▶ 294].

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.22 F_iecCmpAddrOctets

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-101/104 Unterstation v3.0.2 / IEC60870-5-104 Zentralstation v1.0.2 und höher.

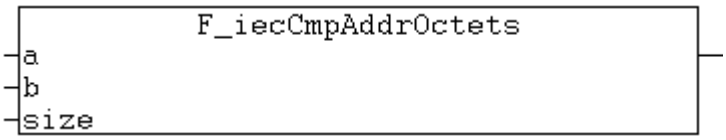


Abb. 3: F_iecCmpAddrOctets

Diese Funktion vergleicht zwei Adressen (z.B. Idie Verbindungsadresse, Objektadresse oder die gemeinsame ASDU-Adresse).

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: Erste Adresse.

b: Zweite Adresse.

size: Bytegrösse der Adresse.

Rückgabeparameter	Bedeutung
TRUE	(a und b sind gleich) oder (b ist eine Adresse an alle(broadcast)) oder (size ist 0),
FALSE	Alle anderen Fälle.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1307	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.23 F_iecGetSPI

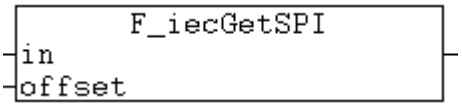


Abb. 4: F_iecGetSPI

Mit dieser Hilfsfunktion kann aus einer Byte-Variablen die Information einer Einzelmeldung (single point information) ausmaskiert werden. In der TwinCAT SPS belegt die Information einer Einzelmeldung 1 Bit an Prozessdaten. In einer Byte-Variablen können somit Informationen von bis zu 8 Einzelmeldungen gemappt werden.

FUNCTION F_iecGetSPI: E_IEC870_5_101SPI

E_IEC870_5_101SPI [▶ 341]

```
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
```

Beispiel:

Die Information der vier Einzelmeldungen wird ausmaskiert.

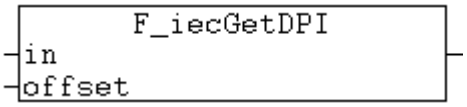
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eSPI : E_IEC870_5_101SPI;
ENd_VAR
```

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.24 F_iecGetDPI



Mit dieser Hilfsfunktion kann aus einer Byte-Variablen die Information einer Doppelmeldung (double point information) ausmaskiert werden. In der TwinCAT SPS belegt die Information einer Doppelmeldung 2 Bit an Prozessdaten. In einer Byte-Variablen kann somit die Information von bis zu 4 Doppelmeldungen gemappt werden.

FUNCTION F_iecGetDPI: E_IEC870_5_101DPI

E_IEC870_5_101DPI [▶ 341]

```
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
```

Beispiel:

Die Information der vier Doppelmeldungen wird ausmaskiert.

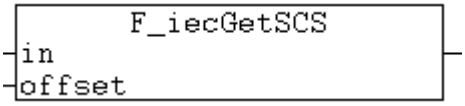
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eDPI : E_IEC870_5_101DPI;
ENd_VAR
```

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.25 F_iecGetSCS



Mit dieser Hilfsfunktion kann aus einer Byte-Variablen der Status eines Einzelbefehls (single command state) ausmaskiert werden. In der TwinCAT SPS belegt der Status eines Einzelbefehls 1 Bit an Prozessdaten. In einer Byte-Variablen kann somit der Status von bis zu 8 Einzelbefehlen gemappt werden.

FUNCTION F_iecGetSCS: E_IEC870_5_101SCS

E_IEC870_5_101SCS [▶ 336]

```
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
```

Beispiel:

Der Status der vier Einzelbefehle wird ausmaskiert.

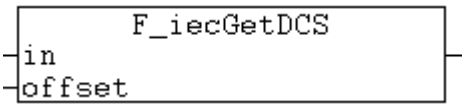
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eSCS : E_IEC870_5_101SCS;
ENd_VAR
```

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.26 F_iecGetDCS



Mit dieser Hilfsfunktion kann aus einer Byte-Variablen der Status eines Doppelbefehls (double command state) ausmaskiert werden. In der TwinCAT SPS belegt der Status eines Doppelbefehls 2 Bit an Prozessdaten. In einer Byte-Variablen kann somit der Status von bis zu 4 Doppelbefehlen gemappt werden.

FUNCTION F_iecGetDCS: E_IEC870_5_101DCS

E_IEC870_5_101DCS [▶ 336]

```
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
```

Beispiel:

Der Status der vier Doppelbefehle wird ausmaskiert.

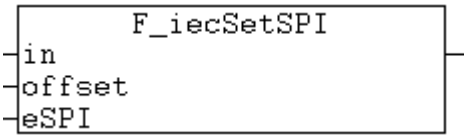
```
PROGRAM MAIN
VAR
    memarea AT%MB0 : ARRAY[0..10] OF BYTE;
    eDCS : E_IEC870_5_101DCS;
ENd_VAR
```

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.27 F_iecSetSPI



Diese Hilfsfunktion kopiert die Information einer Einzelmeldung (single point information) in eine Byte-Variable. In der TwinCAT SPS belegt die Information einer Einzelmeldung 1 Bit an Prozessdaten. In einer Byte-Variablen können somit Informationen von bis zu 8 Einzelmeldungen gemappt werden.

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
```

Beispiel:

Die Information der vier Einzelmeldungen wird auf ON gesetzt.

```
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 );
```

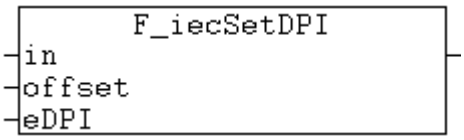
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

Sehen Sie dazu auch

E_IEC870_5_101SPI [▶ 341]

5.2.28 F_iecSetDPI



Diese Hilfsfunktion kopiert die Information einer Doppelmeldung (double point information) in eine Byte-Variable. In der TwinCAT SPS belegt die Information einer Doppelmeldung 2 Bit an Prozessdaten. In einer Byte-Variablen kann somit die Information von bis zu 4 Doppelmeldungen gemappt werden.

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
```

Beispiel:

Die Information der vier Doppelmeldungen wird auf ON gesetzt.

```
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 );
```

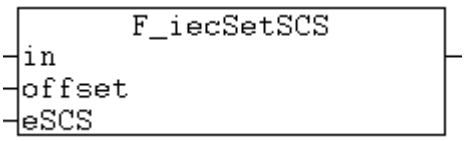
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

Sehen Sie dazu auch

E_IEC870_5_101DPI [▶ 341]

5.2.29 F_iecSetSCS



Diese Hilfsfunktion kopiert den Status eines Einzelbefehls (single command state) in eine Byte-Variable. In der TwinCAT SPS belegt der Status eines Einzelbefehls 1 Bit an Prozessdaten. In einer Byte-Variablen kann somit der Status von bis zu 8 Einzelbefehlen gemappt werden.

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
```

Beispiel:

Der Status der vier Einzelbefehle wird auf ON gesetzt.

```
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 );
```

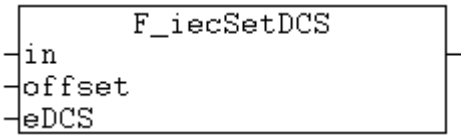
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

Sehen Sie dazu auch

E_IEC870_5_101SCS [▶ 336]

5.2.30 F_iecSetDCS



Diese Hilfsfunktion kopiert den Status eines Doppelbefehls (double command state) in eine Byte-Variable. In der TwinCAT SPS belegt der Status eines Doppelbefehls 2 Bit an Prozessdaten. In einer Byte-Variablen kann somit der Status von bis zu 4 Doppelbefehlen gemappt werden.

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
```

Beispiel:

Der Status der vier Doppelbefehle wird auf ON gesetzt.

```
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 );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

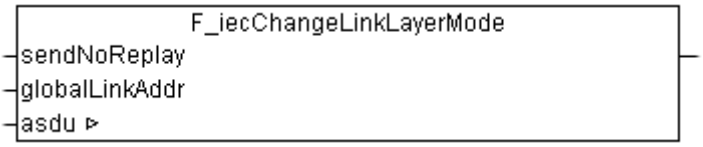
Sehen Sie dazu auch

E_IEC870_5_101DCS [▶ 336]

5.2.31 F_iecChangeLinkLayerMode

Ab der Produktversion:

- **TwinCAT PLC Library IEC60870-5-101 Zentralstation v2.0.3;**
- **TwinCAT PLC Library IEC60870-5-101 Unterstation v4.0.3;**
- **TwinCAT PLC Library IEC60870-5-102 Zentralstation v2.0.3;**
- **TwinCAT PLC Library IEC60870-5-103 Zentralstation v2.0.3;**



Mit dieser Funktion kann das Default-Verhalten jedes einzelnen ASDU-Frames auf der Link-Layer-Ebene in der Primärstation festgelegt bzw. verändert werden. Das zu versendende ASDU-Frame kann als SEND/NO REPLY-Frame (Funktion 4) oder Broadcast-Frame gekennzeichnet werden. Die Kennzeichnung der Frames erfolgt vor dem Ablegen des Frames in dem TX-Puffer. D.h. vor dem Aufruf der FB_IEC870_5_101TBufferCtrl [▶ 269].TxAddObj-Aktion.

Wenn Sie die Funktion nicht verwenden, dann werden in der Primärstation alle Frames als SEND/CONFIRM-Frames (Funktion 3) versendet. Als Linkadresse wird in diesem Fall die konfigurierte Stationsadresse verwendet.

- Im Unbalanced-Modus hat die Verwendung dieser Funktion in der Unterstation keine Bedeutung. Die Unterstation agiert in diesem Modus niemals als Primärstation.
- Bei der Verwendung des IEC 6087-5-104 Protokolls hat die Funktion keine Bedeutung.
- Diese Funktionalität wird nur bei der Verwendung der "Low level"-Schnittstelle unterstützt.

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 : Dieser Parameter legt fest ob beim Versenden des Frames die Link-Layer-Funktion: SEND/NO REPLY (TRUE = Funktion 4) oder SEND/CONFIRM (FALSE = Funktion 3) verwendet werden soll;

globalLinkAddr : Dieser Parameter legt fest ob beim Versenden des Frames statt der konfigurierten Stations-Linkadresse eine globale (Broadcast) Adresse verwendet werden soll. Beim Wert TRUE wird als Linkadresse im gesendeten Frame 16#FF bzw. 16#FFFF verwendet (one octet size, two octets size link address);

asdu : Das zu versendende ASDU-Frame [▶ 312] als VAR_IN_OUT-Variable;

Rückgabeparameter	Beschreibung
FALSE	Funktion fehlgeschlagen.
TRUE	Kein Fehler.

Beispiel 1 (Ausschnitt)

Die spontanen Daten eines Bitstrings sollen mit der Hilfe der Funktion SEND/NO REPLY an die Zentralstation gesendet werden (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

...
```

Die txAsdu-Variable wird zum Versenden weiterer Datenpunkte verwendet. Die Default-Konfiguration des txAsdu-Frames wird durch einen weiteren F_iecChangeLinkLayerMode(FALSE, FALSE, ...)-Funktionsaufruf hergestellt. Andere ASDUs sollen nicht als SEND/NO REPLY-Telegramme gesendet werden.

Beispiel 2 (Ausschnitt)

Ein Einzelbefehl soll mit der Hilfe der Funktion SEND/NO REPLAY an die Unterstation gesendet werden (Unbalanced-Mode).

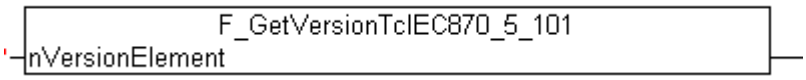
```
...

(* 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

...
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build >= 1554	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.2.32 F_GetVersionTcIEC870_5_101



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_101: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3 Datentypen

5.3.1 ST_IEC870_5_101TBuffer

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 / IEC60870-5-101 Unterstation v2.0.2 und höher.

Diese Datenstruktur wird beim Datenaustausch (TX/RX-Datenpuffer) über das IEC60870-5-104/101 Transport Interface benutzt.

```
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-Ausgabe-Parameter [▶ 334].

asduFmt: ASDU-Formatparameter [▶ 318].

asduSize: Maximale Bytelänge der ASDU.

mode: Reserviert, wird zur Zeit nicht benutzt. Dieser Wert sollte Null sein.

dataLink: Auf die Elemente dieser Datenstruktur sollte nicht direkt, sondern nur mit einer Instanz des FB_IEC870_5_101TBufferCtrl [▶ 269]-Funktionsbausteins zugegriffen werden.

Der TX/RX-Datenpuffer verwendet intern zwei Sende-Fifos und einen Empfangs-Fifo:

1. Class 1 Sende-Fifo mit (hochprioren) Daten;
2. Class 2 Sende-Fifo mit (niederprioren) Daten;
3. Receive-Fifo (für Class 1 und Class 2 Daten);

Die unteren Transportfunktionen der Bibliothek leeren zuerst den Class 1-Fifo und dann den Class 2-Fifo. Die Class 2 Daten werden nur dann versendet, wenn der Class 1-Fifo keine zu versendenden Daten enthält.

Jeder der internen Fifos hat eine feste Größe von 10000 Bytes. Dies dürfte für die meisten Anwendungen ausreichen, weil die Anzahl der Frames die einmalig verschickt, werden können durch die iK und iW-Protokollparameter begrenzt wird. Erfahrungsgemäß können in jedem Fifo ca. 200 ASDUs mit einem Informations-Element (Objekt) oder ca. 20 ASDUs mit einer Sequence von 100 Informations-Elementen (Objekten) abgelegt werden.

Wenn eine größere Anzahl der zu versendenden oder zu empfangenen Frames zwischengespeichert werden soll (z.B. ~20000), so können diese in externen, vom SPS-Programmierer festgelegten Puffern/Fifos zwischengehalten werden. Die SPS-Applikation kann dann zur Laufzeit die TwinCAT-Sende-Fifos mit den eigenen Fifo-Einträgen nachfüllen oder bei vielen empfangenen Frames den TwinCAT-Receive-Fifo leeren. Eine andere Möglichkeit ist z.B. zwei Puffer zu benutzen und diese abwechselnd zu füllen/lesen und an den Kommunikationsbaustein übergeben.

bOverwrite: Implementiert ab IEC870-5-104 Slave Library **v3.0.14** und höher. Aktiviert/deaktiviert das Überschreiben der ältesten Meldungen im Sendepuffer bei der Überschreitung der max. zulässigen Puffergröße. Dieser Parameter sollte nur im Offline-Mode aktiviert werden (d.h. wenn die Verbindung unterbrochen wurde) und wenn Offline-Datenspeicherung zusätzlich über den **bRetainBuffer**-Parameter in den Protokollparametern (ST_IEC870_5_10xProtPara) aktiviert wurde. Im Online-Mode (d.h. wenn der Datentransfer gestartet wurde) sollten keine älteren Meldungen überschrieben werden (sonst fehlen möglicherweise einige Zwischenwerte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

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
```

Ein IEC Applikationsobjekt-Datenbankeintrag. Die IEC-Applikationsobjekt Datenbank wird als Array-Variable von Typ ST_IEC870_5_101AODBEntry deklariert. Die Membervariablen dieses strukturierten Typs werden nicht direkt, sondern nur mit Hilfe der zur Verfügung stehenden Funktionen oder Funktionsbausteine verändert. Die F_iecIniAOEntry [▶ 277] gehört z.B. zu einer solchen Funktion.

Beispiel für eine Deklaration von einer Applikationsdatenbank mit 2001 Objekten:

```
VAR_GLOBAL
    slave1AODB : ARRAY[0..2000] OF ST_IEC870_5_101AODBEntry;
ENd_VAR
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.3 ST_IEC870_5_101AOGen

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.5 und höher.

Variablen von diesem Typ repräsentieren ein ASDU-Objekt.

```
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 (reserviert).

ident:Identifikationsfelder [▶ 313] der Dateneinheit (ASDU).

info: Informationsobjekt-/Informationselement-Datenfeld [▶ 314].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.4 ST_IEC870_5_101AOCfg

Ab der Produktversion: TwinCAT PLC Library: IEC60870-5-104 Unterstation v3.0.0 / IEC60870-5-104 Zentralstation v1.0.0 und höher.

ASDU-Objekt-Konfigurationsparameter. Diese Parameter werden während der Konfiguration der Datenpunkte (F_iecInitAOEntry oder F_iecAddTableEntry) gesetzt und sollten nicht direkt aus der SPS-Applikation beschrieben werden.

```
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-Konfigurationsflags. Hier finden Sie die Beschreibung aller Group-Flags [▶ 347]. Die Flags können mit OR-Verknüpfung kombiniert werden. Nicht alle Kombinationen sind aber sinnvoll!

multiplier: Basis-Zykluszeit-Multiplikator für zyklische/periodische Datenübertragung. 0 = Deaktiviert. Die Basis-Zykluszeit kann über den *tPerCyclicBase*-Parameter in den Systemparametern [▶ 315] konfiguriert werden.

opt: Reserviert.

ioMapType: TwinCAT SPS-Prozessdatenbereich. Dieser Parameter [▶ 329] legt fest wie die TwinCAT SPS und IEC-Applikationsobjekt Prozessdaten gemappt werden sollen.

byteOffs: TwinCAT SPS-Prozessdaten-Byte-Offset.

bitOffs: TwinCAT SPS-Prozessdaten-Bit-Offset.

hUser: Benutzer-Handle. Frei definierbarer 32-Bit-Wert.

ext: Reserviert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.5 ST_IEC870_5_101DataUnit_Ident

Identifikationsfeld der Dateneinheit (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: Typ [▶ 326].

bSQ: Sequence-Flag.

nObj: Anzahl der Informationsobjekte oder Informationselemente.

bT: Test-Flag

bPN: Positive/negative Bestätigung.

eCOT: Übertragungsursache [▶ 332].

nORG: Quelladresse.

asduAddr: Gemeinsame ASDU-Adresse

eClass: Prioritätsklasse [▶ 332].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.6 ST_IEC870_5_101AOInfoObj

Informationsobjektbeschreibung.

```
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Sehen Sie dazu auch

- ST_IEC870_5_101Stream [▶ 314]

5.3.7 ST_IEC870_5_101Stream

Variablen von diesem Typ werden als Datenpuffer (Stream) benutzt. Im Stream werden "rohe" Datenbytes zwischengespeichert, mit der Reihenfolge, wie sie später verschickt werden sollen oder wie sie empfangen wurden. Beim Senden oder Empfang wird zuerst das Datenbyte Null verschickt bzw. empfangen.

```
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: Aktuelle Anzahl der Datenbytes im Stream;

data: Stream-Datenpuffer;

Speicherdarstellung einer Stream-Variablen mit einer zwischengespeicherten DWORD-Variablen mit dem Wert: **16#BECF1234**. Beachten Sie die vertauschen Datenbytes im Intel-Format!

length	data											
4	16#34	16#12	16#CF	16#BE	...	...	...	...	...	...	...	IEC870_MAX_ASDU_DATA_BYTE

Benutzen Sie die in der Tabelle aufgeführten Funktionen, um den Speicherinhalt einer Stream-Variablen zu verändern:

Function	Description
F_iecResetStream [▶ 292]	Stream-Initialisierung/-Reset.
F_iecCopyBufferToStream [▶ 285]	Kopiert Datenbytes von einer externen Puffervariablen in den Stream. Der Speicherinhalt der Stream-Variablen wird vergrößert.
F_iecCopyStreamToBuffer [▶ 286]	Kopiert Datenbytes vom Stream in eine externe Puffervariable. Der Speicherinhalt der Stream-Variablen bleibt unverändert.
F_iecCopyStreamToStream [▶ 288]	Kopiert Datenbytes von einem Stream zum anderen Stream. Der Speicherinhalt der Quell-Variablen bleibt unverändert. Der Speicherinhalt der Ziel-Variablen wird vergrößert.
F_iecMoveStreamToBuffer [▶ 289]	Kopiert Datenbytes vom Stream in eine externe Puffervariable und löscht anschließend die kopierten Datenbytes im Stream. Der Speicherinhalt der Stream-Variablen wird verkleinert.
F_iecMoveStreamToStream [▶ 291]	Kopiert Datenbytes vom Quell-Stream zum Ziel-Stream und löscht anschließend die kopieren Datenbytes im Quell-Stream. Der Speicherinhalt der Quell-Variablen wird verkleinert und der Ziel-Variablen vergrößert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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: Gemeinsame ASDU-Adresse in Standardrichtung. Default: 11.

asduAddrRev: Gemeinsame ASDU-Adresse in Umkehrrichtung. Default: 22.

asduFmt: ASDU-Formatparameter [▶ 318] (z.B. Oktetlänge der Übertragungsursache-, ASDU-Adresse- und der Object-Adresse);

bEndOfInit: Wenn TRUE, sende M_EI_NA_1 (End of init) nachdem die Stationsinitialisierung abgeschlossen wurde. Eine Stationsinitialisierung wird beim Systemstart oder nach einem Prozess Reset durchgeführt. Default: TRUE.

bSyncTime: Wenn TRUE, aktiviere IEC-Systemzeit Synchronisation beim Empfang des C_CS_NA_1-Kommandos. Default: TRUE.

bSyncPCTime: Wenn TRUE, synchronisiere neben der IEC-Systemzeit auch die Systemzeit des TwinCAT PC's (die Windows-Systemzeit in der Taskleiste). Default: FALSE.

bUsePCTime: Wenn TRUE, synchronisiere die IEC-Systemzeit mit der Systemzeit des PC's. Nach der Initialisierung der Unterstation wird die IEC-Systemzeit zuerst mit der Systemzeit des TwinCAT PC's (Windows-Systemzeit) synchronisiert. Default: TRUE.

tSyncToPCTime : Steuert die zyklische Synchronisation der IEC-Systemzeit mit der Systemzeit des TwinCAT PC's (im Online- und Offline-Mode). Implementiert in IEC870-5-104 slave Bibliothek v3.0.3 und höher.

Die Zeit bestimmt die Zykluszeit in der die Synchronisation durchgeführt wirde. Bei einem Wert = T#0s ist die zyklische Synchronisation deaktiviert. Default: T#0s.

sPCTimeNetID: TwinCAT Netzwerkadresse des PCs dessen Systemzeit bei der Synchronisierung benutzt werden soll. Für den lokalen PC kann auch ein Leerstring angegeben werden. Default: Leerstring = Lokaler PC.

bTimeCOT3: Wenn TRUE, sende die Systemzeit zur Zentralstation mit der Übertragungsursache <*3*> *Spontan* beim Stundenwechsel. Default: FALSE.

tSyncTimeout: Uhrzeitsynchronisationsintervall-Timeoutüberwachung. Implementiert in IEC870-5-101/104 slave Bibliothek v2.0.0 und höher. In früheren Versionen wird dieser Parameter nicht benutzt. Die Zeitstempel haben einen IV-Quality-Flag (invalid). Bei einem Wert <> T#0m (z.B. T#60m) wird das IV-Quality-Flag bei allen darauffolgenden Zeitstempeln auf Invalid gesetzt wenn innerhalb einer Stunde keine Uhrzeitsynchronisation durchgeführt wurde. Das IV-Quality-Flag des Zeitstempels kann vom Master als Indikator für die Qualität des Zeitstempels benutzt werden. Bei tSyncTimeout = T#0m ist die Überwachung nicht aktiv.

bACTCONRes: Wenn TRUE, sende ACTCON response.

bACTTERMRes: Wenn TRUE, sende ACTTERM response.

bPerCyclic: Wenn TRUE, aktiviere zyklische/periodische Datenübertragung.

tPerCyclicBase: Basiszeit der zyklischen/periodischen Datenübertragung.

bBackScan: Wenn TRUE, aktiviere Hintergrundabfrage.

tBackScanCycle: Hintergrundabfrage-Zykluszeit.

bPerFRZ: Wenn TRUE, aktiviere lokales Umspeichern/Rücksetzen der Zählerwerte.

tPerFRZCycle: Zykluszeit für lokales Umspeichern/Rücksetzen.

tSelExecTimeout: Max. Timeoutzeit zwischen dem Anwahlbefehl und dem Ausführungsbefehl (nur Zentralstation).

tActConTimeout :Max. Timeoutzeit für den Empfang der Bestätigung der Befehlaktivierung (nur Zentralstation).

tActTermTimeout : Max. Timeoutzeit für den Empfang der Befehlterminierung (nur Zentralstation).

tDeactConTimeout : Max. Timeoutzeit für den Empfang der Bestätigung des Befehlabbruchs (nur Zentralstation).

tReadResTimeout : Max. Timeoutzeit für die Ausführung des Lesebefehls (C_RD_NA_1) (nur Zentralstation).

dbgMode:Debug-Flags:

Wert	Beschreibung
IEC870_DEBUGMODE_DISABLED	Loggen deaktiviert
IEC870_DEBUGMODE_ASDU	Die ASDUs werden als hexadezimale Ausgabe im Application-Log geloggt.
IEC870_DEBUGMODE_DEVSTATE	Statusänderungen der Station werden im Application-Log geloggt.
IEC870_DEBUGMODE_LINKLAYER	Link layer frames werden im Application-Log geloggt IEC870-5-101 slave Bibliothek: bei allen Versionen verfügbar; IEC870-5-104 slave Bibliothek: ab v2.0.0 und höher verfügbar;
IEC870_DEBUGMODE_LINKERROR	Link layer Fehlermeldungen werden im Application-Log geloggt.

Die Flags können in der gewünschten Kombination verodert werden.

orgAddr: Quelladresse (wird zurzeit nicht benutzt).

bOverwrite: Implementiert ab IEC870-5-104 slave library **v3.0.14** und höher. Aktiviert/deaktiviert das Überschreiben der ältesten Meldungen im Sendepuffer bei der Überschreitung der max. zulässigen Puffergrösse. Dieser Parameter ist nur im Offline-Mode aktiv (d.h. wenn die Verbindung unterbrochen wurde) und wenn Offline-Datenspeicherung zusätzlich über den **bRetainBuffer**-Parameter in den Protokollparametern (ST_IEC870_5_104ProtPara) aktiviert wurde. Im Online-Mode (d.h. wenn der Datentransfer gestartet wurde) werden keine älteren Meldungen überschrieben (sonst würden möglicherweise einige Zwischenwerte fehlen).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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: Gemeinsame ASDU-Adresse in Standardrichtung;

asduAddrRev: Gemeinsame ASDU-Adresse in Umkehrrichtung.

dataLink: Interne ASDU-Send/Receive-Fifos;

clock: IEC-Systemzeitobjekt;

comp: Kompatibilitätsliste mit der Zuordnung der ASDU-Typen zu den Übertragungsursachen;

errors: Gerätefehler-Fifo [▶ 268];

status: Globaler-Gerätestatus. 1=OK, 0=Not OK;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Sehen Sie dazu auch

- E_IEC870_5_101COTSize [▶ 330]
- E_IEC870_5_101AsduAddrSize [▶ 330]
- E_IEC870_5_101ObjAddrSize [▶ 331]

5.3.11 ST_IEC870_5_101ErrorFifoEntry

IEC60870-5-10x Gerätefehler.

```
TYPE ST_IEC870_5_101ErrorFifoEntry :
STRUCT
    nErrId : UDINT;
    eSrcId : E_IEC870_5_101ErrorSourceID;
END_STRUCT
END_TYPEEND_TYPE
```

nErrID: Fehlercode.

eSrcID: Fehlerquelle [▶ 331]. Die Fehlerquelle gibt nähere Informationen über die Fehlerursache, die Komponente oder den Dienst der den Fehler gemeldet hat.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.12 ST_IEC870_5_101AcquisitionParams

Ab der Produktversion:

- **TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher;**
- **TwinCAT PLC Library IEC60870-5-101 Zentralstation v1.0.1 und höher;**

Konfigurationsparameter für die zyklische Datenerfassung. Die Initialisierungssequence, Kommandos, die im *arrInitSeq*-Array konfiguriert wurden, werden einmalig nach dem Empfang von M_EI_NA_1 (end of initialisation) oder nach der Herstellung der Kommunikationsverbindung ausgeführt. Danach beginnt die Ausführung der zyklischen Befehle, die über die Strukturparameter: *testCmd*, *clockSync*, *arrGenro, arrCoro odergenCmd* konfiguriert wurden.

```
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 *)
    eIEC870_ISTEP_UNUSED, (* Reserved *)
    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: Initialisierungssequence [▶ 335]. Die Initialisierungssequence wird immer nach dem Empfang von M_EI_NA_1 (end of initialisation) einmalig ausgeführt.

testCmd: Parameter für zyklische Testbefehle [▶ 321]. Standardwert: Ein Testkommando alle 60s.

clockSync: Parameter für zyklische Uhrzeitsynchronisationsbefehle [▶ 321]. Standardwert: Uhrzeitsynchronisation alle 60s.

arrGenro: Parameter für zyklische Stationsabfragebefehle [▶ 322]. Es können Stationsabfragen von bis zu 17 Datenpunktgruppen konfiguriert werden. Standardwert: Eine Stationsabfrage der Gruppe 'Allgemein' alle 60s.

arrCoro: Parameter für zyklische Zählerabfragebefehle [▶ 322]. Es können Zählerabfragen von bis zu 5 Zählergruppen konfiguriert werden. Standardwert: Eine Zählerabfrage der Gruppe Allgemein alle 60s.

genCmd: Parameter für zyklische Übertragung der Datenpunkte in Steuerungsrichtung [▶ 323] (Einzelbefehl, Doppelbefehl, Sollwerte usw.). Standardwert: Die Befehle werden alle 60min. übertragen.

delayCmd: Parameter für die Verzögerung [▶ 323] des nächsten Initialisierungsschrittes während der Ausführung der Initialisierungssequence. Standardwert: Der nächste Initialisierungsschritt wird um 5s verzögert.

eAODBType: Applikationsdatenbank-Typ [▶ 335]. Dieser Parameter legt fest wie die Datenpunkte in der Applikationsdatenbank abgespeichert werden.

Beispiel in ST:

Im folgenden Programmausschnitt wird die zyklische Datenerfassung wie folgt konfiguriert: Alle Initialisierungsschritte werden deaktiviert. Das zyklische Testkommando und Uhrzeitsynchronisationskommando werden auch deaktiviert. Zusätzlich zu der Standard-Stationsabfrage wird noch eine weitere Stationsabfrage der Gruppe 1 alle 100s konfiguriert. Außerdem wird zu der Standard-Zählerabfrage eine Zählerabfrage (Umspeichern) der Zählergruppe 1 alle 200s konfiguriert.

```
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;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.13 ST_IEC870_5_101TestPollParams

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.

Konfigurationsparameter für den zyklischen Testbefehl.

```
TYPE ST_IEC870_5_101TestPollParams:
STRUCT
    asduAddr    : DWORD := IEC870_ASDUADDR_SYSPARA;
    tPollCycle  : TIME := T#60s;
    bEnable     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

asduAddr: Zieladresse;

tPollCycle: Zykluszeit des Testbefehls.

bEnable: Aktiviert (TRUE) oder deaktiviert (FALSE) die zyklischen Testbefehle.

Voraussetzungen

Entwicklingsumgebung	Zielsystem	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.14 ST_IEC870_5_101ClockPollParams

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.

Konfigurationsparameter für den zyklischen Uhrzeitsynchronisationsbefehl. Die Station, welche die Uhrzeitsynchronisationsbefehle sendet, besitzt eine eigene interne Software-Uhr. Diese Uhr wird während der Stationsinitialisierung mit der lokalen Windows-Systemzeit (der Uhrzeit in der Windows Taskleiste) synchronisiert. Im Betrieb werden die Zielstationen über die Uhrzeitsynchronisationsbefehle mit der Uhrzeit der internen Software-Uhr synchronisiert.

```
TYPE ST_IEC870_5_101ClockPollParams:
STRUCT
    asduAddr    : DWORD := IEC870_ASDUADDR_SYSPARA;
    tPollCycle  : TIME := T#60s;
    bEnable     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

asduAddr: Zieladresse.

tPollCycle: Zykluszeit des Uhrzeitsynchronisationsbefehls.

bEnable: Aktiviert/deaktiviert die zyklischen Uhrzeitsynchronisationsbefehle.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.15 ST_IEC870_5_101GenroPollParams

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.

Konfigurationsparameter für den zyklischen Stationsabfragebefehl.

```
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: Zieladresse.

tPollCycle: Stationsabfrage-Zykluszeit.

eQOI: Abfrageparameter [▶ 337]/Kennung.

bEnable: Aktiviert (TRUE) oder deaktiviert (FALSE) die Stationsabfragebefehle.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.16 ST_IEC870_5_101CoroPollParams

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.

Konfigurationsparameter für den zyklischen Zählerabfragebefehl.

```
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: Zieladresse;

tPollCycle: Zykluszeit des Zählerabfragebefehls.

eRQT: Kennung für die Zählerabfrage [▶ 338].

eFRZ: FREEZE-/RESET [▶ 338]-Kennung.

bEnable: Aktiviert/deaktiviert die zyklischen Zählerabfragebefehle.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.17 ST_IEC870_5_101GenCmdPollParams

Parameter für zyklische Übertragung der Datenpunkte in Steuerungsrichtung (Einzelbefehl, Doppelbefehl, Sollwerte usw.).

```
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: Übertragungszykluszeit.

bEnable: Aktiviert/deaktiviert die zyklische Übertragung der Datenpunkte.

options: Zusätzliche Parameter.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.18 ST_IEC870_5_101DelayPollParams

Konfigurationsparameter für die Verzögerung des nächsten Initialisierungsschrittes.

```
TYPE ST_IEC870_5_101DelayPollParams:
STRUCT
    tDelay : TIME := T#5s; (* Delay time *)
END_STRUCT
END_TYPE
```

tDelay: Verzögerungszeit.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.19 ST_IEC870_5_101HashTableKey

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v3.0.0 / IEC60870-5-104 Zentralstation v1.0.0 und höher.

Applikationsobjekt-Datenbank-Lookup-Schlüssel. Mit Hilfe des Schlüssels können die Hash-Tabelleneinträge lokalisiert und modifiziert werden.

```
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: Applikationsobjekt Typ, ASDU identifier [▶ 326] (z.B.: M_SP_NA_1 für Single-Point oder M_DP_NA_1 für Double-Point usw.).

asduAddr: Gemeinsame ASDU-Adresse.

objAddr: Objektadresse, frei wählbar.

group: Object-Group-Konfigurationsflags. Hier finden Sie die Beschreibung aller Group-Flags [▶ 347]. Die Flags können mit OR-Verknüpfung kombiniert werden. Wenn dieser Parameter den Wert Null hat werden die group-Parameter ignoriert.

lookup: Zusätzliche Lookup-Schlüsselparameter. Die verfügbaren Parameter sind als Konstanten deklariert (siehe untere Tabelle). Diese können mit OR-Verknüpfung kombiniert werden.

Konstante	Wert	Beschreibung
IEC870_LOOKUP_KEY_ALL_ON	0	Bei der Suche werden alle Parameter berücksichtigt (*eType, asduAddr, objAddr, group*).
IEC870_LOOKUP_KEY_TYPE_OFF	1	Bei der Suche wird der *eType*-Parameter ignoriert.
IEC870_LOOKUP_KEY_ASDUADDR_OFF	2	Bei der Suche wird der *asduAddr*-Parameter ignoriert.
IEC870_LOOKUP_KEY_GROUP_OFF	4	Bei der Suche wird der *group*-Parameter ignoriert.
IEC870_LOOKUP_KEY_OBJADDR_OFF	8	Bei der Suche wird der *objAddr*-Parameter ignoriert. Dieser Parameter ist nicht zu empfehlen da alle Datenpunkte über eine eindeutige Objektadresse identifizierbar sein müssen.

Beispiel in ST:

Siehe in der Beschreibung der Funktion: F_iecLookupTableEntry [▶ 298], F_iecRemoveTableEntry [▶ 299], F_iecGetPosOfTableEntry [▶ 296].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.20 ST_IEC870_5_101FBufferCfg

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-101/104 Unterstation v3.0.2 / IEC60870-5-104 Zentralstation v1.0.2 und höher.

Konfigurationseinstellungen für den Offline-ASDU-Dateipuffer. Diese Struktur wird von dem FB_IEC870_5_101FBufferCtrl [▶ 271] Funktionsbaustein benutzt.

```
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 fut
ure use *)
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT; (* ADS (file access) timeout *)
ND_STRUCT
END_TYPE
```

sNetID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Pufferdatei erstellt werden soll. Für den lokalen PC kann auch ein Leerstring angegeben werden.

sPathName: Enthält den Pfad- und Dateinamen der zu öffnenden Datei. Der Pfad kann nur auf das lokale File System des Rechners zeigen. Das bedeutet, Netzwerkpfade können hier nicht angegeben werden.

ePath: Über diesen Eingang kann ein TwinCAT - Systempfad auf dem Zielgerät zum Öffnen der Datei angewählt werden.

cbBuffer: Max. Bytegröße der Pufferdatei. Wenn bOverwrite = FALSE gesetzt wurde und die max. Größe überschritten wurde wird ein Fehler zurückgemeldet.

bOverwrite: Beim Erreichen der max. Größe werden die ältesten Einträge überschrieben, wenn diese Variable auf TRUE gesetzt wurde.

bFilter: Zurzeit noch nicht implementiert. Aktiviert/deaktiviert einen COT-Filter (Cause of transfer). Nur ASDUs mit bestimmten Übertragungsursachen werden in die Datei gepuffert.

cotFilter: Zurzeit noch nicht implementiert. Über diese Variable können Übertragungsursachen [▶ 344] (COTs) konfiguriert werden, die in die Datei gepuffert werden sollen.

tTimeout: Maximale Timeoutzeit die bei dem Dateizugriff nicht überschritten werden sollte.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1313	PC or CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.21 ST_IEC870_5_101FBufferStatus

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-101/104 Unterstation v3.0.2 / IEC60870-5-104 Zentralstation v1.0.2 und höher.

Statusinformationen zum Offline ASDU Dateipuffer. Diese Struktur wird von dem FB_IEC870_5_101FBufferCtrl [▶ 271] Funktionsbaustein benutzt.

```
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: Liefert den Dateipuffer Status [▶ 336] (Datei wird geschrieben, geladen, ist geschlossen oder es ist ein Fehler beim Schreiben/Laden aufgetreten).

nErrID: Liefert bei einem Dateizugriffsfehler die ADS-Fehlernummer.

bCorrupted: Bei TRUE war die zuletzt geöffnete Datei als korrupt erkannt und wurde durch eine neue leere Datei ersetzt. Eine korrupte Datei kann z.B. dann entstehen, wenn die max. Dateipuffergröße geändert wurde oder die Datei nicht richtig geschlossen wurde.

nCount: Aktuelle Anzahl der gepufferten Einträge im Dateipuffer. Der Dateipuffer muss zuerst geöffnet werden, um die Anzahl der Einträge in einer existierenden Datei ermitteln zu können. D.h. die Verbindung muss kurz in den Offline-Mode gehen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1313	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.22 E_IEC870_5_101TcTypeID

Datenpunkt-Typbezeichner.

```
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 synchronisation 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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

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 SPS-Prozessdatenbereich (inputs, outputs, memory, data) in den bzw. aus dem die IEC-Prozessdaten gemappt (kopiert) werden sollen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.28 E_IEC870_5_101ErrorSourceID

Fehlerquelle.

```
TYPE E_IEC870_5_101ErrorSourceID :
(
    eIEC870_ESRC_NONE,                  (* Fehlerquelle ist unbekannt *)
    eIEC870_ESRC_COUNTER_INTERROGATION,         (* Fehler während der Ausführung der Zählwertabfrage
 *)
    eIEC870_ESRC_SYNC_CLOCK_CTRL,       (* Fehler während der Ausführung der Uhrzeitsynchronisation
*)
    eIEC870_ESRC_CLOCK_EVENT,               (* Fehler während der spontanen Übertragung der Uhrzeit
*)
    eIEC870_ESRC_GETPCTIME,             (* Fehler während der Synchronisierung der Geräte-
Uhrzeit mit der Uhrzeit des PC's *)
    eIEC870_ESRC_SETPCTIME,             (* Fehler während der Synchronisierung der PC-
Uhrzeit mit der Uhrzeit des Gerätes *)
    eIEC870_ESRC_STATION_INTERROGATION,         (* Fehler während der Generalabfrage *)
    eIEC870_ESRC_READ_DATA_CMD,         (* Fehler beim Lesekommando *)
    eIEC870_ESRC_RESET_PROCESS,         (* Fehler beim Process-Reset-Kommando *)
    eIEC870_ESRC_TEST_CMD,          (* Fehler beim Testkommando *)
    eIEC870_ESRC_ENDOFINIT,         (* Fehler bei M_EI_NA_1 (Ende der Initialisierung) *)
    eIEC870_ESRC_BACKGROUND_SCAN,       (* Fehler bei der Ausführung der Hintegrundabfrage *)
    eIEC870_ESRC_COMMAND_CTRL,              (* Fehler bei der Initialisierung des Befehls (single-
command, double-command, setpoint-command ...) *)
    eIEC870_ESRC_COMMAND_EXEC,              (* Fehler bei der Ausführung des Befehls *)
    eIEC870_ESRC_LOCAL_FREEZE_RESET,        (* Fehler beim Umspeichern / Zurücksetzen *)
    eIEC870_ESRC_PERIODIC_CYCLIC,       (* Fehler bei der periodischen/
zyklischen Datenübertragung *)
    eIEC870_ESRC_USERAPP_OBJECT,        (* Fehler im Applikationsobjekt (fehlerhafte Konfiguration/
Wert) *)
    eIEC870_ESRC_USERAPP_SETQUALITY,        (* Fehler bei der Quality-Flag-Bearbeitung *)
    eIEC870_ESRC_IEC60870_5_104LINK,        (* Fehler im IEC60870-5-104 Link Layer *)
    eIEC870_ESRC_IEC60870_5_101LINK,        (* Fehler im IEC60870-5-101 Link Layer *)
    eIEC870_ESRC_COMLIB,            (* Fehler in der unterlagerten Kommunikation über die seriellen
Schnittstellen *)
    eIEC870_ESRC_POLLING_SEVICE             (* Fehler in der Acquisition-State-
Machine (Abarbeitung der zyklische Pollanfragen)*)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.29 E_IEC870_5_101ClassType

```
TYPE E_IEC870_5_101ClassType :
(
    eIEC870_Class_None,
    eIEC870_Class_1,
    eIEC870_Class_2
);
END_TYPE
```

Prioritätsklasse der ASDU. Hochpriore Daten werden der Klasse 1 zugeordnet und niederpriore Daten der Klasse 2.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.30 E_IEC870_5_101COTType

Übertragungsursache (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
```

Wert	Beschreibung
eIEC870_COT_UNUSED	Wird nicht benutzt
eIEC870_COT_CYCLIC	Zyklische Daten
eIEC870_COT_BACKGROUND	Hintergrundabfrage
eIEC870_COT_SPONTAN	Spontane Daten
eIEC870_COT_INIT	Ende der Initialisierung
eIEC870_COT_REQ	Read-Request
eIEC870_COT_ACT	Kommando-Aktivierung
eIEC870_COT_ACT_CON	Bestätigung der Kommando-Aktivierung
eIEC870_COT_DEACT	Kommando-Abbruch
eIEC870_COT_DEACT_CON	Bestätigung des Kommando-Abbruchs
eIEC870_COT_ACT_TERM	Terminierung der Kommando-Aktivierung
eIEC870_COT_RETREM	Erwidert wegen einem Fern-Kommando
eIEC870_COT_RETLOC	Erwidert wegen einem Lokal-Kommando
eIEC870_COT_FILE	Dateizugriff
eIEC870_COT_INROGEN	Stationsabfrage (allgemein)
eIEC870_COT_INRO1	Stationsabfrage der Gruppe 1
eIEC870_COT_INRO2	Stationsabfrage der Gruppe 2
eIEC870_COT_INRO3	Stationsabfrage der Gruppe 3
eIEC870_COT_INRO4	Stationsabfrage der Gruppe 4
eIEC870_COT_INRO5	Stationsabfrage der Gruppe 5
eIEC870_COT_INRO6	Stationsabfrage der Gruppe 6
eIEC870_COT_INRO7	Stationsabfrage der Gruppe 7
eIEC870_COT_INRO8	Stationsabfrage der Gruppe 8
eIEC870_COT_INRO9	Stationsabfrage der Gruppe 9
eIEC870_COT_INRO10	Stationsabfrage der Gruppe 10
eIEC870_COT_INRO11	Stationsabfrage der Gruppe 11
eIEC870_COT_INRO12	Stationsabfrage der Gruppe 12

Wert	Beschreibung
eIEC870_COT_INRO13	Stationsabfrage der Gruppe 13
eIEC870_COT_INRO14	Stationsabfrage der Gruppe 14
eIEC870_COT_INRO15	Stationsabfrage der Gruppe 15
eIEC870_COT_INRO16	Stationsabfrage der Gruppe 16
eIEC870_COT_REQCOGEN	Zählerabfrage (allgemein)
eIEC870_COT_REQCO1	Zählerabfrage der Gruppe 1
eIEC870_COT_REQCO2	Zählerabfrage der Gruppe 2
eIEC870_COT_REQCO3	Zählerabfrage der Gruppe 3
eIEC870_COT_REQCO4	Zählerabfrage der Gruppe 4
eIEC870_COT_UNKNOWN_TYPE	Unbekannter Typ
eIEC870_COT_UNKNOWN_CAUSE	Unbekannte Übertragungsursache
eIEC870_COT_UNKNOWN_ASDU_ADDRESS	Unbekannte gemeinsame Asdu-Adresse
eIEC870_COT_UNKNOWN_OBJECT_ADDRESS	Unbekannte Objekt-Adresse
eIEC870_COT_14..eIEC870_COT_19 eIEC870_COT_42..eIEC870_COT_43 eIEC870_COT_48..eIEC870_COT_63	Bereich, der reserviert/unbenutzt ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.31 E_IEC870_5_101FifoDbgFlags

Debug-Ausgabe-Flags. Bei einer aktivierten Debug-Ausgabe werden die ASDU-Hexbytes in TwinCAT System Manager Loggerfenster geloggt.

```
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
```

Beachten Sie, dass die Debug-Ausgabe eine höhere Systemauslastung erzeugt.

Wert	Beschreibung
eIEC870_FIFO_DBG_OFF	Keine Debug-Ausgabe
eIEC870_FIFO_DBG_PUT	Debug-Ausgabe beim Hinzufügen der Fifo-Elemente
eIEC870_FIFO_DBG_GET	Debug-Ausgabe beim Entfernen der Fifo-Elemente
eIEC870_FIFO_DBG_ALL	Debug-Ausgabe beim Hinzufügen und Entfernen der Fifo-Elemente

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.32 E_IEC870_5_101AODBType

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 IEC60870-5-104 Zentralstation v1.0.0 und höher.

Typ der Applikationsdatenbank. Zu Zeit wird nur der Typ eIEC870_AODB_STATIC unterstützt. D.h. alle Applikationsobjekte (Datenpunkte) müssen von der SPS-Applikation konfiguriert werden.

```
TYPE E_IEC870_5_101AODBType :
(
    eIEC870_AODB_STATIC  := 0,
    eIEC870_AODB_DYNAMIC := 1
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.33 E_IEC870_5_101InitSeqStep

Befehle, die nach dem Empfang vom M_EI_NA_1 (end of initialisation) oder nach der Herstellung der Kommunikationsverbindung einmalig ausgeführt werden sollen (Initialisierungssequence).

```
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
```

Initialisierungsschritt	Bedeutung
eIEC870_ISTEP_UNUSED	Der Initialisierungsschritt wird nicht benutzt.
eIEC870_ISTEP_CLOCK	Der Server soll ein Uhrzeitsynchronisationsbefehl senden.
eIEC870_ISTEP_TEST	Der Server soll ein Testbefehl senden.
eIEC870_ISTEP_GENRO	Der Server soll ein Generalabfragebefehl senden.
eIEC870_ISTEP_CORO	Der Server soll ein Zählerabfragebefehl senden.
eIEC870_ISTEP_COMMAND	Der Server soll alle Kommandos einmalig senden (Datanpunkte in Steuerungsrichtung: Einzelbefehle, Doppelbefehle, Sollwerte usw.).
eIEC870_ISTEP_DELAY	Der Server soll mit der Ausführung des nächsten Initialisierungsschrittes warten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC or CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.34 E_IEC870_5_101FBufferState

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-101/104 Unterstation v3.0.2 / IEC60870-5-104 Zentralstation v1.0.2 und höher.

Aktueller Offline-ASDU-Dateipuffer-Status.

```
TYPE E_IEC870_5_101QU :
(
    eIEC870_FBUFFER_IDLE := 0,
    eIEC870_FBUFFER_SAVING := 1,
    eIEC870_FBUFFER_LOADING := 2,
    eIEC870_FBUFFER_ERROR := 3
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1313	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.43 E_IEC870_5_101QU

Qualifier der Befehle.

```
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC or CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.50 T_HAODBTable

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Zentralstation v1.0.0 und höher.

Applikationsobjekt-Datenbankhandle (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

```
TYPE T_HAODBTable :
STRUCT
(*...*)
END_STRUCT
END_TYPE
```

Auf Variablen dieses strukturierten Typs wird nicht direkt, sondern nur mit Hilfe der zur Verfügung stehenden Funktionen oder Funktionsbausteine zugegriffen. Benutzen Sie die in der Tabelle aufgeführten Funktionen:

Function	Description
F_iecCreateTableHnd [▶ 293]	Initialisiert das Hash-Tabellenhandle
F_iecAddTableEntry [▶ 294]	Konfiguriert und fügt einen einen neuen Hash-Tabelleneintrag
F_iecRemoveTableEntry [▶ 299]	Entfernt einen Hash-Tabelleneintrag
F_iecLookupTableEntry [▶ 298]	Prüft ob ein bestimmter Hash-Tabelleneintrag existiert
F_iecGetPosOfTableEntry [▶ 296]	Ermittelt die lineare Position eines Hash-Tabelleneientrags

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.51 T_CP16Time2a

Binäres zweibyte Zeitformat.

```
TYPE T_CP16Time2a :
STRUCT
    Milliseconds    : WORD; (* 0..59.999ms = 60sec = 1min *)
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.52 T_CP24Time2a

Binäres dreibyte Zeitformat

```
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.53 T_CP32Time2a

Binäres vierbyte Zeitformat.

```
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
```

Sommerzeit (SU): 1 = Sommerzeit, 0 = Normalzeit;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1307	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

5.3.54 T_CP56Time2a

Binäres 7 byte Zeitformat.

```
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 = 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) *)
    DOWDay          : BYTE; (* Bits 5..7 = Day of week (1..7, not used 0 !!!), Bits 0..4 = Day of mo
nth (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
```

Wochentag (DOW): 1 = Montag, 7 = Sonntag, 0 = wird nicht benutzt;

Sommerzeit (SU): 1 = Sommerzeit, 0 = Normalzeit;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

5.3.55 T_CP24IOA

Ab der Produktversion: TwinCAT PLC Library: IEC60870-5-104 Unterstation v3.0.0 / IEC60870-5-104 Zentralstation v1.0.0 und höher.

Strukturierte TwinCAT Objektadresse.

```
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 SPS-Prozessdatenbereich. Dieser Parameter [▶ 329] legt fest wie die TwinCAT SPS und IEC-Applikationsobjekt Prozessdaten gemappt werden sollen.

byteOffs: TwinCAT SPS-Prozessdaten Byte-Offset. Wertebereich: 0..65535.

bitOffs: TwinCAT SPS-Prozessdaten Bit-Offset. Wertebereich: 0..7.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86, ARM)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)

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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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> = Ausführen (execute);
- <1> = Anwählen (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 Konstanten

5.4.1 Group Konfigurationsflags

Mit den Group-Flags werden Applikationsobjekte bestimmten Gruppen zugeordnet. Die Group-Flags für Stationsabfrage und Zählwertübermittlung sind auf folgende Weise definiert worden:

- Stationsabfrage: Alle Applikationsobjekte, die der Gruppe 1 bis 16 zugeordnet wurden, gehören automatisch auch zur globalen Gruppe: IEC870_GRP_INROGEN;
- Zählwertübertragung: Alle Applikationsobjekte, die der Gruppe 1 bis 4 zugeordnet wurden, gehören automatisch auch zur globalen Gruppe: IEC870_GRP_REQCOGEN;

Sie können aber die automatische Zugehörigkeit der Applikationsobjekte zur globalen Gruppe verhindern in dem Sie die entsprechenden Bits während der Konfiguration ausmaskieren.

Beispiele:

IEC870_GRP_INRO3 AND NOT IEC870_GRP_INROGEN

oder

IEC870_GRP_REQCO1 AND NOT IEC870_GRP_REQCOGEN

Constant	Value	Description
Stationsabfrage		
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
Zählwertübertragung		
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

Constant	Value	Description
IEC870_GRP_LOCFREEZE	16#00400000	Enable cyclic local freeze of counter value
IEC870_GRP_LOCRESET	16#00800000	Enable cyclic local reset of counter value
Andere		
IEC870_GRP_IV_OFF	16#01000000	Don't set invalid quality bit
IEC870_GRP_REVERSE	16#02000000	Object used in reverse direction (reserved)
IEC870_GRP_SELECTCMD	16#04000000	Force select/execute command 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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

6 TcIEC870_5_101Master: IEC 60870-5-101 Zentralstation (master)

Mit den SPS-Funktionen und Funktionsbausteinen können Zentralstationen (Master) nach der IEC60870-5-101 Norm in der TwinCAT SPS realisiert werden.

Die SPS-Bibliothek verfügt über zwei Software-Schnittstellen. Die Endapplikation setzt auf einer dieser Schnittstellen auf. Die Wahl der Schnittstelle hängt von den Anforderungen an die Endapplikation ab. Im Folgenden werden die Eigenschaften beider Schnittstellen kurz beschrieben.

"High level"-Schnittstelle: IEC 60870-5-101 Zentralstation

Bei dieser Schnittstelle handelt es sich um eine sogenannte "Ein-Baustein-Lösung". Alle Funktionalitäten sind in einem SPS-Baustein gekapselt. Der Baustein implementiert die wichtigsten Dienste und Funktionen. Diese Implementierung ist für über 90% der Anwendungen ausreichend.

Pro: Sehr kleiner SPS-Programmieraufwand um eine Applikation zu erstellen; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw. sind bereits in dem Baustein implementiert und werden automatisch ausgeführt; Das Mapping der IEC<->SPS Prozessdaten und das der Datenpunkte wird über Funktionsaufrufe konfiguriert; Der SPS-Programmierer muss nicht sehr gut mit der Protokollnorm vertraut sein;

Contra: Die SPS-Applikation hat nur einen geringen Einfluss auf die Protokollausführung; Kein Einfluss auf die Ausführung der Dienste, diese werden intern automatisch ausgeführt; Zeitstempel werden von dem Baustein automatisch generiert und können nicht verändert (von extern übergeben) werden; Es ist z.B. nur die direkte Befehlsausführung möglich; Schlechtere Performance bei vielen Datenpunkten.

Diese Schnittstelle empfiehlt sich wenn Sie:

- Mit der Protokollnorm nicht vertraut sind;
- Eine einfache Applikation mit wenigen Datenpunkten implementieren (<1000);
- Keine großen Performance-Anforderungen an die Applikation stellen;
- Keine besondere Befehlsausführung wie Select/Execute oder Daten + Zeitstempel von externen Geräten versenden;
- Keine Funktionalitäten benötigen, die laut Kompatibilitätsliste nicht unterstützt werden;

"Low level"-Schnittstelle: IEC 60870-5-101 Serial Link Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können verändert werden (z.B. ein eigener Zeitstempel, Select/Execute oder eine besondere Befehlsausführung usw.); Weil nur die benötigten Dienste implementiert werden kann eine hohe Performance erreicht werden; Hohe Performance bei vielen Datenpunkten;

Contra: Größerer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Diese Schnittstelle empfiehlt sich wenn Sie:

- Mit der Protokollnorm vertraut sind;
- Eine Protokollkonverter-Applikation implementieren;
- In der Applikation die verfügbaren Norm-Funktionalitäten fast vollständig implementieren;

- Besondere Funktionalitäten verwenden, wie z.B. das Weiterleiten der Zeitstempel von einem Modbus-Gerät oder die Kontrolle über die Befehlsausführung erlangen;
- Funktionalitäten benötigen, die laut Kompatibilitätsliste nicht unterstützt werden;
- Viele Datenpunkte haben (>1000) und eine hohe Performance benötigen;

Kompatibilitätsliste

für TwinCAT SPS Bibliothek: IEC 60870-5-101 Zentralstation (bezieht sich auf die "High level"-Schnittstelle). Hier können Sie die https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11763692683.zip

Systemvoraussetzungen

Programmierumgebung:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT Installation Level: TwinCAT PLC oder höher;
- TwinCAT System Version 2.10.0 Build >= 1328 oder höher;

Zielplattform:

- Industrie PC oder Embedded PC/CX (x86, ARM);
- Betriebssystem:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86, ARM) (image v2.16 oder höher);
- TwinCAT SPS-Laufzeitsystem Version 2.10.0 oder höher;
- Serielle COM Schnittstelle oder KL6xxx- oder EL6xxx-Klemmen (**EL6xxx Unterstützung ab der Produktversion 1.0.3 und höher**);

Produktkomponenten

- **TcIEC870_5_101Master.Lib** (implementiert die Beckhoff IEC60870-5-101 Zentralstation). Diese Bibliothek muss in dem SPS-Projekt eingebunden werden. Alle anderen Bibliotheken werden automatisch hinzugefügt.
- TcIEC870_5_101Link.Lib (Basisbibliothek, implementiert Übertragungsprozeduren für den Transport der ASDUs über die seriellen Schnittstellen des PCs und die Beckhoff KL6xxx-/EL6xxx-Klemmen);
- TcIEC870_5_101.Lib (implementiert die Verbindungsfunktionen und gemeinsame Datentypen);
- COMLibV2.Lib (implementiert die Funktionen für die serielle COM- oder KL6xxx-/EL6xxx-Kommunikation);

Installation auf Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

Die SPS-Bibliotheken werden während der Installation auf der Programmierumgebung in den Ordner .. \TwinCAT\Plc\Lib kopiert:

Installation auf Windows CE

Auf der CE Plattform werden keine zusätzlichen Komponenten installiert.

Dies ist nur eine kurze Produktinformation (kein vollständiges Handbuch). Bitte installieren Sie sich die vollständige Ausgabe des Beckhoff Information System.

Sie finden es

- auf sämtlichen Beckhoff-Produkt-DVDs
- auf unserem Web-Server http://www.beckhoff.com unter Download.

Beispiele

Beispiele befinden sich in der Dokumentation der SPS-Bibliotheken (im Beckhoff Information System enthalten).

Link zu "High level" Beispiel-Übersichtsseite: IEC 60870-5-101 Zentralstation [▶ 368];

Link zu "Low level" Beispiel-Übersichtsseite: IEC 60870-5-101 Serial Link Interface [▶ 413];

Weiterführende Dokumentation

- Dokumentation zur TwinCAT PLC Library ("Low level"-Schnittstelle): IEC 60870-5-101 Serial Link Interface [▶ 396];
- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268];
- Dokumentation zur TwinCAT PLC Library: Serielle Kommunikation;
- 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 Einführung (Tutorial)

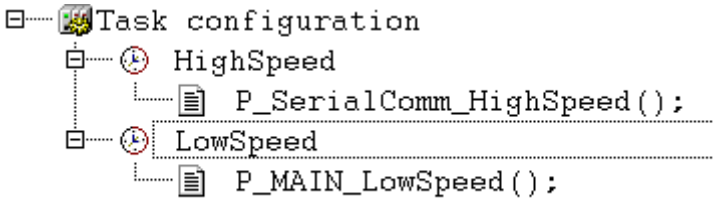
Die Einführung ist eine Anleitung wie Sie in der TwinCAT SPS eine IEC60870-5-101 Zentralstation (master) implementieren und konfigurieren können.

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

6.1.1 SPS-Projekt anlegen, SPS-Bibliotheken einbinden

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

1. Starten Sie TwinCAT PLC Control.
2. Mit Datei -> Neu legen Sie ein neues SPS-Projekt an. Wählen Sie als Zielsystem PC or CX (x86, ARM).
3. Als nächstes wird automatisch ein neuer Programmbaustein MAIN angelegt. Wählen Sie als Sprache des Bausteins ST (Strukturierter Text). Bestätigen Sie dies. Nennen Sie den MAIN Programmbaustein in *P_MAIN_LowSpeed* um.
4. Fügen Sie ein weiteres Programmbaustein hinzu und nennen diesen *P_SerialComm_HighSpeed*.
5. Konfigurieren Sie in der Taskkonfiguration 2 Tasks, eine schnelle (T#1ms) und eine langsame (T#10ms). Ordnen Sie den Programmbaustein *P_MAIN_HighSpeed* der schnellen Task und den *P_MAIN_LowSpeed* der langsamen Task (siehe Bild).



6. Aus dem Menü wählen Sie Fenster -> Bibliotheksverwaltung und dann Einfügen -> Weitere Bibliothek...
7. Aus der Liste der TwinCAT Bibliotheken wählen Sie **TcIEC870_5_101Master.Lib** aus und bestätigen dies.

6.1.2 Die schnelle SPS-Task

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip.

Fügen Sie im Deklarationsteil folgenden SPS-Code hinzu:

```
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
```

und im Programmcode:

```
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_STANDARD,
SIZEOF( serial_in ), SIZEOF( KL6_in ) ),
                    hSerial     := hSerial );
```

Eine zu diesem Beispiel passende TwinCAT System Manager Konfiguration finden Sie auf der Beispiele-Übersichtsseite. DIe Mode-Variable kann dazu verwendet werden um zwischen zwei Kommunikationswegen umzuschalten.

Kommunikation über die standard PC COMx-Schnittstellen

- In diesem Fall wird der Mode-Parameter auf den Wert: **SERIALLINEMODE_PC_COM_PORT** gesetzt.
- Im TwinCAT System Manager werden die *serial_in*- und *serial_out*- Variablen mit den entsprechenden IO-Variablen der seriellen Schnittstelle verknüpft.
- Die Schnittstelle wird und muss im TwinCAT System Manager konfiguriert werden (Baudrate, Parity usw.). Andere Kommunikationsparameter an dem FB_IEC870_SerialLineCtrl-Funktionsbaustein sind in diesem Mode irrelevant.

Kommunikation über die seriellen Beckhoff Busklemmen KL6xxx

- In diesem Fall wird der Mode-Parameter auf den Wert: **SERIALLINEMODE_KL6_5B_STANDARD** gesetzt.

- Im TwinCAT System Manager werden die *KL6_in* - und *KL6_out* - Variablen mit den entsprechenden I/O-Variablen der seriellen Klemme KL6xxx verknüpft.
- Die Schnittstelle wird in der TwinCAT SPS durch die Instanz des FB_IEC870_SerialLineCtrl-Funktionsbausteins konfiguriert. Die Kommunikationsparameter wie Baudrate, Parity usw. sind an diesem Baustein einzustellen.

Kommunikation über die seriellen Beckhoff Busklemmen 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 diesem Fall wird der Mode-Parameter auf den Wert: **SERIALLINEMODE_EL6_22B** gesetzt.
- Im TwinCAT System Manager werden die *EL6_in* - und *EL6_out* - Variablen mit den entsprechenden I/O-Variablen der seriellen Klemme EL6xxx verknüpft.
- Die Schnittstelle wird und muss im TwinCAT System Manager konfiguriert werden (Baudrate, Parity usw.). Andere Kommunikationsparameter an dem FB_IEC870_SerialLineCtrl-Funktionsbaustein sind in diesem Mode irrelevant.

6.1.3 Applikationsobjekt-Datenbank der Zentralstation definieren und konfigurieren

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Applikationsobjekte = Single Points, Double Points, Measured Values, Short Floating Point Values usw.

In diesem Beispiel wurden die Befehle so konfiguriert, dass die Prozessdaten in Steuerungsrichtung (Befehle) im gleichen Speicherbereich aber auf einem anderen Byte-/Bit-Offset wie die Daten der Information in Überwachungsrichtung liegen. Sie können aber auch die Befehle auf den gleichen Byte- Bit-Offset wie die Information in Überwachungsrichtung legen.

Beispiel:
C_SC_NA_1 mit IOA = 10 auf den gleichen Byte- und Bit-Offset wie M_SP_NA_1 mit IOA = 100 (beide Byte-Offset = 100 und Bit-Offset = 0). Bei einer Wertänderung von M_SP_NA_1 wird ein neues C_SC_NA_1-Kommando ausgelöst.

Als Beispiel konfigurieren wir in dem Einführungsprojekt folgende Applikationsobjekte:

ASDU identifier	Objekt-adresse IOA	Group-Konfi-gurationspara-meter	Basiszeit-multiplika-tor	SPS-Pro-zessda-tenbereich	Offset Byte	Offset Bit	Prozessdaten-breite in der TwinCAT SPS
M_SP_NA_1	100	Generalabfrage	0	Merker	100	0	1 Bit
M_SP_TA_1	101	Generalabfrage	0	Merker	100	1	1 Bit
M_SP_TB_1	102	Generalabfrage	0	Merker	100	2	1 Bit
M_DP_NA_1	200	Generalabfrage	0	Merker	200	0	2 Bits
M_DP_TA_1	201	Generalabfrage	0	Merker	200	2	2 Bits
M_DP_TB_1	202	Generalabfrage	0	Merker	200	4	2 Bits
M_ST_NA_1	300	Generalabfrage	0	Merker	300	0	1 Byte
M_ST_TA_1	301	Generalabfrage	0	Merker	301	0	1 Byte
M_ST_TB_1	302	Generalabfrage	0	Merker	302	0	1 Byte
M_BO_NA_1	400	Generalabfrage	0	Merker	400	0	4 Byte

ASDU identifier	Objekt-adresse IOA	Group-Konfi-gurationspara-meter	Basiszeit-multiplika-tor	SPS-Pro-zessda-tenbereich	Offset Byte	Offset Bit	Prozessdaten-breite in der TwinCAT SPS
M_BO_TA_1	401	Generalabfrage	0	Merker	404	0	4 Byte
M_BO_TB_1	402	Generalabfrage	0	Merker	408	0	4 Byte
M_ME_NA_1	500	Generalabfrage	0	Merker	500	0	2 Byte
M_ME_TA_1	501	Generalabfrage	0	Merker	502	0	2 Byte
M_ME_TD_1	502	Generalabfrage	0	Merker	504	0	2 Byte
M_ME_NB_1	600	Generalabfrage	0	Merker	600	0	2 Byte
M_ME_TB_1	601	Generalabfrage	0	Merker	602	0	2 Byte
M_ME_TE_1	602	Generalabfrage	0	Merker	604	0	2 Byte
M_ME_NC_1	700	Generalabfrage	0	Merker	700	0	4 Byte
M_ME_TC_1	701	Generalabfrage	0	Merker	704	0	4 Byte
M_ME_TF_1	702	Generalabfrage	0	Merker	708	0	4 Byte
M_IT_NA_1	800	Generalzählera bfrage und Mode A (lokal Umspeichern mit Spontanübertra gung alle 15s)	0	Merker	800	0	4 Byte
M_IT_TA_1	801	Generalzählera bfrage und Mode A (lokal Umspeichern mit Spontanübertra gung alle 15s)	0	Merker	804	0	4 Byte
M_IT_TB_1	802	Generalzählera bfrage und Mode A (lokal Umspeichern mit Spontanübertra gung alle 15s)	0	Merker	808	0	4 Byte
Commands							
C_SC_NA_1	10	-	0	Merker	2100	0	1 Bit
C_SC_NA_1	11	-	0	Merker	2100	1	1 Bit
C_SC_TA_1	12	-	0	Merker	2100	2	1 Bit
C_DC_NA_1	20	-	0	Merker	2200	0	2 Bit
C_DC_NA_1	21	-	0	Merker	2200	2	2 Bit
C_DC_TA_1	22	-	0	Merker	2200	4	2 Bit
C_BO_NA_1	40	-	0	Merker	2400	0	4 Byte
C_BO_NA_1	41	-	0	Merker	2404	0	4 Byte
C_BO_TA_1	42	-	0	Merker	2408	0	4 Byte
C_SE_NA_1	50	-	0	Merker	2500	0	2 Byte
C_SE_NA_1	51	-	0	Merker	2502	0	2 Byte
C_SE_TA_1	52	-	0	Merker	2504	0	2 Byte
C_SE_NB_1	60	-	0	Merker	2600	0	2 Byte
C_SE_NB_1	61	-	0	Merker	2602	0	2 Byte

ASDU identifier	Objekt-adresse IOA	Group-Konfi-gurationspara-meter	Basiszeit-multiplika-tor	SPS-Pro-zessda-tenbereich	Offset Byte	Offset Bit	Prozessdaten-breite in der TwinCAT SPS
C_SE_TB_1	62	-	0	Merker	2604	0	2 Byte
C_SE_NC_1	70	-	0	Merker	2700	0	4 Byte
C_SE_NC_1	71	-	0	Merker	2704	0	4 Byte
C_SE_TC_1	72	-	0	Merker	2708	0	4 Byte

Datenbankvariable deklarieren

Die Applikationsobjekt-Datenbank ist eine Array-Variable des Typen ST_IEC870_5_101AODBEntry [▶ 312]. Jedes Array-Element entspricht einem Applikationsobjekt. Die Array-Elemente werden aber nicht direkt, sondern nur mit Hilfe der speziell dafür zur Verfügung gestellten Funktionen und ein Datenbank-Handle (Tabellen-Handle) manipuliert. Das Datenbank-Handle muss vor der Benutzung durch einen einmaligen F_iecCreateTableHnd [▶ 293]-Funktionsaufruf initialisiert werden. Dabei werden auch die Array-Elemente miteinander als Hash-Tabelle verknüpft. Bei einer größeren Anzahl der Datenpunkte ermöglicht die Hash-Tabelle einen schnelleren Zugriff auf einen einzelnen Datenpunkt.

Die maximale Anzahl der Applikationsobjekte ist frei wählbar und nur durch den verfügbaren Speicher begrenzt. Sie müssen sich auf eine konstante maximale Anzahl während der SPS-Programmierung festlegen. Zur Laufzeit kann die maximale Anzahl der Applikationsobjekte nicht mehr verändert werden. In unserem Beispiel werden 50 Applikationsobjekte deklariert. Diese Anzahl reicht für die meisten Anwendungen aus. Beachten Sie, dass sehr viele Applikationsobjekte auch entsprechend viel Speicher und Laufzeit benötigen.

Definieren Sie folgende Variablen in P_MAIN_LowSpeed:

```
PROGRAM P_MAIN_LowSpeed
VAR
     AODB         : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry
    hTable       : T_HAODBTable;
END_VAR
```

Applikationsobjekte konfigurieren

Die Konfiguration der gewünschten Applikationsobjekte wird zur Programmlaufzeit durchgeführt. Während der Konfiguration werden unter anderem der Objekt-Typ (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 usw.), die gemeinsame ASDU-Adresse, die Objekt-Adresse und weitere Objekt-Parameter festgelegt.

Nach der Initialisierung des Datenbank-Handles ist die Applikationsobjekt-Datenbank (Datenbank-Array) leer und muss mit den gewünschten Daten (Datenpunkten) gefüllt werden. Die konfiguration der Datenpunkte der Zentralstation muss der Konfiguration der Datenpunkte in der Unterstation entsprechen! D.h. in der Zentralstation müssen Datenpunkte vom gleichen Typ, gleicher gemeinsamen ASDU-Adresse und mit der gleichen Informationsobjekt-Adresse wie in der Unterstation konfiguriert werden. Andere Parameter wie z.B. das Mapping-Bereich, Byte-, Bit-Offset können beliebig konfiguriert werden.

Es stehen folgende Funktionen zur Manipulation der Applikationsdatenbank zur Verfügung:

Function	Beschreibung
F_iecCreateTableHnd [▶ 293]	Initialisiert das Hash-Tabellenhandle
F_iecAddTableEntry [▶ 294]	Konfiguriert und fügt einen einen neuen Hash-Tabelleneintrag
F_iecRemoveTableEntry [▶ 299]	Entfernt einen Hash-Tabelleneintrag
F_iecLookupTableEntry [▶ 298]	Prüft ob ein bestimmter Hash-Tabelleneintrag existiert
F_iecGetPosOfTableEntry [▶ 296]	Ermittelt die lineare Position eines Hash-Tabelleneientrags

Das Datenbank-Handle muss an die Funktion per VAR_IN_OUT übergeben werden. Im Regelfall wird die Konfiguration beim SPS-Programmstart einmalig in einer Init-Routine durchgeführt.

Um die Applikationsobjekte beim Programmstart zu konfigurieren, wird in P_MAIN_LowSpeed folgender SPS-Code hinzugefügt:

```
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 der SPS- und IEC-Prozessdaten

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Die TwinCAT SPS-Prozessdaten werden zur Programmlaufzeit zyklisch in die IEC-Prozessdaten (Applikationsobjekte) und umgekehrt gemappt (kopiert). Für das Mapping der IEC<->SPS Prozessdaten können bis zu 4 Prozessdatenbereiche (IO-Eingänge, IO-Ausgänge, Merkerbereich, Datenbereich) als Puffervariablen im SPS-Programm deklariert werden. Die Bytegröße der Puffer ist frei wählbar und kann für jeden Bereich unterschiedlich gewählt werden. Unbenutzte Bereiche müssen nicht unbedingt deklariert werden.

In unserem Einführungsbeispiel deklarieren wir 4 SPS-Prozessdatenbereiche mit jeweils 3000 Bytes:

```
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
```

Die Zuordnung, wie die Prozessdaten zur Laufzeit gemappt werden sollen, wird während der Konfiguration der Applikationsobjekte mit der F_iecAddTableEntry [▶ 294]-Funktion festgelegt.

Siehe auch in: Applikationsobjekte definieren und konfigurieren [▶ 353].

Die Puffervariablen wurden nun als Byte-Arrays deklariert. Um auf die gewünschten Daten besser zugreifen zu können definieren wir die einzelnen Variablen ein zweites Mal und legen diese auf die entsprechenden Byte/Bit-Offsetadressen. Bei einer Änderung im Byte-Array wird die entsprechende einzelne Variable gleichzeitig geändert und umgekehrt. Dies ist aber nicht zwingend notwendig. Sie können direkt auf die Bytes/Bits der Byte-Array-Puffervariablen zugreifen.

```
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 der IEC<->SPS Prozessdaten in der Zentralstation

Prozessdaten in Überwachungsrichtung (Slave->Master information)

Beispiel 1

Single point information (M_SP_NA_1) mit der IOA = 100, SPS Merkerbereich, Byteoffset = 100, Bitoffset = 0.

Unterstation -> ... -> Zentralstation FB -> memory[100].0 == msgSingle_0

Beispiel 2

Measured value, short floating point value (M_ME_NC_1) mit der IOA = 700, SPS Merkerbereich, Byteoffset = 700, Bitoffset = 0 (bedeutungslos).

Unterstation -> ... -> Zentralstation FB -> memory[700..703] == msgFloating_0

Prozessdaten in Steuerungsrichtung (Master->Slave commands)

Beispiel 1

Single command state (C_SC_NA_1) mit der IOA = 10, SPS Merkerbereich, Byteoffset = 2100, Bitoffset = 0.

cmdSingle_0 == memory[2100].0 -> Zentralstation FB -> ... -> Unterstation

Beispiel 2

Set point, short floating point value (C_SE_NC_1) mit der IOA = 70, SPS Merkerbereich, Byteoffset = 2700, Bitoffset = 0 (bedeutungslos).

cmdFloating_0 == memory[2700..2703] -> Zentralstation FB -> ... -> Unterstation

6.1.5 Instanz der IEC60870-5-101 Zentralstation deklarieren und aufrufen

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Die gesamte Funktionalität einer Zentralstation ist im Funktionsbaustein FB_IEC870_5_101Master gekapselt. Mit einer Instanz kann eine Verbindung zur Unterstation aufgebaut werden. Für eine weitere Verbindung deklarieren Sie eine weitere Instanz des Funktionsbausteins.

Fügen Sie im Deklarationsteil von P_MAIN_LowSpeed folgenden SPS-Code ein:

```
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;

END_VAR
```

und im Programmteil wird die Instanz aufgerufen:

```
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-Protokollparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Über die IEC60870-5-101-Protokollparameter kann das Verhalten der Zentralstation an die Anforderungen des Unterstation angepasst werden. Die meisten Parameter sind mit Defaultwerten vorbelegt, so dass diese nicht verändert werden müssen.

Im unserem Beispiel konfigurieren wir die Verbindungsadresse, die Oktetlänge der Verbidnungsadresse und die Zykluszeit in der die Class 1- und Class 2-Daten gepollt werden sollen.

```
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
```

Die Dokumentation aller Übertragungsprotokoll-Parameter finden Sie hier: ST_IEC870_5_101PotocolParams [▶ 406].

6.1.7 Systemparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Über die Systemparameter wird z.B. die gemeinsame ASDU-Adresse und die Anwenderfunktionen der Zentralstation konfiguriert.

In unserer Einführung konfigurieren wir folgende Systemparameter:

- Die gemeinsame ASDU-Adresse wird auf 7 gesetzt. (*asduAddr*);
- Die Oktetlänge der Übertragungsursache wird auf 2 gesetzt (1 Oktet für COT + 1 Oktet für Quelladresse) (*eCOTSize*);
- Die Oktetlänge der gemeinsamen ASDU Adresse wird auf 2 gesetzt (*eAsduAddrSize*);
- Die Oktetlänge der Informationsobjektadresse wird auf 3 gesetzt (*eObjAddrSize*);
- Das loggen der Debugmeldungen im Application-Log wird aktiviert (*dbgMode*). Es werden Änderungen im Gerätestatus und Fehlermeldungen der Verbindungsschicht gelogt;

Fügen Sie folgenden SPS-Code in Ihr SPS-Projekt ein:

```
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_DEBUGM
ODE_ASDU OR IEC870_DEBUGMODE_LINKLAYER *)

...
ELSE
    client( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

Die Dokumentation aller Systemparameter finden Sie hier: ST_IEC870_5_101SystemParams [▶ 315].

6.1.8 Initialisierungssequence

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/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 Stationsabfrage

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

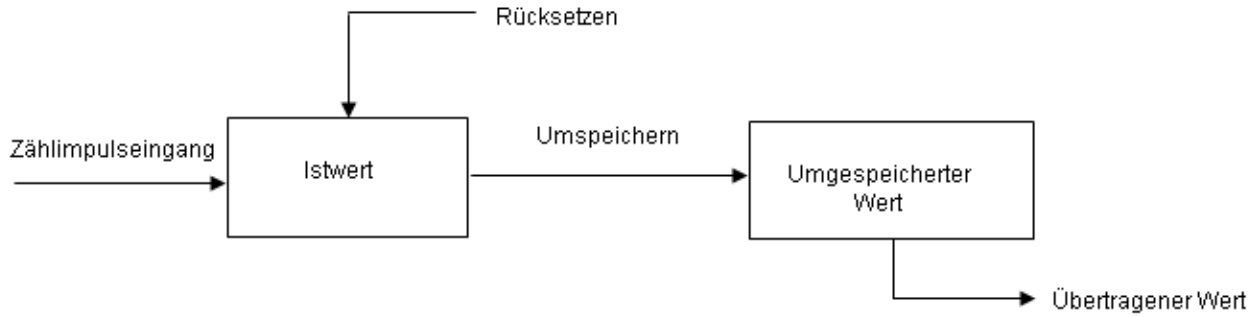
Der Stationsabfragebefehl wird von der Zentralstation eingeleitet. Im Kennungsfeld des Befehls ist auch die Gruppe (1 bis 16 oder allgemein) festgelegt. Die Unterstation überträgt die zu dieser Gruppe dazugehörigen Applikationsobjekte mit der Übertragungsursache *<20>* bis *<36>* an die Zentralstation. Applikationsobjekte mit Zeitmarken werden ohne Zeitmarken übertragen.

```
client.acqPara.arrGenro[0].tPollCycle := T#60s;
client.acqPara.arrGenro[0].eQOI := eIEC870_QOI_INROGEN;
client.acqPara.arrGenro[0].bEnable := TRUE;
```

6.1.10 Zählwertübertragung (counter interrogation)

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Allgemeines Modell für die Zählwertübertragung:



Die Istwerte werden durch Zähler aufsummiert. Die Istwerte können durch einen Umspeicherbefehl, der entweder von der Zentralstation empfangen oder örtlich (lokal in der Unterstation) erzeugt wird, periodisch in umgespeicherte Werte umgespeichert (kopiert) werden. Nach dem Umspeichern wird der erfasste Wert entweder auf Null zurückgesetzt (Erfassen von Inkrementalwerten) oder der Zähler fährt mit seinem Betrieb fort (Erfassen von Zählerständen).

Applikationsobjekte mit Zählwerten werden Gruppen zugeordnet. Die Gruppen werden einzeln umgespeichert (frozen), zurückgesetzt (reset) oder übertragen. Die Zentralstation sendet Zählwertabfragebefehle an die Unterstation. In einem Kennungsfeld des Befehls (QCC) wird die durchzuführende Aktion (FRZ) und Gruppe (RQT) festgelegt.

Die Zuordnung der Applikationsobjekte zu den einzelnen Gruppen (1 bis 4 oder allgemein) wird während der Konfiguration durch den Group-Flagparameter festgelegt. Es gibt vier Betriebsarten für die Erfassung von Zählerständen und Inkrementalwerten. Zu jeder Betriebsart sind einige Hinweise zur Konfiguration der Systemparameter oder der Applikationsobjekte aufgeführt.

Betriebsart A: Örtlich Umspeichern mit Spontanübertragung

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden spontan übertragen, nachdem die Funktion Umspeichern oder Umspeichern mit Rücksetzen durchgeführt wurde. Die Zentralstation gibt in dieser Betriebsart keine Zählwertabfragebefehle aus.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

Betriebsart B: Örtliches Umspeichern mit Zählerabfrage

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden durch Zählwertabfragebefehle von der Zentralstation abgefragt. In diesem Fall darf die Zentralstation im Befehlskennungsfeld das Umspeichern oder Umspeichern mit Rücksetzen nicht benutzen (FRZ=0). Die Zählwerte werden allgemein oder in Gruppen (groups) 1 bis 4 abgefragt.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

Betriebsart C: Zentralstation leitet das Umspeichern, Umspeichern mit Rücksetzen oder Rücksetzen ein

Ein Zählwertabfragebefehl wird periodisch von der Zentralstation an die Unterstation ausgegeben, um das Umspeichern oder (und) Rücksetzen zu steuern. Dieser Befehl hat aber noch keine Übertragung der Zählwerte zur Folge. Erst ein nachfolgender Zählwertabfragebefehl wird von der Zentralstation gesendet, um die umgespeicherten Zählwerte einzusammeln. Ähnlich, wie bei der Betriebsart B.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

Betriebsart D: Zentralstation leitet das Umspeichern und (oder) Rücksetzen ein und die umgespeicherten Werte werden spontan übertragen

Diese Betriebsart ist eine Kombination des Zählwertbefehls von der Zentralstation wie für Betriebsart C mit einer spontanen Übertragung der Zählwerte wie bei der Betriebsart A.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

6.1.11 Uhrzeitsynchronisation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

In Vorbereitung....

6.1.12 Test der Kommunikation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Durch das Setzen der *bExecuteCmd*-Variable auf TRUE wird eine einfache Simulation der Befehle in Steuerungsrichtung aktiviert und mit FALSE deaktiviert. Bei einer aktiven Verbindung wird in unsem Beispiel ein Einzeln-Befehl (C_SC_NA_1, IOA = 10) zyklisch alle 10 Sekunden zur Unterstation übertragen.

```
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 Übertragungs- und Kommunikationsfehler

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip

Die Stationsfehlermeldungen werden in einem FIFO abgelegt. Es können bis zu 10 Fehlermeldungen zwischengespeichert werden. Bei fatalen Kommunikationsfehlern (z.B. Fehler der Verbindungsschicht, die Checksumme des Frames passt nicht) wird die Verbindung unterbrochen und muss neu aufgebaut werden. Fehler in der Applikationsschicht (z.B. der ASDU-Sendepuffer ist wegen zu vieler Frames übergelaufen)

werden nur geloggt und führen nicht zum Verbindungsabbruch. Es immer noch möglich auch bei diesen Fehlern die Verbindung aus der Applikation zu unterbrechen. Neben dem Fehler-Code wird auch die Fehlerquelle in der Fehlermeldung abgelegt. Dieses erleichtert die Lokalisierung des Fehlers.

Beispiel

Die anfallenden Fehlermeldungen einer IEC 60870-5-101 Zentralstation können durch folgenden Aufruf ausgelesen werden:

```
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 SPS-API

Einfache Projekte mit kompletten Sourcen finden Sie hier: IEC60870-5-101 Zentralstation [▶ 368].

Kompatibilitätsliste finden Sie hier: Interoperability check list

Übersicht der Fehlercodes finden Sie hier: Fehlercodes

Eine ausführliche Anleitung zur Implementierung der Zentralstation in der SPS finden Sie hier: TUTORIAL [▶ 351]

Kurzanleitung

Applikationsobjekt-Datenbank

Die Applikationsobjekt-Datenbank der Zentralstation muss mit der Funktion: F_iecCreateTableHnd [▶ 293] als Hash-Tabelle konfiguriert werden. Die einzelnen Arrayelemente werden dabei als Hash-Tabelle untereinander verlinkt. Dies ermöglicht u. a. einen schnelleren Zugriff auf die einzelnen Datenpunkte, bringt aber auch einige Nachteile mit sich, die beachtet werden müssen:

- Die Größe der Applikationsdatenbank (Arraygröße) darf nicht zur Laufzeit (z.B. durch Online-Change) verändert werden. Die Zentralstation stoppt sofort die Ausführung und meldet einen Fehler. Der Grund: Die Verlinkung der Hash-Tabelle passt nicht mehr. Bei Programmänderungen laden Sie am besten das komplette Projekt ins Laufzeitsystem.
- Auf die einzelnen Array-Elemente darf nicht per Index sondern nur mit Hilfe der speziellen Funktionen zugegriffen werden (z.B.: F_iecAddTableEntry [▶ 294] usw.).
- Bei einem indizierten Zugriff auf die Tabellenelemente dürfen die internen Konfigurationsparameter nicht beschrieben oder verändert werden. Bei einer Änderung des Typs, der ASDU-Adresse oder der Objektadresse kann der Datenpunkt nicht mehr gefunden werden. Ein Datenpunkt, der umkonfiguriert werden soll wird aus der Tabelle durch den Funktionsaufruf F_iecRemoveTableEntry [▶ 299] zuerst entfernt. Danach kann der neue Datenpunkt hinzugefügt werden.

Bei einer Implementierung als lineare Tabelle müsste die Zentralstation bei jeder empfangenen ASDU (Dateneinheit) das komplette Array nach dem passenden Element durchsuchen. Dies würde bei vielen Datenpunkten sehr lange Ausführungszeiten generieren.

Protokollparameter

Die meisten Protokollparameter sind bereits mit Defaultwerten vorbelegt und müssen nicht explizit gesetzt werden.

Systemparameter

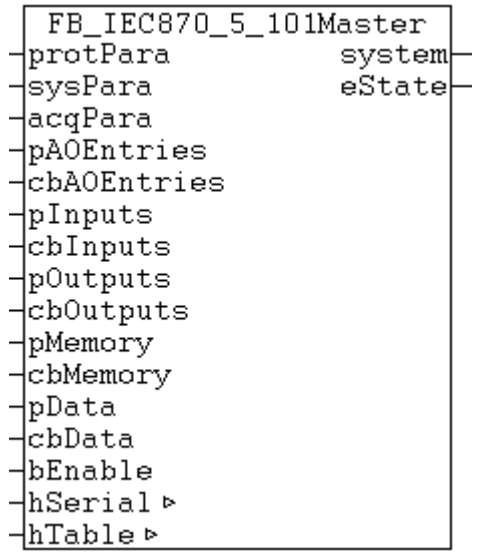
Die Systemparameter sind ebenfalls mit Defaultwerten vorbelegt. Während der Inbetriebnahme ist es nützlich die Debug-Ausgabe zu aktivieren (*dbgMode*) um mögliche Fehler lokalisieren zu können.

Parameter für die zyklische Datenerfassung

Folgende Parameter sind bereits mit Defaultwerten vorkonfiguriert:

- Initialisierungssequence (besteht aus einem Testbefehl, Uhrzeitsynchronisation, Stationsabfrage und Zählerabfrage);
- Zyklische Befehle:
  - Testbefehl alle 60s;
  - Uhrzeitsynchronisation alle 60s;
  - Stationsabfrage der Gruppe: Allgemein alle 60s;
  - Zählerabfrage der Gruppe: Allgemein alle 60s;

6.2.1 FB_IEC870_5_101Master



Mit einer Instanz des Funktionsbausteins FB_IEC870_5_101Master kann in der TwinCAT SPS eine IEC60870-5-101 Zentralstation (Master) implementiert werden. Pro Instanz des Funktionsbausteins wird eine Verbindung zum Slave aufgebaut. Im Normalfall wird der Datenaustausch automatisch gestartet, nachdem die Verbindung hergestellt wurde. Standardmäßig ist der Funktionsbaustein auch so konfiguriert.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial : T_HSERIALCTRL;
    hTable  : T_HAODBTable;
END_VAR
```

hTable: Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

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-Protokolparameter [▶ 406].

sysPara: Systemparameter [▶ 315].

acqPara: Parameter [▶ 319] für die zyklische Datenerfassung.

pAOEntries: Adresse der Applikationsobjekt-Datenbankvariablen [▶ 312].

cbAOEntries: Bytegröße der Applikationsobjekt-Datenbankvariablen.

pInputs: Adresse des SPS-Prozessdatenbereichs der Eingänge.

cbInputs: Bytegröße des SPS-Prozessdatenbereichs der Eingänge.

pOutputs: Adresse des SPS-Prozessdatenbereichs der Ausgänge.

cbOutputs: Bytegröße des SPS-Prozessdatenbereichs der Ausgänge.

pMamory: Adresse des SPS-Prozessdatenbereichs der Merker.

cbMamory: Bytegröße des SPS-Prozessdatenbereichs der Merker.

pData: Adresse des SPS-Datenbereichs.

cbData: Bytegröße des SPS-Datenbereichs.

bEnable : Aktiviert/Deaktiviert den Funktionsbaustein (Kommunikation und Verbindungen).

Die Adressen können mit dem ADR- und die Bytegrößen mit dem SIZEOF-Operator ermittelt werden.

VAR_OUTPUT

```
VAR_OUTPUT
    system  : ST_IEC870_5_101ExSystemInterface;
    eState  : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED;
END_VAR
```

system: System-Interface [▶ 367]. Diese Variable dient anderen IEC-Applikationsfunktionen als Kommunikationsschnittstelle zum IEC-Gerät (hier: Zentralstation).

- Membervariable *system.device* wird z.B. von der F_iecSetAOQuality [▶ 279]-Funktion als VAR_IN_OUT-Parameter erwartet.
- Membervariable *system.device.errors* ist ein Gerätefehler-Fifo. Die registrierten Fehler können von der SPS-Applikation ausgelesen und ausgewertet werden.

eState: Status [▶ 411] der Verbindung zum Slave.

Beispiel in ST: IEC60870-5-101 Zentralstation [▶ 368]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1328	PC oder CX (X86, ARM)	TcIEC870_5_101Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; COMLIbV2.Lib;

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; werden automatisch eingebunden)

Sehen Sie dazu auch

T_HSERIALCTRL [▶ 413]

6.2.2 F_GetVersionTcIEC870_5_101Master

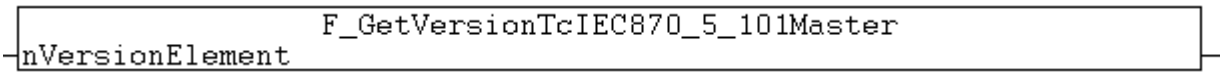


Abb. 5: F_GetVersionTcIEC870_5_101Master

Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_101Master: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1328	PC oder 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 werden automatisch eingebunden)

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: Kommunikationsschnittstelle [▶ 318] des IEC-Gerätes.

service: IEC-Gerätedienste;

hSOTable : Systemobjekt Datenbank Handle [▶ 342];

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1328	PC oder CX (x86, ARM)	TcIEC870_5_101Master.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; COMLibV2.Lib;

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		TcIEC870_5_101Link.Lib; TcIEC870_5_101.Lib werden automatisch eingebunden)

6.3 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE Image-Version usw.).
2. Vergleichen/überprüfen Sie die Kompatibilitätsliste der Leitstation mit der Kompatibilitätsliste der Unterstation.
3. Überprüfen Sie die IO-Konfiguration und das Mapping der SPS-Variablen in TwinCAT System Manager (Konfiguration der seriellen Schnittstellen, Baudrate, Parity, Stopbits usw.). Vergleichen Sie die Parameter mit Parametern in der Unterstation.
4. Überprüfen Sie ob der Funktionsbaustein einen Fehlercode/Fehlerquelle [▶ 363] ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes.
5. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 406] (Link-Adresse, Länge der Link-Adresse, FRAMELength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern in der Unterstation.
6. Überprüfen Sie die am Funktionsbaustein eingestellten Systemparameter [▶ 315] (ASDU-Adresse, Länge der ASDU-Adresse, Länge der Informationsobjekt-Adresse, Länge der Übertragungsursache COT, usw.). Vergleichen Sie die Systemparameter mit den Parametern in der Unterstation.
7. Überprüfen Sie die am Funktionsbaustein eingestellten Parameter für die zyklische Datenerfassung [▶ 319] (Akquisition) (Initialisierungssequenz, zyklische Generalabfrage, zyklische Zählerabfrage, zyklische Testkommandos, usw.).
8. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, Adresse des Informationsobjektes usw.).
9. Überprüfen Sie ob die Unterstation einen Fehlercode ausgibt.
10. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben.

6.4 Beispiele

In den Beispielprojekten sind folgende Stationsparameter für die Zentralstation eingestellt:

- 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**

Voraussetzungen

SPS Project	TwinCAT System Manager Konfiguration	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697520139.zip	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697520139.zip	Einfache TwinCAT IEC60870-5-101 Zentralstation. Ein **minimalistisches** SPS-Projekt mit einem Single-Point in

SPS Project	TwinCAT System Manager Konfiguration	Beschreibung
		Überwachungsrichtung (IOA = 100) und einem Single-Command in Steuerungsrichtung (IOA = 10).
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11697518731.zip	Einfache **IEC60870-5-101 Zentralstation**

7 TcIEC870_5_101Slave: IEC 60870-5-101 Unterstation (slave)

Mit den SPS-Funktionen und -Funktionsbausteinen können Unterstationen (Slaves) nach der IEC60870-5-101 Norm in TwinCAT SPS realisiert werden.

Die SPS-Bibliothek verfügt über zwei Software-Schnittstellen. Die Endapplikation setzt auf einer der Schnittstellen auf. Die Wahl der Schnittstelle hängt von den Anforderungen an die Endapplikation ab. Im Folgenden werden die Eigenschaften beider Schnittstellen kurz beschrieben.

"High level"-Schnittstelle: IEC 60870-5-101 Unterstation

Bei dieser Schnittstelle handelt es sich um eine sogenannte "Ein-Baustein-Lösung". Alle Funktionalitäten sind in einem SPS-Baustein gekapselt. Der Baustein implementiert die wichtigsten Dienste und Funktionen. Diese Implementierung ist für über 90% der Anwendungen ausreichend.

Pro: Sehr kleiner SPS-Programmieraufwand um eine laufende Applikation zu erhalten; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw sind bereits in dem Baustein implementiert und werden automatisch ausgeführt; Das Mapping der IEC<->SPS Prozessdaten und das der Datenpunkte wird über Funktionsaufrufe konfiguriert; Der SPS-Programmierer muss nicht sehr gut mit der Protokollnorm vertraut sein;

Contra: Die SPS-Applikation hat nur einen geringen Einfluss auf die Protokollausführung; Kein Einfluss auf die Ausführung der Dienste, diese werden intern automatisch ausgeführt; Zeitstempel werden von dem Baustein automatisch generiert und können nicht verändert (von extern übergeben) werden; Es ist z.B. nur die direkte Befehlsausführung möglich; Schlechtere Performance bei vielen Datenpunkten.

Diese Schnittstelle empfiehlt sich wenn Sie:

- Mit der Protokollnorm nicht vertraut sind;
- Eine einfache Applikation mit einer handvoll Datenpunkten implementieren möchten (<1000);
- Keine grossen Performace-Anforderungen an die Applikation stellen;
- Keine besondere Befehlsausführung wie Select/Execute oder Daten + Zeistempel von externen Geräten versenden möchten;
- Keine Funktionalitäten benötigen die laut Kompatibilitätsliste nicht unterstützt werden;

"Low level"-Schnittstelle: IEC 60870-5-101 Serial Link Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können manipuliert werden (z.B. ein eigener Zeitstempel, Select/Execute oder eine besondere Befehlsausführung usw); Weil nur die benötigten Dienste implementiert werden, kann eine hohe Performance erreicht werden; Hohe Performance bei vielen Datenpunkten;

Contra: Größerer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Diese Schnittstelle empfiehlt sich wenn Sie:

- Mit der Protokollnorm vertraut sind;
- Eine Protokollkonverter-Applikation implementieren möchten;
- In der Applikation die verfügbaren Norm-Funktionalitäten fast vollständig implementieren müssen;

- Besondere Funktionalitäten wie z.B. das weiterleiten der Zeitstempel von einem Modbusgerät oder die Kontrolle über die Befehlsausführung haben möchten;
- Funktionalitäten benötigen die laut Kompatibilitätsliste nicht unterstützt werden;
- Viele Datenpunte haben (>1000) und eine hohe Performace benötigen;

Kompatibilitätsliste

für TwinCAT SPS Bibliothek: IEC 60870-5-101 Unterstation (bezieht sich auf die "High level"-Schnittstelle). Hier können Sie die https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11763793547.zip

Systemvoraussetzungen

Programmierumgebung:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT Installation Level: TwinCAT PLC oder höher;
- TwinCAT System Version 2.9.0 Build >= 1030 (CX (ARM) Build >= 1301) oder höher;

Zielplattform:

- Industrie PC or Embedded PC/CX (x86, ARM);
- Betriebssystem:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86) (image v1.75 oder höher);
  - Windows CE (ARM) (image v2.13 oder höher);
- TwinCAT SPS-Laufzeitsystem Version 2.9.0 oder höher;
- Serielle COM Schnittstelle oder KL6xxx- oder EL6xxx-Klemmen (**EL6xxx Unterstützung ab der Produktversion 3.0.9 und höher**);

Produktkomponenten

- **TcIEC870_5_101Slave.Lib** (implementiert die Beckhoff IEC60870-5-101 Unterstation). Diese Bibliothek muss in dem SPS-Projekt eingebunden werden. Alle anderen Bibliotheken werden automatisch hinzugefügt.
- TcIEC870_5_101.Lib (implementiert die Verbindungsfunktionen und gemeinsame Datentypen);
- TcIEC870_5_101Link.Lib (implementiert Übertragungsprozeduren für den Transport der ASDUs über die seriellen Schnittstellen des PCs und die Beckhoff KL6xxx-/EL6xxx-Klemmen);
- COMLibV2.Lib (implementiert die Funktionen für die serielle COM- oder KL6xxx-/EL6xxx-Kommunikation);

Installation auf Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

Die SPS-Bibliotheken werden während der Installation auf der Programmierumgebung in den ..\TwinCAT\Plc\Lib-Ordner hineinkopiert.

Installation auf Windows CE

Auf der CE Plattform werden keine zusätzlichen Komponenten installiert.

Dies ist nur eine kurze Produktinformation (kein vollständiges Handbuch). Bitte installieren Sie sich die vollständige Ausgabe des Beckhoff Information System.

Sie finden es

- auf sämtlichen Beckhoff-Produkt-DVDs
- auf unserem Web-Server http://www.beckhoff.com unter Download.

Beispiele

Beispiele befinden sich in der Beckhoff Information System Dokumentation der SPS-Bibliotheken.

Link zu "High level" Beispiel-Übersichtsseite: IEC 60870-5-101 Unterstation [▶ 394];

Link zu "Low level" Beispiel-Übersichtsseite: IEC 60870-5-101 Serial Link Interface [▶ 413];

Weiterführende Dokumentation

- Dokumentation zur TwinCAT PLC Library ("Low level"-Schnittstelle): IEC 60870-5-101 Serial Link interface [▶ 396];
- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268];
- Dokumentation zur TwinCAT PLC Library: Serielle Kommunikation;
- 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 Einführung (Tutorial)

Die Einführung ist eine Anleitung wie Sie in der TwinCAT SPS eine IEC60870-5-101 Unterstation (slave) implementieren und konfigurieren können.

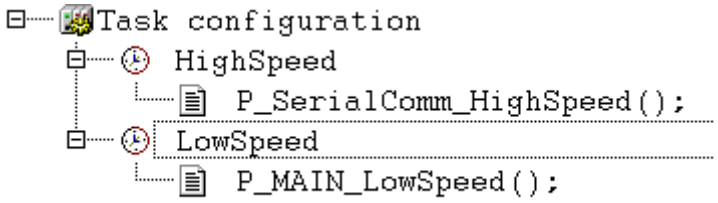
Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

7.1.1 SPS-Projekt anlegen, SPS-Bibliotheken einbinden

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

1. Starten Sie TwinCAT PLC Control.
2. Mit Datei -> Neu legen Sie ein neues SPS-Projekt an. Wählen Sie als Zielsystem PC or CX (x86, ARM).
3. Als nächstes wird automatisch ein neuer Programmbaustein MAIN angelegt. Wählen Sie als Sprache des Bausteins ST (Strukturierter Text). Bestätigen Sie dies. Nennen Sie den MAIN Programmbaustein in *P_MAIN_LowSpeed* um.
4. Fügen Sie ein weiteres Programmbaustein hinzu und nennen diesen *P_SerialComm_HighSpeed*.

5. Konfigurieren Sie in der Taskkonfiguration 2 Tasks, eine schnelle (T#1ms) und eine langsame (T#10ms). Ordnen Sie den Programmbaustein *P_MAIN_HighSpeed* der schnellen Task und den *P_MAIN_LowSpeed* der langsamen Task (siehe Bild).



1. Aus dem Menü wählen Sie Fenster -> Bibliotheksverwaltung und dann Einfügen -> Weitere Bibliothek...
2. Aus der Liste der TwinCAT Bibliotheken wählen Sie **TcIEC870_5_101Slave.Lib** aus und bestätigen dies.

7.1.2 Die schnelle SPS-Task

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip.

Fügen Sie im Deklarationsteil folgenden SPS-Code hinzu:

```
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
```

und im Programmcode:

```
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_STANDARD,
SIZEOF( serial_in ), SIZEOF( KL6_in ) ),
                    hSerial     := hSerial );
```

Eine zu diesem Beispiel passende TwinCAT System Manager Konfiguration finden Sie auf der Beispiele-Übersichtsseite. DIe Mode-Variable kann dazu verwendet werden um zwischen zwei Kommunikationswegen umzuschalten.

Kommunikation über die standard PC COMx-Schnittstellen

- In diesem Fall wird der Mode-Parameter auf den Wert: **SERIALLINEMODE_PC_COM_PORT** gesetzt.
- Im TwinCAT System Manager werden die *serial_in*- und *serial_out*- Variablen mit den entsprechenden IO-Variablen der seriellen Schnittstelle verknüpft.

- Die Schnittstelle wird und muss im TwinCAT System Manager konfiguriert werden (Baudrate, Parity usw.). Andere Kommunikationsparameter an dem FB_IEC870_SerialLineCtrl-Funktionsbaustein sind in diesem Mode irrelevant.

Kommunikation über die seriellen Beckhoff Busklemmen KL6xxx

- In diesem Fall wird der Mode-Parameter auf den Wert: **SERIALLINEMODE_KL6_5B_STANDARD** gesetzt.
- Im TwinCAT System Manager werden die *KL6_in* - und *KL6_out* - Variablen mit den entsprechenden I/O-Variablen der seriellen Klemme KL6xxx verknüpft.
- Die Schnittstelle wird in der TwinCAT SPS durch die Instanz des FB_IEC870_SerialLineCtrl-Funktionsbausteins konfiguriert. Die Kommunikationsparameter wie Baudrate, Parity usw. sind an diesem Baustein einzustellen.

Kommunikation über die seriellen Beckhoff Busklemmen 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 diesem Fall wird der Mode-Parameter auf den Wert: **SERIALLINEMODE_EL6_22B** gesetzt.
- Im TwinCAT System Manager werden die *EL6_in* - und *EL6_out* - Variablen mit den entsprechenden I/O-Variablen der seriellen Klemme EL6xxx verknüpft.
- Die Schnittstelle wird und muss im TwinCAT System Manager konfiguriert werden (Baudrate, Parity usw.). Andere Kommunikationsparameter an dem FB_IEC870_SerialLineCtrl-Funktionsbaustein sind in diesem Mode irrelevant.

7.1.3 Applikationsobjekt-Datenbank der Unterstation definieren und konfigurieren

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Applikationsobjekte = Single Points, Double Points, Measured Values, Short Floating Point Values usw.

In diesem Beispiel wurden die Befehle so konfiguriert, dass die Prozessdaten der Befehle im gleichen Speicherbereich aber auf einem anderen Byte-/Bit-Offset wie die Daten der Information in Überwachungsrichtung liegen. Sie können aber auch die Befehle auf den gleichen Byte- Bit-Offset wie die Information in Überwachungsrichtung legen.

Beispiel:

C_SC_NA_1 mit IOA = 10 auf den gleichen Byte- und Bit-Offset wie M_SP_NA_1 mit IOA = 100 (beide Byte-Offset = 100 und Bit-Offset = 0). In diesem Fall wird eine Wertänderung durch ein Kommando von der Leitstation eine Übertragung des M_SP_NA_1 mit der Objektadresse 100 und Übertragunsursache <11> (returned by remote command) zur Folge haben.

Als Beispiel konfigurieren wir in dem Einführungsprojekt folgende Applikationsobjekte:

Voraussetzungen

Array-element	ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
0	M_SP_NA_1	100	Generalabfrage	0	Merker	100	0	1 Bit
1	M_SP_NA_1	101	Generalabfrage	0	Merker	100	1	1 Bit
2	M_SP_TB_1	102	Generalabfrage	0	Merker	100	2	1 Bit
3	M_DP_NA_1	200	Generalabfrage	0	Merker	200	0	2 Bits
4	M_DP_NA_1	201	Generalabfrage	0	Merker	200	2	2 Bits
5	M_DP_TB_1	202	Generalabfrage	0	Merker	200	4	2 Bits
6	M_ST_NA_1	300	Generalabfrage	0	Merker	300	0	1 Byte
7	M_ST_NA_1	301	Generalabfrage	0	Merker	301	0	1 Byte
8	M_ST_TB_1	302	Generalabfrage	0	Merker	302	0	1 Byte
9	M_BO_NA_1	400	Generalabfrage	0	Merker	400	0	4 Byte
10	M_BO_NA_1	401	Generalabfrage	0	Merker	404	0	4 Byte
11	M_BO_TB_1	402	Generalabfrage	0	Merker	408	0	4 Byte
12	M_ME_NA_1	500	Generalabfrage	0	Merker	500	0	2 Byte
13	M_ME_NA_1	501	Generalabfrage	0	Merker	502	0	2 Byte
14	M_ME_TD_1	502	Generalabfrage	0	Merker	504	0	2 Byte
15	M_ME_NB_1	600	Generalabfrage	0	Merker	600	0	2 Byte
16	M_ME_NB_1	601	Generalabfrage	0	Merker	602	0	2 Byte
17	M_ME_TE_1	602	Generalabfrage	0	Merker	604	0	2 Byte
18	M_ME_NC_1	700	Generalabfrage	0	Merker	700	0	4 Byte
19	M_ME_NC_1	701	Generalabfrage	0	Merker	704	0	4 Byte
20	M_ME_TF_1	702	Generalabfrage	0	Merker	708	0	4 Byte
21	M_IT_NA_1	800	Generalzählerabfrage und Mode A (lokal Umspeichern mit Spontanübertragung alle 15s)	0	Merker	800	0	4 Byte
22	M_IT_NA_1	801	Generalzählerabfrage und Mode A (lokal Umspeichern mit Spontanübertragung alle 15s)	0	Merker	804	0	4 Byte
23	M_IT_TB_1	802	Generalzählerabfrage und Mode A (lokal Umspeichern mit Spontanübertragung alle 15s)	0	Merker	808	0	4 Byte
Commands								
24	C_SC_NA_1	10	-	0	Merker	2100	0	1 Bit
25	C_SC_NA_1	11	-	0	Merker	2100	1	1 Bit
26	C_SC_TA_1	12	-	0	Merker	2100	2	1 Bit
27	C_DC_NA_1	20	-	0	Merker	2200	0	2 Bit
28	C_DC_NA_1	21	-	0	Merker	2200	2	2 Bit

Array-element	ASDU identifier	Objekt-adresse IOA	Group-Konfigurationsparameter	Basiszeit-multiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
29	C_DC_TA_1	22	-	0	Merker	2200	4	2 Bit
30	C_RC_NA_1	30	-	0	Merker	2300	0	1 Byte
31	C_RC_NA_1	31	-	0	Merker	2301	0	1 Byte
32	C_RC_TA_1	32	-	0	Merker	2302	0	1 Byte
33	C_BO_NA_1	40	-	0	Merker	2400	0	4 Byte
34	C_BO_NA_1	41	-	0	Merker	2404	0	4 Byte
35	C_BO_TA_1	42	-	0	Merker	2408	0	4 Byte
36	C_SE_NA_1	50	-	0	Merker	2500	0	2 Byte
37	C_SE_NA_1	51	-	0	Merker	2502	0	2 Byte
38	C_SE_TA_1	52	-	0	Merker	2504	0	2 Byte
39	C_SE_NB_1	60	-	0	Merker	2600	0	2 Byte
40	C_SE_NB_1	61	-	0	Merker	2602	0	2 Byte
41	C_SE_TB_1	62	-	0	Merker	2604	0	2 Byte
42	C_SE_NC_1	70	-	0	Merker	2700	0	4 Byte
43	C_SE_NC_1	71	-	0	Merker	2704	0	4 Byte
44	C_SE_TC_1	72	-	0	Merker	2708	0	4 Byte

Datenbankvariable deklarieren

Die Applikationsobjekt-Datenbank ist eine Array-Variable vom Typ ST_IEC870_5_101AODBEntry [▶ 312]. Jedes Array-Element entspricht einem Applikationsobjekt. Die maximale Anzahl der Applikationsobjekte ist frei wählbar und nur durch den verfügbaren Speicher begrenzt. Sie müssen sich auf eine konstante maximale Anzahl während der SPS-Programmierung festlegen. Zur Laufzeit kann die maximale Anzahl der Applikationsobjekte nicht mehr verändert werden.

In unserem Beispiel werden 50 Applikationsobjekte deklariert. Diese Anzahl reicht für die meisten Anwendungen aus. Beachten Sie, dass sehr viele Applikationsobjekte auch entsprechend viel Speicher und Laufzeit benötigen.

Definieren Sie folgende Variable in MAIN:

```
PROGRAM MAIN
VAR
     AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
END_VAR
```

Applikationsobjekte konfigurieren

Während der Konfiguration der einzelnen Applikationsobjekte werden unter anderem der Objekt-Typ (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 usw.), die Objekt-Adresse und weitere Objekt-Parameter festgelegt.

Die Konfiguration der gewünschten Applikationsobjekte wird zur Programmlaufzeit durchgeführt. Jedes Applikationsobjekt (Datenbank-Array-Element) wird durch einen einmaligen Aufruf der F_iecInitAOEntry [▶ 277]-Funktion konfiguriert. Das zu konfigurierende Array-Element wird an die Funktion per VAR_IN_OUT übergeben. Im Regelfall wird die Konfiguration beim SPS-Programmstart einmalig in einer Init-Routine durchgeführt. Die Funktion F_iecInitAOEntry erwartet folgende Funktionsparameter (von links nach rechts):

```
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: Applikationsobjekt-Typ (ASDU identifier [▶ 326], z.B.: M_SP_NA_1 für Single Point oder M_DP_NA_1 für Double Point). Beachten Sie, dass nur die in der Kompatibilitätsliste aufgeführten ASDU-Typen verwendet werden können. Unzulässige Typen werden ignoriert.

objAddr : Objektadresse, z.B. 100. Jedes Applikationsobjekt sollte mit einer eindeutigen Adresse konfiguriert werden.

group: Group-Konfigurationsparameter. Die verfügbaren Group-Parameter sind als Konstanten definiert und können mit ODER-Operator kombiniert werden. Z.B.: IEC870_GRP_INROGEN OR IEC870_GRP_PERCYC.

Hier finden Sie die Beschreibung aller Group-Konfigurationsparameter [▶ 347].

mutiplier: Basiszeitmultiplikator für die zyklische/periodische Datenübertragung (0=Deaktiviert). Die Basiszeit wird über die Systemparameter konfiguriert. Wurde die Basiszeit z.B. auf T#10s gesetzt und der Multiplikator auf den Wert 2, dann werden die periodischen/zyklischen Daten des Applikationsobjekts alle 20 Sekunden gesendet.

ioMapType: Dieser Parameter [▶ 329] legt fest, aus oder in welchen Prozessdatenbereich der TwinCAT SPS die IEC-Prozessdaten zur Laufzeit gemappt werden sollen (inputs, outputs, memory, data).

byteOffs: Prozessdatenbereich Byteoffset;

bitOffs: Prozessdatenbereich Bitoffset;

dbEntry: Applikationsobjekt das konfiguriert werden soll (ein Datenbankvariable-Arrayelement, das an die Funktion per VAR_IN_OUT übergeben wird).

Um die Applikationsobjekte beim Programmstart zu konfigurieren wird in MAIN folgender SPS-Code hinzugefügt:

```
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 der SPS- und IEC-Prozessdaten

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Die TwinCAT SPS-Prozessdaten werden zur Programmlaufzeit zyklisch in die IEC-Prozessdaten (Applikationsobjekte) und umgekehrt gemappt (kopiert). Für das Mapping der IEC<->SPS Prozessdaten können bis zu 4 Prozessdatenbereiche (IO-Eingänge, IO-Ausgänge, Merkerbereich, Datenbereich) als Puffervariablen im SPS-Programm deklariert werden. Die Bytegröße der Puffer ist frei wählbar und kann für jeden Bereich unterschiedlich gewählt werden. Unbenutzte Bereiche müssen nicht unbedingt deklariert werden.

In unserem Einführungsbeispiel deklarieren wir 4 SPS-Prozessdatenbereiche mit jeweils 3000 Bytes:

```
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
```

Die Zuordnung, wie die Prozessdaten zur Laufzeit gemappt werden sollen, wird während der Konfiguration der Applikationsobjekte mit der F_iecInitAOEntry [▶ 277]-Funktion festgelegt.
Siehe auch in: Applikationsobjekte definieren und konfigurieren [▶ 374].

Die Puffervariablen wurden nun als Byte-Arrays deklariert. Um auf die gewünschten Daten besser zugreifen zu können definieren wir die einzelnen Variablen ein zweites Mal und legen diese auf die entsprechenden Byte/Bit-Offsetadressen. Bei einer Änderung im Byte-Array wird die entsprechende einzelne Variable gleichzeitig geändert und umgekehrt. Dies ist aber nicht zwingend notwendig. Sie können direkt auf die Bytes/Bits der Byte-Array-Puffervariablen zugreifen.

```
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 der IEC<->SPS Prozessdaten in der Unterstation

Prozessdaten in Überwachungsrichtung (Slave->Master information)

Beispiel 1

Single point information (M_SP_NA_1) mit der IOA = 100, SPS Merkerbereich, Byteoffset = 100, Bitoffset = 0.

msgSingle_0 == memory[100].0 -> Unterstation FB -> ... -> Leitstation

Beispiel 2

Measured value, short floating point value (M_ME_NC_1) mit der IOA = 700, SPS Merkerbereich, Byteoffset = 700, Bitoffset = 0 (bedeutungslos).

msgFloating_0 == memory[700..703] -> Unterstation FB -> ... -> Leitstation

Prozessdaten in Steuerungsrichtung (Master->Slave commands)

Beispiel 1

Single command state (C_SC_NA_1) mit der IOA = 10, SPS Merkerbereich, Byteoffset = 2100, Bitoffset = 0.

Leitstation -> ... -> Unterstation FB -> memory[2100].0 == cmdSingle_0

Beispiel 2

Set point, short floating point value (C_SE_NC_1) mit der IOA = 70, SPS Merkerbereich, Byteoffset = 2700, Bitoffset = 0 (bedeutungslos).

Leitstation -> ... -> Unterstation FB -> memory[2700..2703] == cmdFloating_0

7.1.5 Instanz der IEC60870-5-101 Unterstation deklarieren und aufrufen

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Die gesamte Funktionalität einer Unterstation ist im Funktionsbaustein FB_IEC870_5_101Slave gekapselt. Mit einer Instanz kann eine Verbindung zum Master aufgebaut werden. Das *hSerial*-Verbindungshändle der schnellen Task muss als VAR_IN_OUT-Variable an die Unterstation übergeben werden.

Fügen Sie im Deklarationsteil von *P_MAIN_LowSpeed* folgenden SPS-Code ein:

```
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
```

und im Programmteil wird die Instanz aufgerufen:

```
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-Protokollparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Über die IEC60870-5-101-Protokollparameter kann das Verhalten der Unterstation an die Anforderungen des Masters angepasst werden. Die meisten Parameter sind mit Defaultwerten vorbelegt, so dass diese nicht verändert werden müssen.

Im unserem Beispiel setzen wir die Linkadresse und die Bytelänge der Linkadresse:

```
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
```

Die Dokumentation aller Übertragungsprotokoll-Parameter finden Sie hier: ST_IEC870_5_101PotocolParams [▶ 406].

7.1.7 Systemparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Über die Systemparameter wird z.B. die gemeinsame ASDU-Adresse und die Anwenderfunktionen der Unterstation konfiguriert.

In unserer Einführung konfigurieren wir folgende Systemparameter:

- Die gemeinsame ASDU-Adresse wird auf 7 gesetzt. (*asduAddr*)
- Die Systemzeit der Unterstation wird während der Initialisierung mit der Systemzeit des lokalen TwinCAT PC's synchronisiert (*bUsePCTime*).
- Die Synchronisierung der Systemzeit der Unterstation mit dem Uhrzeitsynchronisationsbefehl wird aktiviert (*bSyncTime*).
- Während der Synchronisierung der Systemzeit in der Unterstation soll die Systemzeit des TwinCAT PC's nicht synchronisiert werden (*bSyncPCTime*).
- Das Senden von M_EI_NA_1 (End of init) an die Zentralstation wird aktiviert (*bEndOfInit*).
- Das Senden der periodischen/zyklischen Daten wird deaktiviert (*bPerCyclic*). Die Basiszeit fürs Senden dieser Daten wird auf 5s gesetzt.
- Hintergrundabfrage wird deaktiviert (*bBackScan*). Die Zykluszeit für Hintergrundabfrage wird auf 30s gesetzt.
- Das lokale Umspeichern und Reset der Zählerstände wird aktiviert (*bPerFRZ*) und die Zykluszeit fürs Umspeichern und Reset auf 15s gesetzt.
- Das loggen der Debugmeldungen im Application-Log wird aktiviert (*dbgMode*). Es werden Änderungen im Gerätestatus gelogt.

Fügen Sie folgenden SPS-Code in Ihr SPS-Projekt ein:

```
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
```

Die Dokumentation aller Systemparameter finden Sie hier: ST_IEC870_5_101SystemParams [▶ 315].

7.1.8 Stationsabfrage

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Der Stationsabfragebefehl wird von der Zentralstation eingeleitet. Im Kennungsfeld des Befehls ist auch die Gruppe (1 bis 16 oder allgemein) festgelegt. Die Unterstation überträgt die zu dieser Gruppe dazugehörigen Applikationsobjekte mit der Übertragungsursache *<20>* bis *<36>* an die Zentralstation. Applikationsobjekte mit Zeitmarken werden ohne Zeitmarken übertragen.

Konfiguration der Systemparameter

- Es müssen keine speziellen Systemparameter gesetzt werden;

Konfiguration der Applikationsobjekte

- Der Datenpunkt muss einer oder mehreren Groupen zugeordnet werden. Der Groupparameter muss gesetzt werden. Eine Übersicht aller verfügbaren Grouppen finden Sie hier: Group configuration flags [▶ 347].

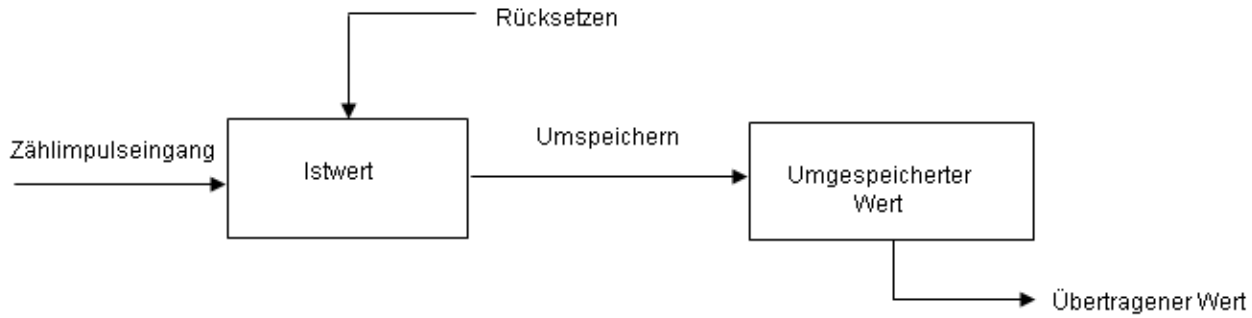
Beispielkonfiguration für einen Datenpunkt der der Gruppe: 1 und der Gruppe: Allgemein zugerodnet wurde.

```
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 Zählwertübertragung (counter interrogation)

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Allgemeines Modell für die Zählwertübertragung:



Die Istwerte werden durch Zähler aufsummiert. Die Istwerte können durch einen Umspeicherbefehl, der entweder von der Zentralstation empfangen oder örtlich (lokal in der Unterstation) erzeugt wird, periodisch in umgespeicherte Werte umgespeichert (kopiert) werden. Nach dem Umspeichern wird der erfasste Wert entweder auf Null zurückgesetzt (Erfassen von Inkrementalwerten) oder der Zähler fährt mit seinem Betrieb fort (Erfassen von Zählerständen).

Applikationsobjekte mit Zählwerten werden Gruppen zugeordnet. Die Gruppen werden einzeln umgespeichert (frozen), zurückgesetzt (reset) oder übertragen. Die Zentralstation sendet Zählwertabfragebefehle an die Unterstation. In einem Kennungsfeld des Befehls (QCC) wird die durchzuführende Aktion (FRZ) und Gruppe (RQT) festgelegt.

Die Zuordnung der Applikationsobjekte zu den einzelnen Gruppen (1 bis 4 oder allgemein) wird während der Konfiguration durch den Group-Flagparameter festgelegt. Es gibt vier Betriebsarten für die Erfassung von Zählerständen und Inkrementalwerten. Zu jeder Betriebsart sind einige Hinweise zur Konfiguration der Systemparameter oder der Applikationsobjekte aufgeführt.

Betriebsart A: Örtlich Umspeichern mit Spontanübertragung

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden spontan übertragen, nachdem die Funktion Umspeichern oder Umspeichern mit Rücksetzen durchgeführt wurde. Die Zentralstation gibt in dieser Betriebsart keine Zählwertabfragebefehle aus.

Konfiguration der Systemparameter:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

Der erste Parameter aktiviert das örtliche Umspeichern oder Umspeichern mit Rücksetzen. Der zweite Parameter gibt die Zykluszeit an in der das Umspeichern oder Umspeichern mit Rücksetzen durchgeführt wird (z.B.: alle 60 Sekunden).

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF-Groupparameter darf nicht gesetzt werden. Dieser würde die spontane Datenübertragung der Zählwerte verhindern.
- Der Zählwert wird umgespeichert, wenn IEC870_GRP_LOCFREEZE-Groupparameter gesetzt wurde.
- Der Zählwert wird zurückgesetzt, wenn IEC870_GRP_LOCRESET-Groupparameter gesetzt wurde.
- Das Örtliche Umspeichern oder Umspeichern mit Rücksetzen wird gleichzeitig für alle Gruppen (1 bis 4 oder allgemein) durchgeführt.

Betriebsart B: Örtliches Umspeichern mit Zählerabfrage

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden durch Zählwertabfragebefehle von der Zentralstation abgefragt. In diesem Fall darf die Zentralstation im Befehlskennungsfeld das Umspeichern oder Umspeichern mit Rücksetzen nicht benutzen (FRZ=0). Die Zählwerte werden allgemein oder in Gruppen (groups) 1 bis 4 abgefragt.

Konfiguration der Systemparameter:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

Der erste Parameter aktiviert das örtliche Umspeichern oder (und) Rücksetzen. Der zweite Parameter gibt die Zykluszeit an in der das Umspeichern oder Umspeichern mit Rücksetzen durchgeführt wird (z.B.: alle 60 Sekunden).

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF-Groupparameter muss gesetzt werden. Die Zählwerte sollen nicht spontan zur Zentralstation übertragen werden.
- Der Zählwert wird umgespeichert, wenn IEC870_GRP_LOCFREEZE-Groupparameter gesetzt wurde.
- Der Zählwert wird zurückgesetzt, wenn IEC870_GRP_LOCRESET-Groupparameter gesetzt wurde.
- Das örtliche Umspeichern oder Umspeichern mit Rücksetzen wird gleichzeitig für alle Gruppen (1 bis 4 oder allgemein) durchgeführt.

Betriebsart C: Zentralstation leitet das Umspeichern, Umspeichern mit Rücksetzen oder Rücksetzen ein

Ein Zählwertabfragebefehl wird periodisch von der Zentralstation an die Unterstation ausgegeben, um das Umspeichern oder (und) Rücksetzen zu steuern. Dieser Befehl hat aber noch keine Übertragung der Zählwerte zur Folge. Erst ein nachfolgender Zählwertabfragebefehl wird von der Zentralstation gesendet, um die umgespeicherten Zählwerte einzusammeln. Ähnlich, wie bei der Betriebsart B.

Konfiguration der Systemparameter:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Das örtliche Umspeichern oder (und) Rücksetzen muss deaktiviert werden. Der zweite Parameter wird ignoriert.

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF muss gesetzt werden. Die Zählwerte sollen nicht spontan zur Zentralstation übertragen werden.
- IEC870_GRP_LOCFREEZE- und IEC870_GRP_LOCRESET-Groupparameter dürfen nicht gesetzt werden. Die Zentralstation leitet das Umspeichern oder (und) Rücksetzen ein.
- Die Zählwerte können einzelnen Gruppen (1 bis 4 oder allgemein) zugeordnet und abgefragt werden (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1,IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

Betriebsart D: Zentralstation leitet das Umspeichern und (oder) Rücksetzen ein und die umgespeicherten Werte werden spontan übertragen

Diese Betriebsart ist eine Kombination des Zählwertbefehls von der Zentralstation wie für Betriebsart C mit einer spontanen Übertragung der Zählwerte wie bei der Betriebsart A.

Konfiguration der Systemparameter:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Das örtliche Umspeichern oder (und) Rücksetzen muss deaktiviert werden. Der zweite Parameter wird ignoriert.

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF-Groupparameter darf nicht gesetzt werden. Dieser würde die spontane Datenübertragung der Zählwerte verhindern.
- IEC870_GRP_LOCFREEZE- und IEC870_GRP_LOCRESET-Groupparameter dürfen nicht gesetzt werden. Die Zentralstation leitet das Umspeichern oder (und) Rücksetzen ein.
- Die Zählwerte können einzelnen Gruppen (1 bis 4 oder allgemein) zugeordnet und abgefragt werden (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1,IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

7.1.10 Uhrzeitsynchronisation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Das Verhalten, wie die Systemzeit der Unterstation synchronisiert werden soll, ist über die Systemparameter konfigurierbar.

- Die Systemzeit der Unterstation kann während der Initialisierung mit der Systemzeit des lokalen TwinCAT PC's synchronisiert werden;
- Beim Empfang eines Uhrzeit-Synchronisationsbefehls von der Zentralstation kann die Systemzeit der Unterstation ebenfalls synchronisiert werden;
- Die Systemzeit des lokalen TwinCAT PC's kann beim Empfang eines Uhrzeit-Synchronisationsbefehls auch synchronisiert werden.

7.1.11 Hintergrundabfrage

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Die Hintergrundabfrage wird zum Auffrischen der Prozessinformationen von der Unterstation zur Zentralstation als zusätzlicher Sicherheitsbeitrag zur Stationsabfrage und spontanen Übertragung angewendet.
Applikationsobjekte mit Typkennungen wie für die Stationsabfrage, dürfen mit der Übertragungsursache <2> Hintergrundabfrage stetig mit niedriger Priorität übertragen werden. Die zulässigen ASDU-Typkennungen sind in der Kompatibilitätsliste der Station aufgelistet (Tabelle Typkennung <-> Übertragungsursache). Die Hintergrundabfrage wird von der Unterstation eingeleitet und ist unabhängig von den Stationsabfragebefehlen.

Konfiguration der Systemparameter

Der Übertragungszyklus wird durch Systemparameter [▶ 315] in der Unterstation festgelegt.

```
bBackScan := TRUE;
tBackScanCycle := T#30s;
```

Konfiguration der Applikationsobjekte

Applikationsobjekte, deren Prozessdaten als Hintergrundabfrage übertragen werden sollen, müssen mit dem Group-Flag: IEC870_GRP_BACKGROUND konfiguriert werden.

Beispiel:

```
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 Zyklische Datenübertragung

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Die zyklische Datenübertragung wird ähnlich wie die Hintergrundabfrage von der Unterstation eingeleitet und ist unabhängig von anderen Befehlen aus der Zentralstation. Durch die zyklische Datenübertragung werden die Prozessdaten der Zentralstation kontinuierlich aufgefrischt. Bei den Prozessdaten handelt es sich meistens um Messwerte, die in regulären Zeitabständen erfasst werden. Die zyklische Datenübertragung wird oft benutzt um nicht-zeitkritische, oder sich nicht so schnell ändernde Prozessdaten zu überwachen (z.B. ein Temperatursensor). Die zyklischen/periodischen Daten werden zur Zentralstation mit der Übertragungsursache *<1> Periodic/Cyclic* übertragen. Die zulässigen ASDU-Typkennungen sind in der Kompatibilitätsliste der Station aufgeführt (Tabelle Typkennung <-> Übertragungsursache). Die zyklische Datenübertragung kann über die Systemparameter und die Konfigurationsparameter der Applikationsobjekte konfiguriert werden.

Konfiguration der Systemparameter:

```
bPerCyclic       : BOOL := TRUE;
tPerCyclicBase   : TIME := T#60s;
```

Der erste Parameter aktiviert die zyklische Übertragung. Der zweite Parameter ist die Basiszeit der zyklischen/periodischen Datenübertragung (hier z.B. 60 Sekunden).

Konfiguration der Applikationsobjekte:

- IEC870_GRP_PERCYC-Groupparameter muss gesetzt werden;
- Der Multiplikator-Parameter *(multiplier)* der F_iecInitAOEntry-Funktion muss auf einen Wert <> Null gesetzt werden. Beispiel: Bei einem Multiplikator = 2 und Basiszeit von 60 Sekunden werden die die Prozessdaten des Applikationsobjekts alle 120 Sekunden zur Zentralstation gesendet;

Beispielkonfiguration für einen Messwert der zyklisch alle 120 Sekunden zur Zentralstation übertragen werden soll (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 Befehlsübertragung

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Befehle können von der Zentralstation in Steuerungsrichtung (zur Unterstation) gesendet werden.
Ein Einzelbefehl mit der Typkennung 45 (C_SC_NA_1) wird zur Steuerung eines Applikationsobjekts benutzt, das in Überwachungsrichtung als Einzelmeldung (M_SP_NA_1, M_SP_TA_1 oder M_SP_TB_1) übertragen wird. Ein Doppelbefehl (C_DC_NA_1) wird zur Steuerung eines Applikationsobjekts benutzt, das in Überwachungsrichtung als Doppelmeldung (M_DP_NA1, M_DP_TA_1 oder M_DP_TB_1) übertragen wird, usw.

Konfiguration der Systemparameter:

- Es müssen keine speziellen Systemparameter gesetzt werden;

Konfiguration der Applikationsobjekte:

- Die Applikationsobjekte müssen als Befehle (Datentypen in Steuerungsrichtung) konfiguriert werden;
- Die Addressen der Informationsobjekte (IOA's) müssen den Adressen in der Leitstation entsprechen;

Beispiele:

Single command mit der IOA = 10. Der empfangene Wert wird in den Merkerbereichspuffer, Byteoffset = 100, Bitoffset = 0 kopiert.

```
F_iecInitAOEntry( C_SC_NA_1, 10, 0, 0, MAP_AREA_MEMORY, 100, 0, AODB[24] );
```

Double command mit der IOA = 20. Der empfangene Wert wird in den Merkerbereichspuffer, Byteoffset = 200, Bitoffset = 0..1 kopiert.

```
F_iecInitAOEntry( C_DC_NA_1, 20, 0, 0, MAP_AREA_MEMORY, 200, 0, AODB[27] );
```

Set point, scaled value mit der IOA = 60. Der empfangene Wert wird in den Merkerbereichspuffer, Byteoffset = 600..601, Bitoffset = 0 kopiert.

```
F_iecInitAOEntry( C_SE_NB_1, 60, 0, 0, MAP_AREA_MEMORY, 600, 0, AODB[39] )
```

7.1.14 Abfrage-/Lese-Prozedur

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Die Zentralstation sendet Abfragebefehle an die Unterstation.
In dem Abfragebefehl wird die Adresse des abzufragenden Applikationsobjekts übertragen. Die Daten dieses Applikationsobjekts sollen an die Zentralstation gesendet werden. Die Unterstation sendet die Daten mit der Übertragungsursache *<5> Abfrage oder abgefragt*. Die zulässigen ASDU-Typkennungen sind in der Kompatibilitätsliste der Station aufgelistet (Tabelle Typkennung <-> Übertragungsursache).

Konfiguration der Systemparameter

- Es müssen keine speziellen Systemparameter gesetzt werden;

Konfiguration der Applikationsobjekte

- Es müssen keine speziellen Parameter gesetzt werden;

7.1.15 Doppelübertragung

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Alle Applikationsobjekte (Informationsobjekte), die mit der Übertragungsursache <*3*> *Spontan* übertragen werden, dürfen zweimal übertragen werden, mit und ohne Zeitmarke. Diese Betriebsart wird "Doppelübertragung" genannt.
Doppelübertragung wird von der Unterstation zur Zeit nicht unterstützt.

7.1.16 Quality-Flags

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Die Quality-Flags (Quality-Descriptor) liefern der Zentralstation zusätzliche Informationen zur Qualität eines Applikationsobjekts. Die Quality-Flags können aus der SPS-Applikation mit Hilfe der F_iecSetAOQuality [▶ 279]-Funktion unabhängig voneinander gesetzt/zurückgesetzt werden. Mit der F_iecGetAOQuality [▶ 280]-Funktion kann der Status der Quality-Flags abgefragt werden. Jede Änderung der Quality-Flags führt zu einer spontanen Übertragung der Daten zur Zentralstation.

Folgende Quality-Flags werden intern von der Unterstation zur Laufzeit ausgewertet:

- IECQ_BL_ON (Blocked). Wurden die Prozessdaten des Applikationsobjekts für die Übertragung blockiert, dann wird das Mapping der SPS- und IEC-Prozessdaten für dieses Applikationsobjekt nicht ausgeführt;

Folgende Quality-Flags werden intern von der Unterstation zur Laufzeit gesetzt/zurückgesetzt:

- IECQ_IV_ON (Invalid). Die Unterstation setzt das Invalid-Flag wenn das Mapping der SPS- und IEC-Prozessdaten nicht durchgeführt werden konnte (z.B. wegen fehlerhafter Konfiguration des Applikationsobjekts). Dieses Verhalten kann durch einen gesetzten Group-Parameter: IEC870_GRP_IV_OFF deaktiviert werden.

Alle anderen Quality-Flags werden unverändert zur Zentralstation gesendet.

7.1.17 Test der Kommunikation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Durch das Setzen der *bChangeIO*-Variable auf TRUE wird eine einfache Simulation der Datenpunkte in Überwachungsrichtung aktiviert und mit FALSE deaktiviert. Bei einer aktiven Verbindung werden die Werte Zyklisch alle 3 Sekunden zur Leitstation übertragen.

```
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 Übertragungs- und Kommunikationsfehler

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

Die Stationsfehlermeldungen werden in einem FIFO abgelegt. Es können bis zu 10 Fehlermeldungen zwischengespeichert werden. Bei fatalen Kommunikationsfehlern (z.B. Fehler der Verbindungsschicht, die Checksumme des Frames passt nicht) wird die Verbindung unterbrochen und muss neu aufgebaut werden. Fehler in der Applikationsschicht (z.B. der ASDU-Sendepuffer ist wegen zu vieler Frames übergelaufen) werden nur geloggt und führen nicht zum Verbindungsabbruch. Es immer noch möglich auch bei diesen Fehlern die Verbindung aus der Applikation zu unterbrechen. Neben dem Fehler-Code wird auch die Fehlerquelle in der Fehlermeldung abgelegt. Dieses erleichtert die Lokalisierung des Fehlers.

Beispiel

Die anfallenden Fehlermeldungen einer IEC 60870-5-101 Unterstation können durch folgenden Aufruf ausgelesen werden:

```
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 SPS-API

Einfache Projekte mit kompletten Sourcen finden Sie hier: IEC60870-5-101 Unterstation [▶ 394].

Kompatibilitätsliste finden Sie hier: Interoperability check list

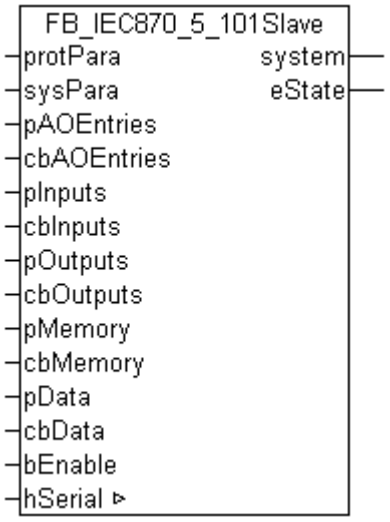
Übersicht der Fehlercodes finden Sie hier: Fehlercodes

Eine ausführliche Anleitung zur Implementierung der Unterstation in der SPS finden Sie hier: TUTORIAL [▶ 372]

Kurzanleitung

1. Erstellen Sie ein neues SPS-Projekt erstellen binden Sie die SPS-Bibliothek: **TcIEC870_5_101Slave.Lib** ein.
2. Legen Sie zwei SPS-Tasks an, eine schnelle (z.B. mit Zykluszeit T#1ms) und eine langsame (z.B. mit Zykluszeit T#10ms).
   Legen Sie zwei Programmbausteine an (z.B. *P_SerialComm_HighSpeed* und *P_MAIN_LowSpeed*). *P_SerialComm_HighSpeed* wird von der schnellen und *P_MAIN_LowSpeed* von der langsamen Task aufgerufen.
3. Legen Sie im *P_SerialComm_HighSpeed* eine Instanz des Funktionsbausteins FB_IEC870_SerialLineCtrl [▶ 403] an, konfigurieren Sie diese und rufen Sie diese auf.
   Je nach dem, ob über die seriellen Beckhoff-Klemmen oder über die serielle Schnittstelle des PCs kommuniziert wird, die Puffer: *KL6inData5B, KL6outData5B* oder *PcComInData, PcComOutData* anlegen und mit den entsprechen IO-Prozessdaten im TwinCAT Systemmanager verknüpfen.
4. Die Instanz der T_HSERIALCTRL [▶ 413]-Variablen dient dem Austausch der Tx/Rx-Telegramme zwischen den beiden Tasks (Programmen). Legen Sie diese Variable z.B. als globale Variable an.
5. Konfiguration der Datenpunkte: Legen Sie eine Array-Variable vom Typ ST_IEC870_5_101AODBEntry [▶ 312] an. Jedes Arrayelement entspricht einem Datenpunkt. Konfigurieren Sie die Datenpunkte mit Hilfe der Funktion F_iecInitAOEntry [▶ 277] zur Laufzeit (z.B. in einem Init-Schritt).
6. Legen Sie im *P_MAIN_LowSpeed* eine Instanz des Protokoll-Bausteins FB_IEC870_5_101Slave [▶ 390] an, konfigurieren Sie diese und rufen Sie diese auf.
7. Konfigurieren Sie die System- und Protokoll-Parameter passend zu den Parametern der Leitstation.

7.2.1 FB_IEC870_5_101Slave



Mit einer Instanz des Funktionsbausteins FB_IEC870_5_101Slave kann in der TwinCAT SPS eine IEC60870-5-101 Unterstation (Slave) implementiert werden.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial         : T_HSERIALCTRL;
END_VAR
```

hSerial : Verbindungs-Handle [▶ 413] zum FB_IEC870_SerialLineCtrl [▶ 403]-Funktionsbaustein. Über diese Variable werden mit dem FB_IEC870_SerialLineCtrl-Funktionsbaustein die zu sendenden und empfangenen Daten ausgetauscht.

VAR_INPUT

```
VAR_INPUT
    protPara        : ST_IEC870_5_101ProtocolParams;  (* IEC60870-5-101 serial link protocol communi
cation 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;  (* Poi
nter to the first element of application database object array *)
    cbAOEntries     : UDINT;                            (* Byte size (length) of application datab
ase 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-Protokollparameter [▶ 406].

sysPara: Systemparameter [▶ 315].

pAOEntries: Adresse der Applikationsobjekt-Datenbankvariablen [▶ 312].

cbAOEntries: Bytegröße der Applikationsobjekt-Datenbankvariablen.

pInputs: Adresse des SPS-Prozessdatenbereichs der Eingänge.

cbInputs: Bytegröße des SPS-Prozessdatenbereichs der Eingänge.

pOutputs: Adresse des SPS-Prozessdatenbereichs der Ausgänge.

cbOutputs: Bytegröße des SPS-Prozessdatenbereichs der Ausgänge.

pMemory: Adresse des SPS-Prozessdatenbereichs der Merker.

cbMemory: Bytegröße des SPS-Prozessdatenbereichs der Merker.

pData: Adresse des SPS-Datenbereichs.

cbData: Bytegröße des SPS-Datenbereichs.

bEnable : Aktiviert/Deaktiviert den Funktionsbaustein (Kommunikation und Verbindung).

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 [▶ 393]. Diese Variable dient anderen SPS Funktionen oder Funktionsbausteinen als Kommunikationsschnittstelle zum IEC-Gerät (hier: Unterstation).

- Membervariable *system.device* wird z.B. von der F_iecSetAOQuality [▶ 279]-Funktion als VAR_IN_OUT-Parameter erwartet.

- Membervariable *system.device.errors* ist ein Gerätefehler-Fifo. Die registrierten Fehler können von der SPS-Applikation ausgelesen und ausgewertet werden.

eState: Verbindungsstatus [▶ 411] zum Master.

Beispiel:

Beispielprojekte: IEC60870-5-101 Unterstation [▶ 394]

Aufruf 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 );
```

Im folgenden Beispiel wird der Gerätefehler-Fifo zyklisch ausgelesen und die registrierten Fehler ins Windows Application Log geschrieben.

```
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

7.2.2 F_GetVersionTcIEC870_5_101Slave

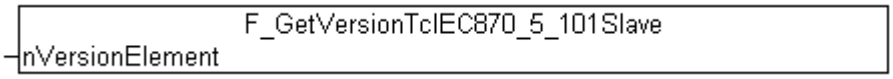


Abb. 6: F_GetVersionTcIEC870_5_101Slave

Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_101Slave: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden)
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: Kommunikationsschnittstelle [▶ 318] des IEC-Gerätes. Diese Variable dient anderen SPS Funktionen oder Funktionsbausteinen als Kommunikationsschnittstelle zum IEC-Gerät (hier: Unterstation).

service: IEC-Gerätedienste;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

7.3 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE Image-Version usw.).

2. Vergleichen/überprüfen Sie die Kompatibilitätsliste der Unterstation mit der Kompatibilitätsliste der Leitstation.

3. Überprüfen Sie die IO-Konfiguration und das Mapping der SPS-Variablen in TwinCAT System Manager (Konfiguration der seriellen Schnittstellen, Baudrate, Parity, Stopbits usw.). Vergleichen Sie die Parameter mit Parametern in der Leitstation.

4. Überprüfen Sie ob der Funktionsbaustein einen Fehlercode ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes.

5. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 406] (Link-Adresse, Länge der Link-Adresse, FRAMELength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern in der Leitstation.

6. Überprüfen Sie die am Funktionsbaustein eingestellten Systemparameter [▶ 315] (ASDU-Adresse, Länge der ASDU-Adresse, Länge der Informationsobjekt-Adresse, Länge der Übertragungsursache COT, usw.). Vergleichen Sie die Systemparameter mit den Parametern in der Leitstation.

7. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, Adresse des Informationsobjektes usw.).

8. Überprüfen Sie, ob die Leitstation einen Fehlercode ausgibt.

9. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben.

7.4 Beispiele

In den Beispielprojekten sind folgende Stationsparameter für die Unterstation eingestellt:

- 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**

Voraussetzungen

SPS Project	TwinCAT System Manager Konfiguration	Beschreibung	Download
MiniSlaveSample.pro	MiniSlaveSample.tsm	Einfache TwinCAT IEC60870-5-101 Unterstation. Ein **minimalistisches** SPS-Projekt mit einem Single-Point in Überwachungsrichtung (IOA = 100) und einem Single-Command in Steuerungsrichtung (IOA = 10).	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699281419.zip
TutorialSample.pro	TutorialSample.tsm	Einfache IEC60870-5-101 Unterstation	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11699280011.zip

8 TcIEC870_5_101Link: IEC 60870-5-101 Serial Link Interface (master/slave)

Die TwinCAT SPS-Bibliothek **TcIEC870_5_101Link.Lib** implementiert Übertragungsprozeduren für den Transport der ASDUs über die seriellen Schnittstellen des PCs und die Beckhoff KL6xxx-/EL6xxx-Klemmen.

"Low level"-Schnittstelle: IEC 60870-5-101 Serial Link Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können verändert werden (z.B. ein eigener Zeitstempel, Select/Execute oder eine besondere Befehlsausführung usw.); Weil nur die benötigten Dienste implementiert werden, kann eine hohe Performance erreicht werden; Hohe Performance bei vielen Datenpunkten;

Contra: Größerer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Diese Schnittstelle empfiehlt sich, wenn Sie:

- Mit der Protokollnorm vertraut sind;
- Eine Protokollkonverter-Applikation implementieren;
- In der Applikation die verfügbaren Norm-Funktionalitäten fast vollständig implementieren;
- Besondere Funktionalitäten verwenden, wie z.B. das Weiterleiten der Zeitstempel von einem Modbus-Gerät oder die Kontrolle über die Befehlsausführung erlangen;
- Funktionalitäten benötigen, die laut Kompatibilitätsliste nicht unterstützt werden;
- Viele Datenpunkte haben (>1000) und eine hohe Performance benötigen;

Innerhalb der Protokollstruktur liegt diese Schnittstelle oberhalb der Verbindungsschicht und implementiert bereits die nötigen Prozeduren und Übertragungstelegrammformate. Anwendungsfunktionen wie z.B. Generalabfrage, Uhrzeitsynchronisation oder Zählerabfrage sind in der Schnittstelle nicht implementiert, der Anwender kann aber mit Hilfe der Schnittstelle diese Anwendungsfunktionen selbst implementieren.

Protokollstruktur des Endsystems:

<table>
<tr><td rowspan="3">Auswahl von Anwendungsfunktionen aus IEC 60870-5-5

Auswahl von ASDU (Dienstdateneinheiten der Anwendungsschicht) aus

IEC 60870-5-3, IEC 60870-5-4 und IEC 60870-5-101

TwinCAT PLC Library: IEC 60870-5-101 Serial Link (low level) Transport Interface</td><td>Anwenderprozess</td></tr>
<tr><td rowspan="2">Anwendungsschicht (7)</td></tr>
<tr><td rowspan="4">N/A</td><td>Darstellungsschicht (6)</td></tr>
<tr><td>Sitzungsschicht (5)</td></tr>
<tr><td>Transportschicht (4)</td></tr>
<tr><td>Vermittlungsschicht (3)</td></tr>
<tr><td>unbalanced / balanced IEC 60870-5-2
IEC 60870-5-1 (FT 1.2)</td><td>Verbindungsschicht (2)</td></tr>
<tr><td>EIA RS485, RS232 (V.24), Fibre Optics</td><td>Physikalische Schicht (1)</td></tr>
</table>

Anmerkung: Die Schichten 3 bis 6 werden nicht benutzt.

8.1 SPS-API

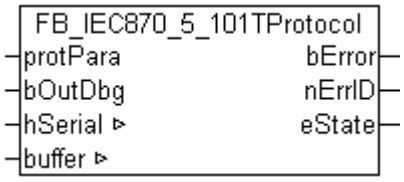
Einfache Projekte mit kompletten Sourcen finden Sie hier: Serial-Link-Interface-Beispiele.Serial-Link-Interface-Beispiele [▶ 413]

Eine SPS-Applikation, die über das Transport Interface mit einer Unterstation oder Zentralstation kommunizieren soll, benötigt folgende Resourcen:

- Eine Instanz des Kommunikationsbausteins: FB_IEC870_5_101TProtocol [▶ 397];
- Eine Instanz des TX/RX-Datenpuffers: ST_IEC870_5_101TBuffer [▶ 37];
- Eine Instanz des Funktionsbausteins zur Manipulation des TX/RX-Datenpuffers: FB_IEC870_5_101TBufferCtrl [▶ 37];

8.1.1 FB_IEC870_5_101TProtocol

Ab der Produktversion: TwinCAT PLC Library IEC870-5-101 Unterstation v2.0.2 und höher.



Der Kommunikationsbaustein FB_IEC870_5_101TProtocol implementiert die Übertragungsprozeduren der Verbindungsschicht nach der IEC 60870-5-1 und IEC 60870-5-2-Norm.
Beim Protokollfehler wird ein entsprechender Fehlercode am Ausgang des Funktionsbausteins ausgegeben und die Datenübertragung unterbrochen. Um den Datenaustausch erneut aktivieren zu können, muss die Aktion INIT aufgerufen werden. Es werden dabei z.B. die TX/RX-Datenpuffer zurückgesetzt. Der Kommunikationsbaustein erwartet eine TX/RX-Datenpuffervariable. Diese Variable muss per VAR_IN_OUT an den Baustein übergeben werden.

Der Funktionsbaustein besitzt folgende Aktionen:

- **INIT** (Führt eine Initialisierung des Funktionsbausteins durch);

Protokollkonfiguration

Der Kommunikationsbaustein besitzt eine protPara-Variable vom strukturierten Typ. Über diese Variable können Protokollparameter z.B. RX/TX-Timeoutzeiten usw. konfiguriert werden.

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: Verbindungs-Handle [▶ 413] zum FB_IEC870_SerialLineCtrl [▶ 403]-Funktionsbaustein. Über diese Variable werden mit dem FB_IEC870_SerialLineCtrl-Funktionsbaustein die zu sendenden und empfangenen Daten ausgetauscht.

buffer: TX/RX Datenpuffer.

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_101ProtocolParams; (* IEC60870-5-101 protocol parameters *)
    bOutDbg     : BOOL;        (* Enable/disable debug output *)
END_VAR
```

protPara: IEC60870-5-101-Protokollparameter [▶ 406].

bOutDbg:: Aktiviert/deaktiviert die Debug-Ausgabe der Frames in der TwinCAT System Manager-Loggeransicht.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED; (* Data transfer state
*)
ND_VAR
```

bError: Dieser Ausgang wird auf TRUE gesetzt, sobald ein Fehler bei der Datenübertragung aufgetreten ist.

nErrID: Liefert bei einem gesetzten bError-Ausgang einen Fehlercode;

eState: Verbindungsstatus [▶ 411] zum Master.

Beispiel:

Beispielprojekte: IEC60870-5-101 Serial Link Interface [▶ 413]

Voraussetzungen

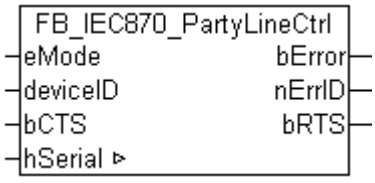
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_101Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; werden automatisch eingebunden)

Sehen Sie dazu auch

M_SP_NA_1 [▶ 37]

8.1.2 FB_IEC870_PartyLineCtrl

Ab der Produktversion: TwinCAT PLC Library IEC870-5-101 Unterstation v2.0.2 und höher.



Mit dem Funktionsbaustein FB_IEC870_PartyLineCtrl kann der Datenaustausch zur Zentralstation im Linienbetrieb (partyline) betrieben werden.

Der Funktionsbaustein muss in der SPS-Task zyklisch aufgerufen werden. Mit dem *eMode*-Eingang kann der Linienbetrieb aktiviert/deaktiviert werden. Wenn Sie den Funktionsbaustein nicht benutzen (Default Einstellung) wird kein Linienbetrieb betrieben. Die *deviceID*-Eingangsvariable muss der Geräte ID aus der TwinCAT System Manager IO-Konfiguration entsprechen (Allgemeinreiter der COM-Schnittstelle) und wird nur dann benötigt wenn Sie für die Kommunikation im Linienbetrieb die serielle PC-Schnittstelle benutzen.

Die hSerial-Variable ist eine Struktur und dient dem internen Datenaustausch zwischen der schnellen und langsamen Kommunikationstask. Jedes Mal, wenn der IEC-Slave senden will wird der *bRTS*-Ausgang (request to send) zuerst auf TRUE gesetzt. Nachdem der Sendebetrieb eingeschaltet wurde wird dies dem IEC-Slave durch das Setzen des *bCTS*-Eingangs (clear to send) auf TRUE mitgeteilt. Danach beginnt der IEC-Slave zu senden. Nachdem die Daten gesendet wurden (interne Hardware-Puffer sind leer) setzt der IEC-Slave den *bRTS*-Ausgang auf FALSE zurück. Jetzt kann die Sendeleitung für den anderen Teilnehmer freigegeben werden. Wenn dies geschehen ist, muss dies ebenfalls am *bCTS*-Eingang mit FALSE dem IEC-Slave mitgeteilt werden. D.h. der Zustand des *bCTS*-Eingangs folgt immer dem Zustand des *bRTS*-Ausgangs.

Bei der seriellen PC-Schnittstelle wird der *bRTS*-Ausgang erst dann auf FALSE gesetzt (Daten gesendet), wenn die ADS-Abfrage des internen Hardware-Sendepuffers Null Bytes im Puffer zurückliefert.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial         : T_HSERIALCTRL;
ND_VAR
```

hSerial : Verbindungs-Handle [▶ 413] zum FB_IEC870_SerialLineCtrl [▶ 403]-Funktionsbaustein. Über diese Variable werden mit dem FB_IEC870_SerialLineCtrl-Funktionsbaustein die zu sendenden und empfangenen Daten ausgetauscht.

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-Aktivierungsmode [▶ 411].

deviceID : TwinCAT System Manager Geräte-ID. Dieser Parameter wird nur bei der Kommunikation über die serielle PC-Schnittstelle benötigt.

bCTS : Clear to send (für den IEC-Link-Layer).

VAR_OUTPUT

```
VAR_IN_OUT
    bError      : BOOL;
    nErrID      : UDINT;
    bRTS        : BOOL := FALSE; (* Request to send *)
ND_VAR
```

bError : Wird TRUE, sobald ein Fehler aufgetreten ist.

nErrID: Liefert bei einem gesetzten *bError*-Ausgang den Fehlercode.

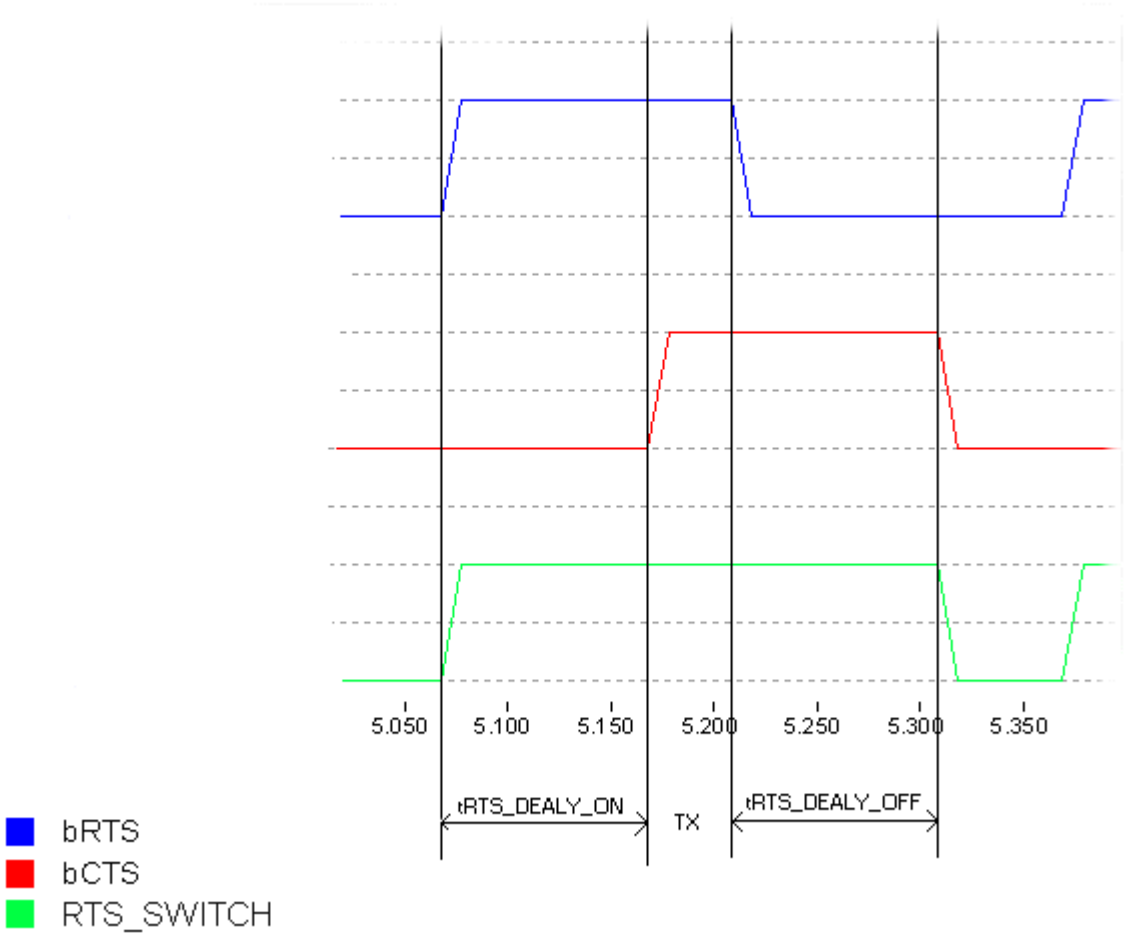
bRTS: Request to send (vom IEC-Link-Layer).

Beispiel für TwinCAT v2.10 Build < 1313 oder älter (CE image < 2.16 oder älter):

Implementierung des Linienbetriebs in der schnellen Kommunikationstask: Über die *RTS_SWITCH*-Variable wird die Leitung für den Sendebetrieb EIN- und AUS-geschaltet.

Die *tRTS_DEALY_ON*-Verzögerungszeit (Vorlaufzeit) stellt sicher, dass die Freischaltung der Leitung für den IEC-Slave abgeschlossen wurde, die *tRTS_DELAY_OFF*-Verzögerungszeit (Nachlaufzeit) stellt sicher, dass auch das letzte gesendete Datenbyte von der Zentralstation empfangen wurde.

FB_IEC870_PartyLineCtrl



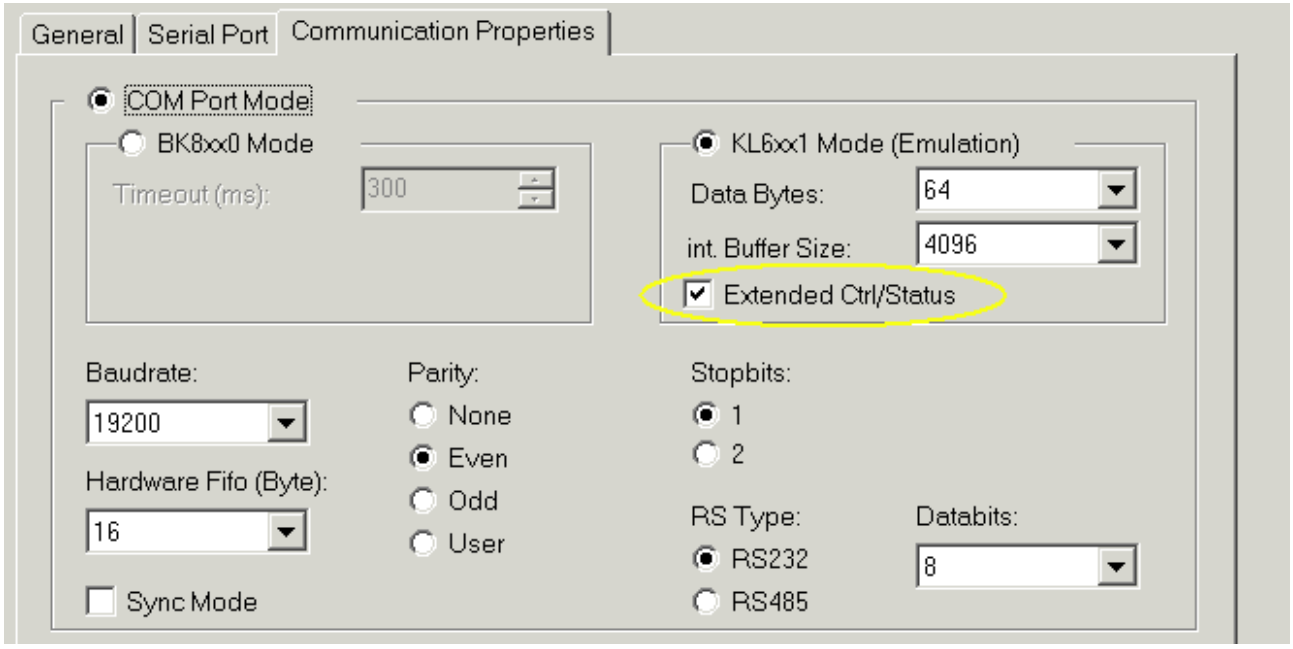

```
PROGRAM P_SerialComm_HighSpeed
VAR
    fbSerialLineCtrl    : FB_IEC870_SerialLineCtrl;
    Mode        : ComSerialLineMode_t := SERIALLINEMODE_PC_COM_PORT; (* SERIALLINEMODE_KL6_5B_STANDA
RD *)

    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 );
```

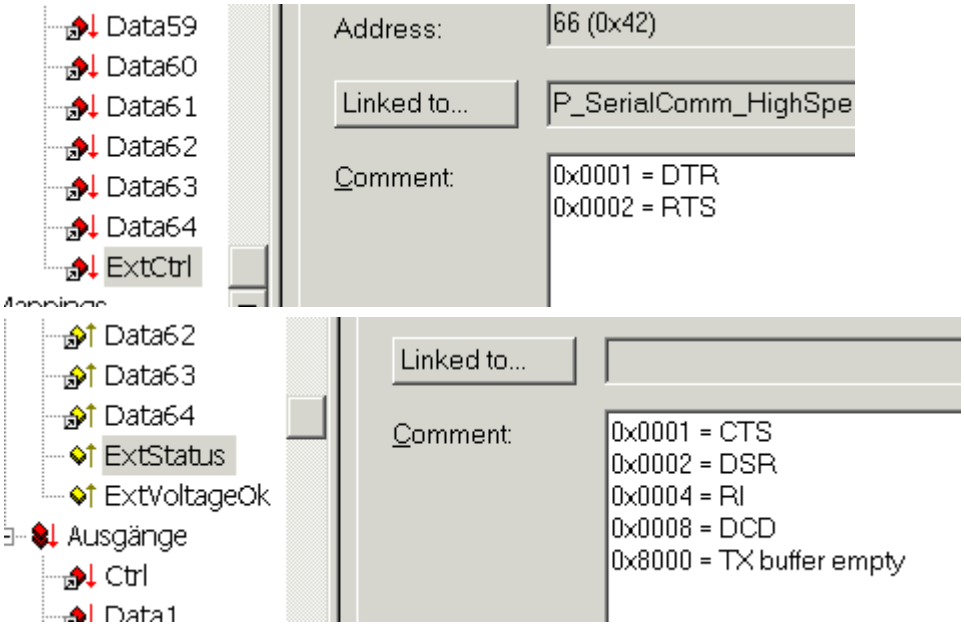
```
(* The deviceID may vary! *)
fbPartyLineCtrl( eMode:= eIEC870_PartylineMode_On, hSerial:= hSerial, deviceID := 1 );
IF fbPartyLineCtrl.bRTS <> fbPartyLineCtrl.bCTS THEN
    IF fbPartyLineCtrl.bRTS THEN
        RTS_SWITCH := TRUE; (* switch RTS line ON *)
        delay( in := TRUE, PT := tRTS_DEALY_ON ); (* 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
```

Beispiel für eine TwinCAT v2.10 Build >= 1313 oder neuer (CE image v2.16 oder neuer):

Bei einer neueren TwinCAT-Version kann die RTS-Leitung direkt als ein IO-Ausgang in die SPS gemappt werden. Sie müssen im TwinCAT System Manager die Extended Ctrl/Status Option aktivieren. Der eMode-Parameter an dem Funktionsbaustein muss *eMode := eIEC870_PartylineMode_Ext_On* gesetzt werden. Der *deviceID*-Parameter wird in dieser Betriebsart nicht benutzt und muss nicht konfiguriert werden.



Die RTS_SWITCH-Variable muss mit dem Bit 1 im ExtCtrl (RTS-Ausgang) und die TX_BUFFER_EMPTY-Variable mit dem Bit 15 im ExtStatus verknüpft werden.




```
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
```

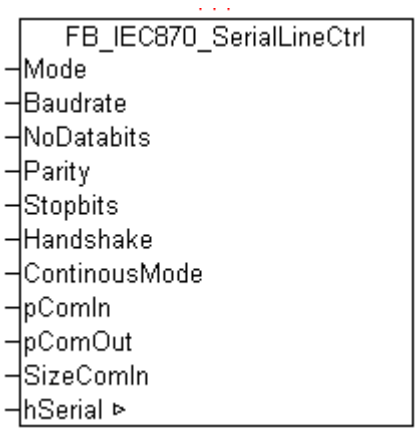
```
fbSerialLineCtrl( Mode := Mode,
        Baudrate := 19200,
        NoDatabits := 8,
        Parity := PARITY_EVEN,
        Stopbits := 1,
        Handshake := HANDSHAKE_NONE,
        ContinousMode := FALSE,
        pComIn := SEL( Mode = SERIALLINEMODE_KL6_5B_STANDARD, ADR( serial_in ), ADR( KL6_in ) ),
        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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.3 FB_IEC870_SerialLineCtrl



Der Funktionsbaustein FB_IEC870_SerialLineCtrl wickelt die Kommunikation zwischen einer seriellen Schnittstelle (KL60xx, EL60xx oder COM-Schnittstelle) und den IEC60870-5-101 SPS-Bausteinen ab. Wenn Sie für die Kommunikation eine serielle Klemme **KL60xx** benutzen, dann wird die Busklemme zuerst initialisiert und konfiguriert (Baudrate, Parity.. usw). Die Konfiguration der **PC-COM-Schnittstelle** und der Klemme **EL60xx** muss aber in TwinCAT System Manager durchgeführt werden. Die empfangenen und zu sendenden Daten werden in den internen Puffern der hSerial-Variablen gehalten. Der Funktionsbaustein muss in der SPS-Task zyklisch aufgerufen werden.

VAR_IN_OUT

```
VAR_IN_OUT
    hSerial         : T_HSERIALCTRL;
ND_VAR
```

hSerial [▶ 413] :

VAR_INPUT

```
VAR_INPUT
    Mode            : ComSerialLineMode_t  := SERIALLINEMODE_PC_COM_PORT; (* or SERIALLINEMODE_PC_CO
M_PORT *)
    Baudrate        : UDINT             := 19200;     (* KL60xx only: 115200, 57600, 38400, 19200, 9
600, 4800, 2400, 1200 *)
 NoDatabits         : BYTE      := 8;           (* KL60xx only: 7 or 8 *)
    Parity          : ComParity_t  := PARITY_EVEN;         (* KL60xx only: PARITY_NONE=0, PARITY_EVE
N=1, PARITY_ODD=2 *)
    Stopbits        : BYTE          := 1;         (* KL60xx only: 1 or 2 *)
    Handshake       : ComHandshake_t       := HANDSHAKE_NONE;      (* KL60xx only: HANDSHAKE_NONE=0,
 HANDSHAKE_RTSCTS=1, HANDSHAKE_XONXOFF=2 *)
    ContinousMode   : BOOL;                     (* KL60xx only: Don't start transmission before tran
smit buffer is filled *)
    pComIn      : POINTER TO BYTE;
```

```
    pComOut         : POINTER TO BYTE;
    SizeComIn       : UINT;
END_VAR
```

Mode : Der Mode-Eingang legt eindeutig fest, welche serielle Hardware verwendet wird.

Baudrate : Die Baudrate, soweit durch die serielle Hardware unterstützt

NoDatabits : Anzahl der Nutzdatenbits in einem Datenbyte

Parity : Typ des Paritybits eines Datenbytes

Stopbits : Anzahl der Stoppbits pro Datenbyte

Handshake : Typ des verwendeten Handshakes soweit durch die serielle Hardware unterstützt

ContinousMode : Schaltet das kontinuierliche Senden ein, wenn es durch die serielle Hardware unterstützt wird.

Wenn ContinousMode TRUE ist, werden gesendete Daten erst dann aus der seriellen Hardware abgeschickt, wenn der Hardware-Sendepuffer voll ist. Dadurch wird ein zeitlückenfreies Senden gewährleistet, solange die Datenmenge in der Größenordnung des Hardware-Sendepuffers liegt. Der continous mode wird nur in besonderen Fällen benötigt, wenn das Endgerät auf Zeitlücken mit einem Timeout reagiert.

pComIn : Universeller Pointer auf die Eingangsvariable der Prozessdaten der seriellen Hardware (Datentypen KL6inData, KL6inData5b, PcComInData, EL6inData22B). Der Pointer wird mit der *ADR*() Funktion zugewiesen.

pComOut : Universeller Pointer auf die Ausgangsvariable der Prozessdaten der seriellen Hardware (Datentypen KL6outData, KL6outData5b, PcComOutData, EL6outData22B). Der Pointer wird mit der *ADR*() Funktion zugewiesen.

SizeComIn : Größe des Eingangs-Prozessabbildes der verwendeten seriellen Hardware. Die Größe wird mit der *SIZEOF*() Funktion ermittelt und zugewiesen.

Beispiel 1:

Das Beispiel zeigt einen Aufruf in ST. Durch das Setzen der Mode-Variablen kann zwischen zwei Kommunikationswegen umgeschaltet werden.

Bei Mode = SERIALLINEMODE_PC_COM_PORT wird über eine serielle COM-Schnittstelle des PC's und bei Mode = SERIALLINEMODE_KL6_5B_STANDARD über eine KL6001 Busklemme kommuniziert (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 );
```

Beispiel 2:

In diesem Beispiel wird über eine EL6001 kommuniziert (22 Byte mode). Die Konfiguration der EL6001 Klemme (Baudrate, Parity usw.) muss in TwinCAT System Manager durchgeführt werden.

```
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
```

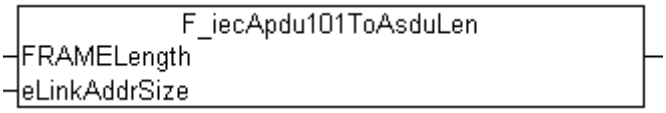
```
fbSerialLineCtrl( pComIn := ADR( EL6_in ),
        pComOut := ADR( EL6_out ),
        SizeComIn := SIZEOF( EL6_in ),
        hSerial := hSerial );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.4 F_iecApdu101ToAsduLen

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-101 Unterstation v2.0.2 und höher.



Die Funktion berechnet für das IEC 60870-5-101 Protokoll die maximal verfügbare ASDU-Octetlänge anhand der konfigurierten APDU-Telegrammlänge und der Adressfeldlänge der Verbindungsschicht. Die maximal verfügbare ASDU-Länge wird z.B. bei der Konfiguration der ST_IEC870_5_101TBuffer-Variablen benötigt. Diese Datenstruktur (TX/RX-Datenpuffer) wird beim Datenaustausch über das IEC60870-5-101 Serial Link Interface benutzt.

FUNCTION F_iecApdu101ToAsduLen: BYTE

```
VAR_INPUT
    FRAMELength     : BYTE;
    eLinkAddrSize   : E_IEC870_5_101LinkAddrSize;
END_VAR
```

FRAMELength : Die max. verfügbare APDU-Telegrammlänge (siehe Kompatibilitätsliste).

eLinkAddrSize: Adressfeldlänge der Verbindungsschicht.

Voraussetzungen

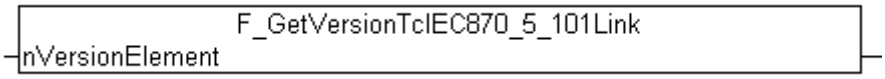
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_101Link.Lib
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; werden automatisch eingebunden)

Sehen Sie dazu auch

M_SP_NA_1 [▶ 37]

8.1.5 F_GetVersionTcIEC870_5_101Link



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_101Link: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.6 ST_IEC870_5_101ProtocolParams

Protokollparameter der Verbindungsschicht. Die Parameter haben abhängig von dem Protokoll- oder Stationstyp unterschiedliche Bedeutung.

```
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
```

In der unterstehenden Tabelle finden Sie Hinweise zur Konfiguration. Bei fixen Werten muss der Parameter dem Wert aus der Tabelle entsprechen (nicht konfigurierbar).

Legende:

- **X** Der Parameter wird benutzt und ist konfigurierbar;
- **N/A** Der Parameter wird nicht benutzt und ist nicht konfigurierbar;

Parameter-name	Initialisie-rungswert	IEC 60870-5-101 controlled station (slave, un-balanced mode)	IEC 60870-5-101 control sta-tion (master, un-balanced mode)	IEC 60870-5-103 control sta-tion (master, un-balanced mode)	IEC 60870-5-102 control sta-tion (master, un-balanced mode)	Beschreibung
eType	eIEC870_101_SLAVE	fix, eIEC870_101_SLAVE	fix, eIEC870_101_MASTER	fix, eIEC870_103_MASTER	fix, eIEC870_102_MASTER	Konfiguriert den Protokoll- und Stationstyp [▶ 412].
eLinkReset	eIEC870_LinkReset_CU	N/A	X	X	fix, eIEC870_LinkReset_CU	Reset-Typ [▶ 412] bei der Intilialisierung der Verbindungsschicht.
linkAddr	1	eigene Adresse	Adresse des Remote-Teilnehmers	Adresse des Remote-Teilnehmers	Adresse des Remote-Teilnehmers	Verbindungsadresse.
eLinkAddrSize	eIEC870_LinkAddr_TwoOctets	X	X	X	X	Octetgröße [▶ 330] der Verbindungsadresse.
eLinkMode	eIEC870_LinkMode_Unbalanced	fix, eIEC870_LinkMode_Unbalanced	fix, eIEC870_LinkMode_Unbalanced	fix, eIEC870_LinkMode_Unbalanced	fix, eIEC870_LinkMode_Unbalanced	Verbindungs-Mode [▶ 410] (balanced/ unbalanced).
eFrameType	eIEC870_FrameType_FT1_2	fix, eIEC870_FrameType_FT1_2	fix, eIEC870_FrameType_FT1_2	fix, eIEC870_FrameType_FT1_2	fix, eIEC870_FrameType_FT1_2	Telegramm-Frameformat [▶ 410]. Zur Zeit wird nur das Format F1.2 unterstützt.
tRxTimeout	T#5s	X	X	X	X	Max. Frame-Receive-Timeout-Zeit.
tTxTimeout	T#5s	X	X	X	X	Max. Frame-Send-Timeout-Zeit.
bForceC1Res	TRUE	X	N/A	N/A	N/A	Ist dieser Parameter gesetzt (TRUE), dann werden Class 1 Daten als Antwort auf Class 2 Anfrage gesendet wenn Class 1 Daten vorliegen (optimierter, höherer Datendurchsatz für Class 1).

Parameter-name	Initialisie-rungswert	IEC 60870-5-101 controlled station (slave, un-balanced mode)	IEC 60870-5-101 control sta-tion (master, un-balanced mode)	IEC 60870-5-103 control sta-tion (master, un-balanced mode)	IEC 60870-5-102 control sta-tion (master, un-balanced mode)	Beschreibung
						Ist dieser Parameter nicht gesetzt (FALSE) dann wird negative Antwort auf Class 2 Anfrage gesendet wenn Class 1 Daten vorliegen (kleinerer Datendurchsatz, leere Telegramme).
bForceC2Res	TRUE	X	N/A	N/A	N/A	Ist dieser Parameter gesetzt (TRUE), dann werden Class 2 Daten als Antwort auf Class 2 Anfrage gesendet obwohl Class 1 Daten vorliegen (optimierter, höherer Datendurchsatz für Class 2). Ist dieser Parameter nicht gesetzt (FALSE) dann wird negative Antwort auf Class 2 Anfrage gesendet wenn Class 1 Daten vorliegen (kleinerer Datendurchsatz, leere Telegramme)
tClass1Poll	T#200ms	N/A	X	X	X	Zykluszeit in der die Class 1 Daten abgefragt werden sollen (unbalanced mode only).
tClass2Poll	T#200ms	N/A	X	X	X	Zykluszeit in der die Class 2 Daten abgefragt werden sollen (unbalanced mode only).
nRetries	3	bis zur v3.0.6 nicht benutzt (Fehler beim ersten gestörten Frame), ab der v3.0.8 und höher konfigurierbar	X	X	X	Maximale Anzahl der Telegrammwiederholungen beim gestörten Datenaustausch.
tRetry	T#100ms	N/A	X	X	X	Verzögerungszeit für Telegrammwiederholungen beim gestörten Datenaustausch.

Parameter-name	Initialisie-rungswert	IEC 60870-5-101 controlled station (slave, un-balanced mode)	IEC 60870-5-101 control sta-tion (master, un-balanced mode)	IEC 60870-5-103 control sta-tion (master, un-balanced mode)	IEC 60870-5-102 control sta-tion (master, un-balanced mode)	Beschreibung
tResponse	T#1s	N/A	X	X	X	Dieser Parameter bestimmt die max. Wartezeit auf eine Telegrammbestätigung. Beim Überschreiten dieser Zeit wird das Telegramm wiederholt, aber max. bis zu *nRetry*-Male, übertragen.
tTestLink	T#5s	N/A	N/A	N/A	N/A	Zykluszeit in der Test-Telegramme verschickt werden sollen (balanced mode only)
tPollDFC	T#1s	N/A	X	X	X	Zykluszeit in der beim gesetzten DFC-Bit der Verbindungsstatus abgefragt werden soll. Das DFC-Bit wird von dem Remote-Teilnehmer dann gesetzt wenn weitere Telegramme einen Überlauf verursachen könnten.
FRAMELength	MAX_IEC870_5_101Link_FRAMELEN	X	X	X	X	Telegramlänge, max. Länge L.
bRetainBuffer	FALSE	ab der Produktversion v3.0.2 und höher	X	X	X	Ist dieser Parameter gesetzt, dann werden die internen TX/RX-Puffer bei der Initialisierung der Verbindungsschicht nicht zurückgesetzt (gelöscht).
tMaxPollDelay	T#0s (= 0: deaktiviert, <> 0: aktiv)	ab der Produktversion v3.0.2 und höher	N/A	N/A	N/A	Beim Fehlen der Poll-Anfragen wird nach der Überschreitung dieser Zeit der Verbindungsstatus als OFFLINE deklariert.
tLinkPollCycle	T#10s	N/A	X	X	X	Zykluszeit in der bei einer fehlenden Verbindung link status telegramme an die Unterstation gesendet werden.

Zusätzliche Hinweise zu **bForceC1Res** und **bForceC2Res**-Parametern:

Unabhängig davon ob Sie beide oder nur eines der Parameter setzen oder nicht wird das ACD-Bit auf jeden Fall passend gesetzt um der Leitstation mitzuteilen ob Class 1 oder Class 2 Daten als nächstes abgefragt werden sollen. Nur bei Classe 2 Abfragen wird das Verhalten der Unterstation von diesen Parametern beeinflusst.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

Sehen Sie dazu auch

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. Zur Zeit wird nur FT1.2 unterstützt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.8 E_IEC870_5_101LinkMode

Konfiguriert den Datenübertragungsmodus.

```
TYPE E_IEC870_5_101LinkMode :
(
    eIEC870_LinkMode_Unbalanced,
```

```
    eIEC870_LinkMode_Balanced
);
END_TYPE
```

eIEC870_LinkMode_Unbalanced: Asymmetrische/Unsymmetrische Datenübertragung. In diesem Übertragungsmodus steuert die Leitstation (master) den Datenaustausch mit den Unterstationen (slaves). Die Leitstation initiiert immer den Datenaustausch. Die Leitstation fragt auf diese Weise sequentiell alle Unterstationen ab.

eIEC870_LinkMode_Balanced: Symmetrische Datenübertraung. Dieser Übertragungsmodus ist nur in neueren Produktversionen verfügbar:

- TwinCAT PLC Library IEC 60870-5-101 Master v2.0.1 und höher;
- TwinCAT PLC Library IEC 60870-5-101 Slave v4.0.1 und höher;

In diesem Übertragunsmodus kann jede Station den Datenaustausch initieren und kann nur in "point-to-point" und "multiple-point-to-point"-Konfigurationen verwendet werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
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
```

Verbindungsstatus.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.1.10 E_IEC870_5_101PartyLineMode

Ab der Produktversion: TwinCAT PLC Library IEC870-5-101 Unterstation v2.0.2 und höher.

```
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 Aktivierungsmode. Siehe auch in der Beschreibung des Funktionsbausteins: FB_IEC870_PartyLineCtrl [▶ 398].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
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
```

Reset-Typ während der Initialisierung der Verbindungsschicht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

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
```

Protokoll- bzw. Stationstyp.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1307	PC or CX (x86, ARM)	TcIEC870_5_101Link.Lib

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; COMlibV2.Lib; werden automatisch eingebunden)

8.1.13 T_HSERIALCTRL

Eine Variable von diesem Typ repräsentiert ein Verbindungs-Handle der seriellen Schnittstelle. Die Strukturelemente sollen nicht direkt beschrieben oder verändert werden. Variablen von diesem Typ werden für den Datenaustausch der zu sendeten oder empfangenen Daten zwischen dem Funktionsbaustein FB_IEC870_SerialLineCtrl [▶ 403] und anderen IEC 60870-5-101 SPS-Bausteinen benutzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

8.2 Beispiele

Im Beispielprojekten sind folgende Stationsparameter für die Unterstation oder Zentralstation eingestellt:

- 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**

Um die Beispiele fehlerfrei übersetzen zu können benötigen Sie die TwinCAT Version 2.10 Build > 1328!

Die Beispielapplikationen beinhalten zwei Programmteile: P_MAIN_LowSpeed() (langsame Task) implementiert das Fernwirkprotokol und P_SerialComm_HighSpeed() (schnelle Task)implementiert die TwinCAT Kommunikation über die seriellen Schnittstellen.

SPS Project	TwinCAT System Manager Konfiguration	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700173963.zip (unbalanced mode)	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700173963.zip (unbalanced mode)	Einfache **Slave-Applikation** die das TwinCAT IEC60870-5-101 Serial Link Interface nutzt. Der Funktionsbaustein FB_IEC870_5_101SServices beinhaltet eine einfache Implementierung folgender

SPS Project	TwinCAT System Manager Konfiguration	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700175371.zip (balanced mode)	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700175371.zip (balanced mode)	Dienste: Test, Uhrzeitsynchronisation, Generalabfrage und Zählerabfrage (mode C). Bei einer hergestellten Verbindung sendet die Unterstation jede Sekunde einen **Bitstring (M_BO_TB_1, mit Zeitstempel CP56Time2a, Objektadresse: 100)** und der Übertragunsursache: Spontan an die Zentralstation. Ein **Zählwert (M_IT_NA_1, ohne Zeitstempel, Objektadresse: 400)** wird zur Leitstation als Antwort auf Zähler-Abfragebefehle (mode C).
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700176779.zip (unbalanced mode) https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700178187.zip (balanced mode)	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700176779.zip (unbalanced mode) https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11700178187.zip (balanced mode)	Einfache **Master-Applikation** die das TwinCAT IEC60870-5-101 Serial Link Interface nutzt. Der Funktionsbaustein FB_IEC870_5_101MServices beinhaltet eine einfache Implementierung folgender Dienste: Test, Uhrzeitsynchronisation, Generalabfrage und Zählerabfrage.

8.3 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE Image-Version usw.);
2. Vergleichen/überprüfen Sie die Kompatibilitätsliste der Leitstation mit der Kompatibilitätsliste der Unterstation;
3. Überprüfen Sie die IO-Konfiguration und das Mapping der SPS-Variablen in TwinCAT System Manager (Konfiguration der seriellen Schnittstellen [▶ 506], Baudrate, Parity, Stopbits usw.). Vergleichen Sie die Parameter mit Parametern des Kommunikationspartners;
4. Überprüfen Sie ob der FB_IEC870_5_101TProtocol [▶ 397]() Funktionsbaustein einen Fehlercode ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes;
5. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 406] (Link-Adresse, Länge der Link-Adresse, FRAMELength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern des Kommunikationspartners;
6. Überprüfen Sie die am TX/RX-Datenpuffer (instanz von ST_IEC870_5_101TBuffer [▶ 37]) konfigurierten Adresslängen: Länge der ASDU-Adresse, Länge der Informationsobjekt-Adresse, Länge der Übertragungsursache, max. ASDU-Länge. Vergleichen Sie die Adresslängen mit der Konfiguration des Kommunikationspartners;
7. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, Adresse des Informationsobjektes usw.);
8. Überprüfen Sie ob der andere Kommunikationspartner einen Fehlercode ausgibt;
9. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben;

9 TcIEC870_5_102Link: IEC 60870-5-102 Serial Link Interface (master)

Mit den SPS-Funktionen und -Funktionsbausteinen können Zentralstationen (Master) nach der IEC60870-5-102 Norm in TwinCAT SPS realisiert werden. Die IEC60870-5-102 Norm ist eine Anwendungsbezogene Norm für die Zählerstandsübertragung in der Elektrizitätsversorgung.

Die Endapplikation (Implementierung einer Zentralstation) wird auf der Software-Schnittstelle der SPS-Bibliothek aufgesetzt. Anders als bei den 101/104-TwinCAT Supplement Produkten steht dem SPS-Programmierer bei der IEC60870-5-102-Protokollimplementierung nur die sogenannte "Low level"-Schnittstelle zur Verfügung. Der Grund: Die meisten Unterstationen (Zählwerteinrichtungen) besitzen viele Herstellerspezifische oder Gerätespezifische Parameterdaten oder Informationen. Die "Low level"-Schnittstelle ermöglicht einen uneingeschränkten Zugriff auf all diese Daten. Im Folgenden werden die Eigenschaften dieser Schnittstelle kurz beschrieben.

"Low level"-Schnittstelle: IEC 60870-5-102 Serial Link Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können manipuliert werden (z.B. information object address, record address usw); Weil nur die benötigten Dienste (abhängig vom Gerät) implementiert werden kann eine hohe Performance erreicht werden; Hohe Performace bei vielen Daten bzw. Zählwerten;

Contra: Grösserer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Kompatibilitätsliste

für TwinCAT SPS Bibliothek: IEC 60870-5-102 Zentralstation. Hier können Sie die https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11763594379.zip

Systemvoraussetzungen

Programmierumgebung:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT Installation Level: TwinCAT PLC oder höher;
- TwinCAT System Version 2.10.0 Build >= 1319 oder höher;

Zielplattform:

- Industrie PC oder Embedded PC/CX (x86, ARM);
- Betriebssystem:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86, ARM) (image v2.16 oder höher);
- TwinCAT SPS-Laufzeitsystem Version 2.10.0 oder höher;
- Serielle COM Schnittstelle oder KL6xxx- oder EL6xxx-Klemmen (**EL6xxx Unterstützung ab der Produktversion 1.0.4 und höher**);

Produktkomponenten

- **TcIEC870_5_102Link.Lib** (implementiert Übertragungsprozeduren für den Transport der ASDUs nach IEC 60870-5-102 Norm); Diese Bibliothek muss in dem SPS-Projekt eingebunden werden. Alle anderen Bibliotheken werden automatisch hinzugefügt.

- TcIEC870_5_101Link.Lib (Basisbibliothek, implementiert Übertragungsprozeduren für den Transport der ASDUs über die seriellen Schnittstellen des PCs und die Beckhoff KL6xxx-/EL6xxx-Klemmen);
- TcIEC870_5_101.Lib (implementiert die Verbindungsfunktionen und gemeinsame Datentypen);
- COMLibV2.Lib (implementiert die Funktionen für die serielle COM- oder KL6xxx-/EL6xxx-Kommunikation);

Installation auf Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

Die SPS-Bibliotheken werden während der Installation auf der Programmierumgebung in den .. \TwinCAT\Plc\Lib-Ordner hineinkopiert:

Installation auf Windows CE

Auf der CE Plattform werden keine zusätzlichen Komponenten installiert.

Dies ist nur eine kurze Produktinformation (kein vollständiges Handbuch). Bitte installieren Sie sich die vollständige Ausgabe des Beckhoff Information System.

Sie finden es

- auf sämtlichen Beckhoff-Produkt-DVDs
- auf unserem Web-Server http://www.beckhoff.com unter Download.

Beispiele

Beispiele befinden sich in der Beckhoff Information System Dokumentation der SPS-Bibliotheken.

Link zu "Low level" Beispiel-Übersichtsseite: IEC 60870-5-102 Serial Link Interface [▶ 429];

Weiterführende Dokumentation

- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 396];
- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268];
- Dokumentation zur TwinCAT PLC Library: Serielle Kommunikation;
- 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;

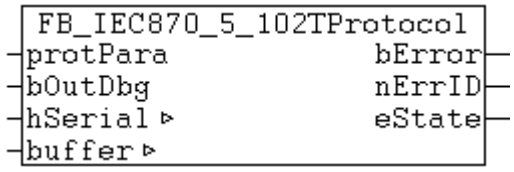
9.1 SPS-API

Einfache Projekte mit kompletten Sourcen finden Sie hier: Transport-Interface-Beispiele.Transport-Interface-Beispiele [▶ 429]

Eine SPS-Applikation, die über das Transport Interface mit einer Unterstation kommunizieren soll, benötigt folgende Resourcen:

- Eine Instanz des Kommunikationsbausteins: FB_IEC870_5_102TProtocol [▶ 418];
- Eine Instanz des TX/RX-Datenpuffers: ST_IEC870_5_102TBuffer [▶ 422];
- Eine Instanz des Funktionsbausteins zur Manipulation des TX/RX-Datenpuffers: FB_IEC870_5_102TBufferCtrl [▶ 419];

9.1.1 FB_IEC870_5_102TProtocol



Der Kommunikationsbaustein FB_IEC870_5_102TProtocol implementiert die Übertragungsprozeduren der Verbindungsschicht nach der IEC 60870-5-1 und IEC 60870-5-2-Norm.
Beim Protokollfehler wird ein entsprechender Fehlercode am Ausgang des Funktionsbausteins ausgegeben und die Datenübertragung unterbrochen. Um den Datenaustausch erneut aktivieren zu können, muss die Aktion INIT aufgerufen werden. Der Kommunikationsbaustein erwartet eine TX/RX-Datenpuffervariable. Diese Variable muss per VAR_IN_OUT an den Baustein übergeben werden.

Der Funktionsbaustein besitzt folgende Aktionen:

- **INIT** (Führt eine Initialisierung des Funktionsbausteins durch). In der Default-Konfiguration werden die TX/RX-Datenpuffer zurückgesetzt. Das Löschen der Puffer kann aber durch das Setzen der bRetainBuffer-Variablen in der Protokolparameterstruktur unterbunden werden.

Protokollkonfiguration

Der Kommunikationsbaustein besitzt eine protPara-Variable vom strukturierten Typ. Über diese Variable können Protokollparameter z.B. RX/TX-Timeoutzeiten usw. konfiguriert werden.

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: Verbindungs-Handle [▶ 413] zum FB_IEC870_SerialLineCtrl [▶ 403]-Funktionsbaustein. Über diese Variable werden mit dem FB_IEC870_SerialLineCtrl-Funktionsbaustein die zu sendenden und empfangenen Daten ausgetauscht.

buffer: TX/RX Datenpuffer [▶ 422].

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-Protokollparameter [▶ 406].

bOutDbg:: Aktiviert/deaktiviert die Debug-Ausgabe der Frames in der TwinCAT System Manager-Loggeransicht.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED; (* Data transfer state
*)
ND_VAR
```

bError: Dieser Ausgang wird auf TRUE gesetzt, sobald ein Fehler bei der Datenübertragung aufgetreten ist.

nErrID: Liefert bei einem gesetzten bError-Ausgang einen Fehlercode;

eState: Verbindungsstatus [▶ 411].

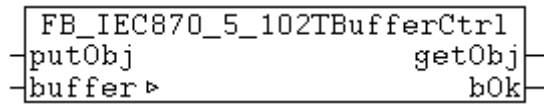
Beispiel:

Beispielprojekte: IEC60870-5-102 Serial Link Interface [▶ 429]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1319	PC oder 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; werden automatisch eingebunden)

9.1.2 FB_IEC870_5_102TBufferCtrl



Mit diesem Funktionsbaustein kann der Inhalt des TX/RX-Datenpuffers manipuliert werden, der bei der Kommunikation über das IEC60870-5-102 Serial Link Interface benutzt wird.

Der Funktionsbaustein besitzt folgende Aktionen:

- **RxRemoveObj** (entfernt den ältesten Fifoeintrag aus dem RX-Fifo);
- **RxReset** (löscht alle RX-Fifoeinträge, setzt den RX-Fifo zurück);
- **TxAddObj** (fügt einen neuen Fifoeintrag in den TX-Fifo);
- **TxReset** (löscht alle TX-Fifoeinträge, setzt den TX-Fifo zurück)

Durch den Aufruf der oben aufgelisteten Aktionen kann der Inhalt des TX/RX-Datenpuffers verändert werden.

VAR_IN_OUT

```
VAR_IN_OUT
    buffer  : ST_IEC870_5_102TBuffer; (* TX/RX data buffer *)
END_VAR
```

buffer: TX/RX-Datenpuffer. Die TX/RX-Pufferparameter (wie z.B. asduSize) müssen vor der Benutzung konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    putObj  : ST_IEC870_5_102AOGen; (* ASDU to send *)
END_VAR
```

putObj: Dateneinheit (ASDU), die gesendet werden soll.

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: Empfangene Dateneinheit (ASDU).

bOk: Diese Variable wird TRUE, wenn ein neuer Eintrag erfolgreich hinzugefügt oder aus dem Fifo entfernt wurde. Diese Variable wird FALSE beim Pufferüberlauf und wenn kein Eintrag entfernt werden konnte, weil der Fifo bereits leer ist.

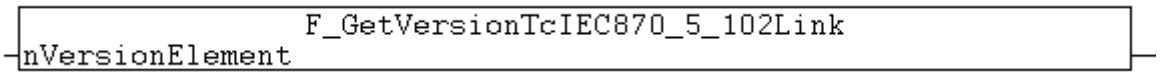
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1319	PC oder 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; werden automatisch eingebunden)

Sehen Sie dazu auch

- ST_IEC870_5_102TBuffer [▶ 422]
- ST_IEC870_5_102AOGen [▶ 423]
- ST_IEC870_5_102AOGen [▶ 423]

9.1.3 F_GetVersionTcIEC870_5_102Link



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_102Link: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

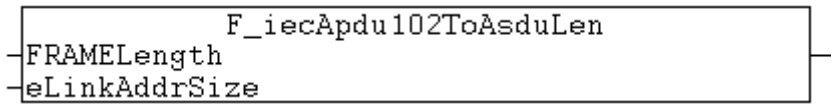
- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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;

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		TcIEC870_5_101Link.Lib; COMlibV2.Lib; werden automatisch eingebunden)

9.1.4 F_iecApdu102ToAsduLen



Die Funktion berechnet für das IEC 60870-5-102 Protokoll die maximal verfügbare ASDU-Octetlänge anhand der konfigurierten APDU-Telegrammlänge und der Adressfeldlänge der Verbindungsschicht. Die maximal verfügbare ASDU-Länge wird z.B. bei der Konfiguration der ST_IEC870_5_102TBuffer-Variablen benötigt. Diese Datenstruktur (TX/RX-Datenpuffer) wird beim Datenaustausch über das IEC60870-5-102 Serial Link Interface benutzt.

FUNCTION F_iecApdu102ToAsduLen: BYTE

```
VAR_INPUT
    FRAMELength     : BYTE;
    eLinkAddrSize   : E_IEC870_5_101LinkAddrSize;
END_VAR
```

FRAMELength : Die max. verfügbare APDU-Telegrammlänge (siehe Kompatibilitätsliste).

eLinkAddrSize: Adressfeldlänge der Verbindungsschicht.

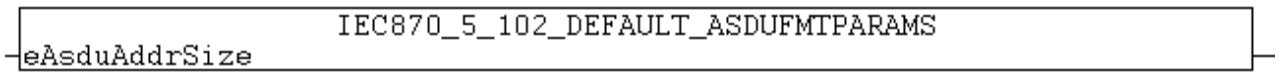
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

Sehen Sie dazu auch

M_IT_TA_2 [▶ 212]

9.1.5 IEC870_5_102_DEFAULT_ASDUFMTPARAMS



Diese Funktion setzt die durch die 102 Norm vorgegebenen Frame-Formatparameter: COT, ASDUADDR, IOA-Länge.

FUNCTION IEC870_5_102_DEFAULT_ASDUFMTPARAMS: ST_IEC870_5_101AsduFmtParams

ST_IEC870_5_101AsduFmtParams [▶ 212]

```
VAR_INPUT
    eAsduAddrSize : E_IEC870_5_101AsduAddrSize;
END_VAR
```

eAsduAddrSize : Oktetlänge der gemeinsamen ASDU-Addresse.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

9.1.6 ST_IEC870_5_102TBuffer

Diese Datenstruktur wird beim Datenaustausch (TX/RX-Datenpuffer) über das IEC60870-5-102 Serial Link Interface benutzt.

```
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-Ausgabe-Parameter.

eAsduAddrSize: Oktetlänge der gemeinsamen ASDU addresse.

asduSize: Maximale Bytelänge der ASDU.

mode: Reserviert, wird zurzeit nicht benutzt. Dieser Wert sollte Null sein.

dataLink: Auf die Elemente dieser Datenstruktur sollte nicht direkt, sondern nur mit einer Instanz des FB_IEC870_5_102TBufferCtrl [▶ 419]-Funktionsbausteins zugegriffen werden.

Der TX/RX-Datenpuffer verwendet intern zwei Sende-Fifos und einen Empfangs-Fifo:

1. Class 1 Sende-Fifo mit (hochprioren) Daten;
2. Class 2 Sende-Fifo mit (niederprioren) Daten;
3. Receive-Fifo (für Class 1 und Class 2 Daten);

Die unteren Transportfunktionen der Bibliothek leeren zuerst den Class 1-Fifo und dann den Class 2-Fifo. Die Class 2 Daten werden nur dann versendet, wenn der Class 1-Fifo keine zu versendenden Daten enthält.

Jeder der internen Fifos hat eine feste Größe von 10000 Bytes. Erfahrungsgemäß können in jedem Fifo ca. 200 ASDUs mit einem Informations-Element (Objekt) mit der maximalen Grösse oder ca. 20 ASDUs mit einer Sequence von 100 Informations-Elementen (Objekten) abgelegt werden.

Wenn eine größere Anzahl der zu versendenden oder zu empfangenen Frames zwischengespeichert werden soll (z.B. ~20000), so können diese in externen, vom SPS-Programmierer festgelegten Puffern/Fifos zwischengehalten werden. Die SPS-Applikation kann dann zur Laufzeit die TwinCAT-Sende-Fifos mit den eigenen Fifo-Einträgen nachfüllen oder bei vielen empfangenen Frames den TwinCAT-Receive-Fifo leeren. Eine andere Möglichkeit ist z.B. zwei Puffer zu benutzen und diese abwechselnd zu füllen/lesen und an den Kommunikationsbaustein übergeben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1319	PC oder CX (x86, ARM)	TcIEC870_5_102Link.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib;

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
		TcIEC870_5_101.Lib; TcIEC870_5_101Link.Lib; COMLibV2.Lib werden automatisch eingebunden)

Sehen Sie dazu auch

- M_IT_TA_2 [▶ 212]

9.1.7 ST_IEC870_5_102AOGen

Variablen von diesem Typ repräsentieren ein ASDU-Objekt.

```
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 (reserviert).

ident:Identifikationsfelder der Dateneinheit (ASDU).

info: Informationsobjekt-/Informationselement-Datenfeld.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1319	PC oder 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 werden automatisch eingebunden)

Sehen Sie dazu auch

- ST_IEC870_5_102DataUnit_Ident [▶ 424]
- ST_IEC870_5_102AOInfoObj [▶ 423]

9.1.8 ST_IEC870_5_102AOInfoObj

Informationsobjektbeschreibung.

```
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: Listenadresse einer Messperiode oder Listenadresse einer Einzelmeldung.

stream: Informationselemente oder Informationsobjekte (Byte->Array-Puffer).

Die Listenadressen sind wie folgt festgelegt: <0..255>

- <0> Default, wenn kein anderer Wert festgelegt ist
- <1> Listenadresse für Zählerstände vom Beginn der Abrechnungsperiode
- <2..10> Reserviert für weitere kompatible Festlegungen
- <11> Listenadresse für Zählerstände der Messperiode 1
- <12> Listenadresse für Zählerstände der Messperiode 2
- <13> Listenadresse für Zählerstände der Messperiode 3
- <14..20> Reserviert für weitere kompatible Festlegungen
- <21> Listenadresse für Zählerstände (Tageswerte) der Messperiode 1
- <22> Listenadresse für Zählerstände (Tageswerte) der Messperiode 2
- <23> Listenadresse für Zählerstände (Tageswerte) der Messperiode 3
- <24..30> Reserviert für weitere kompatible Festlegungen
- <31> Listenadresse für Zählerstände (Monatswerte) der Messperiode 1
- <32> Listenadresse für Zählerstände (Monatswerte) der Messperiode 2
- <33> Listenadresse für Zählerstände (Monatswerte) der Messperiode 3
- <34..40> Reserviert für weitere kompatible Festlegungen
- <41> Listenadresse für Zählerstände (Jahreswerte) der Messperiode 1
- <42> Listenadresse für Zählerstände (Jahreswerte) der Messperiode 2
- <43> Listenadresse für Zählerstände (Jahreswerte) der Messperiode 3
- <44..49> Reserviert für weitere kompatible Festlegungen
- <50> Älteste Einzelmeldung
- <51> Vollständige Liste mit Einzelmeldungen
- <52> Teilliste 1 mit Einzelmeldungen
- <53> Teilliste 2 mit Einzelmeldungen
- <54> Teilliste 3 mit Einzelmeldungen
- <55> Teilliste 4 mit Einzelmeldungen
- <56..127> Reserviert für weitere kompatible Festlegungen
- <128..255> Für besondere Anwendungen (privater Bereich)

Die Größe einer Teilliste ist ein Systemparameter.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1319	PC oder 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 werden automatisch eingebunden)

Sehen Sie dazu auch

M_IT_TA_2 [▶ 212]

9.1.9 ST_IEC870_5_102DataUnit_Ident

Identifikationsfeld der Dateneinheit (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-Typ. Zulässiger Bereich: <0..255>.

bSQ: Sequence-Flag.

nObj: Anzahl der Informationsobjekte oder Informationselemente. Zulässiger Bereich: <0..127>.

bT: Test-Bit.

bPN: FALSE = Positive, TRUE = Negative Bestätigung.

eCOT: Übertragungsursache. Zulässiger Bereich: <0..63>.

asduAddr: Gemeinsame ASDU-Adresse (one or two octets).

eClass: Prioritätsklasse.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1319	PC oder 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 werden automatisch eingebunden)

Sehen Sie dazu auch

- E_IEC870_5_102TypeID [▶ 425]
- E_IEC870_5_102COTType [▶ 427]
- M_IT_TA_2 [▶ 212]

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 and o
f a selected range of addresses *)
 C_CI_NC_2 := 106, (* (106) Read accounting integrated totals of a specific past integration period
*)
 C_CI_ND_2 := 107, (* (107) Read accounting integrated totals of a specific past integration period
and of a selected range of addresses *)
 C_CI_NE_2 := 108, (* (108) Read periodical reset accounting integrated totals of the oldest integra
tion period *)
 C_CI_NF_2 := 109, (* (109) Read periodical reset accounting integrated totals of the oldest integra
tion period and of a selected range of addresses *)
 C_CI_NG_2 := 110, (* (110) Read periodical reset accounting integrated totals of a specific past in
tegration period *)
 C_CI_NH_2 := 111, (* (111) Read periodical reset accounting integrated totals of a specific past in
tegration 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 and
of a selected range of addresses *)
 C_CI_NL_2 := 114, (* (114) Read operational integrated totals of a specific past integration period
 *)
 C_CI_NM_2 := 115, (* (115) Read operational integrated totals of a specific past integration period
 and of a selected range of addresses *)
 C_CI_NN_2 := 116, (* (116) Read periodical reset operational integrated totals of the oldest integr
ation period *)
 C_CI_NO_2 := 117, (* (117) Read periodical reset operational integrated totals of the oldest integr
ation period and of a selected range of addresses *)
 C_CI_NP_2 := 118, (* (118) Read periodical reset operational integrated totals of a specific past i
ntegration period *)
 C_CI_NQ_2 := 119, (* (119) Read periodical reset operational integrated totals of a specific past i
ntegration period and of a selected range of addresses *)
 C_CI_NR_2 := 120, (* (120) 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 := 121, (* (121) Read periodical reset accounting integrated totals of a specific past in
tegration 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 period
 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 past i
ntegration 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-Typkennungen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

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
```

Übertragungsursachen nach der IEC 60870-5-102 Norm.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

9.1.12 T_CP40Time2a

Binäres (5 Byte) Zeitformat .

```
TYPE T_CP40Time2a :
STRUCT
    IVTisMinute : BYTE := 16#00; (* Bit 7 = IV (1=invalid time, 0=valid time), Bit 6 = Tarif informa
tion, Bit 0..5 = Minutes <0..59> *)
    SURes1Hour  : BYTE := 16#00; (* Bit 7 = SU (1=summer time, 0=normal time), Bits 5..6 = spare bit
s <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 Energie ta
rif 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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1319	PC oder 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; werden automatisch eingebunden)

9.1.13 T_CP56Time2b

Binäres (7 Byte) Zeitformat.

```
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 informa
tion, Bit 0..5 = Minutes <0..59> *)
    SURes1Hour  : BYTE := 16#00; (* Bit 7 = SU (1=summer time, 0=normal time), Bits 5..6 = spare bit
s <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 Energie ta
rif 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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1319	PC oder 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 werden automatisch eingebunden)

9.2 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE Image-Version usw.);
2. Vergleichen/überprüfen Sie die Kompatibilitätsliste der Leitstation mit der Kompatibilitätsliste der Unterstation;
3. Überprüfen Sie die IO-Konfiguration und das Mapping der SPS-Variablen in TwinCAT System Manager (Konfiguration der seriellen Schnittstellen, Baudrate, Parity, Stopbits usw.). Vergleichen Sie die Parameter mit Parametern der Unterstation;
4. Überprüfen Sie ob der FB_IEC870_5_102TProtocol [▶ 418]() Funktionsbaustein einen Fehlercode ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes;
5. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 406] (Link-Adresse, FRAMELength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern der Unterstation;
6. Überprüfen Sie die am TX/RX-Datenpuffer (instanz von ST_IEC870_5_102TBuffer [▶ 422]) konfigurierten Adresslängen: Länge der ASDU-Adresse, max. ASDU-Länge. Vergleichen Sie die Adresslängen mit der Konfiguration der Unterstation;
7. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, Adresse des Informationsobjektes usw.);
8. Überprüfen Sie ob die Unterstation einen Fehlercode ausgibt;
9. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben;

9.3 Beispiele

In den Beispielprojekten sind folgende Stationsparameter für die Zentralstation eingestellt:

- Standard PC interface: **COM1**
- **19200 Baud**
- Link address: **1**
- Link address size: **2 octets**
- Common ASDU address: **1**
- Common ASDU address size: **2 octets**

Voraussetzungen

SPS Project	TwinCAT System Manager Konfiguration	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11717569675.zip	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11717569675.zip	Einfache **Master-Applikation** die das TwinCAT IEC60870-5-102 Serial Link Interface nutzt.

10 TcIEC870_5_103Link: IEC 60870-5-103 Serial Link Interface (master)

Mit den SPS-Funktionen und -Funktionsbausteinen können Zentralstationen (Master) nach der IEC60870-5-103 Norm in TwinCAT SPS realisiert werden.

Die Endapplikation (Implementierung einer Zentralstation) wird auf der Software-Schnittstelle der SPS-Bibliothek aufgesetzt. Anders als bei den 101/104-TwinCAT Supplement Produkten steht dem SPS-Programmierer bei der IEC60870-5-103-Protokollimplementierung nur die sogenannte "Low level"-Schnittstelle zur Verfügung. Der Grund: Die meisten Unterstationen (Schutzgeräte) besitzen viele Herstellerspezifische oder Gerätespezifische Parameterdaten oder Informationen. Die "Low level"-Schnittstelle ermöglicht einen uneingeschränkten Zugriff auf all diese Daten. Im Folgenden werden die Eigenschaften dieser Schnittstelle kurz beschrieben.

"Low level"-Schnittstelle: IEC 60870-5-103 Serial Link Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können manipuliert werden (z.B. Funktionscodes, Informationsnummern usw); Weil nur die benötigten Dienste (abhängig vom Gerät) implementiert werden kann eine hohe Performance erreicht werden; Hohe Performace bei vielen Datenpunkten oder einem hohem Datenaufkommen;

Contra: Grösserer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Kompatibilitätsliste

für TwinCAT SPS Bibliothek: IEC 60870-5-103 Zentralstation. Hier können Sie die https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11763843211.zip

Systemvoraussetzungen

Programmierumgebung:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT Installation Level: TwinCAT PLC oder höher;
- TwinCAT System Version 2.10.0 Build >= 1313 oder höher;

Zielplattform:

- Industrie PC oder Embedded PC/CX (x86, ARM);
- Betriebssystem:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86, ARM) (image v2.16 oder höher);
- TwinCAT SPS-Laufzeitsystem Version 2.10.0 oder höher;
- Serielle COM Schnittstelle oder KL6xxx- oder EL6xxx-Klemmen (**EL6xxx Unterstützung ab der Produktversion 1.0.11 und höher**);

Produktkomponenten

- **TcIEC870_5_103Link.Lib** (implementiert Übertragungsprozeduren für den Transport der ASDUs nach IEC 60870-5-103 Norm); Diese Bibliothek muss in dem SPS-Projekt eingebunden werden. Alle anderen Bibliotheken werden automatisch hinzugefügt.

- TcIEC870_5_101Link.Lib (Basisbibliothek, implementiert Übertragungsprozeduren für den Transport der ASDUs über die seriellen Schnittstellen des PCs und die Beckhoff KL6xxx-/EL6xxx-Klemmen);
- TcIEC870_5_101.Lib (implementiert die Verbindungsfunktionen und gemeinsame Datentypen);
- COMLibV2.Lib (implementiert die Funktionen für die serielle COM- oder KL6xxx-/EL6xxx-Kommunikation);

Installation auf Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

Die SPS-Bibliotheken werden während der Installation auf der Programmierumgebung in den .. \TwinCAT\Plc\Lib-Ordner hineinkopiert:

Installation auf Windows CE

Auf der CE Plattform werden keine zusätzlichen Komponenten installiert.

Dies ist nur eine kurze Produktinformation (kein vollständiges Handbuch). Bitte installieren Sie sich die vollständige Ausgabe des Beckhoff Information System.

Sie finden es

- auf sämtlichen Beckhoff-Produkt-DVDs
- auf unserem Web-Server http://www.beckhoff.com unter Download.

Beispiele

Beispiele befinden sich in der Beckhoff Information System Dokumentation der SPS-Bibliotheken.

Link zu "Low level" Beispiel-Übersichtsseite: IEC 60870-5-103 Serial Link Interface [▶ 441];

Weiterführende Dokumentation

- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Serial Link Interface [▶ 396];
- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268];
- Dokumentation zur TwinCAT PLC Library: Serielle Kommunikation;
- 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;

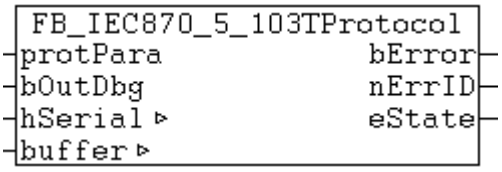
10.1 SPS-API

Einfache Projekte mit kompletten Sourcen finden Sie hier: Transport-Interface-Beispiele.Transport-Interface-Beispiele [▶ 441]

Eine SPS-Applikation, die über das Transport Interface mit einer Unterstation kommunizieren soll, benötigt folgende Resourcen:

- Eine Instanz des Kommunikationsbausteins: FB_IEC870_5_103TProtocol [▶ 433];
- Eine Instanz des TX/RX-Datenpuffers: ST_IEC870_5_103TBuffer [▶ 436];
- Eine Instanz des Funktionsbausteins zur Manipulation des TX/RX-Datenpuffers: FB_IEC870_5_103TBufferCtrl [▶ 434];

10.1.1 FB_IEC870_5_103TProtocol



Der Kommunikationsbaustein FB_IEC870_5_103TProtocol implementiert die Übertragungsprozeduren der Verbindungsschicht nach der IEC 60870-5-1 und IEC 60870-5-2-Norm.
Beim Protokollfehler wird ein entsprechender Fehlercode am Ausgang des Funktionsbausteins ausgegeben und die Datenübertragung unterbrochen. Um den Datenaustausch erneut aktivieren zu können, muss die Aktion INIT aufgerufen werden. Der Kommunikationsbaustein erwartet eine TX/RX-Datenpuffervariable. Diese Variable muss per VAR_IN_OUT an den Baustein übergeben werden.

Der Funktionsbaustein besitzt folgende Aktionen:

- **INIT** (Führt eine Initialisierung des Funktionsbausteins durch). In der Default-Konfiguration werden die TX/RX-Datenpuffer zurückgesetzt. Das Löschen der Puffer kann aber durch das Setzen der bRetainBuffer-Variablen in der Protokolparameterstruktur unterbunden werden.

Protokollkonfiguration

Der Kommunikationsbaustein besitzt eine protPara-Variable vom strukturierten Typ. Über diese Variable können Protokollparameter z.B. RX/TX-Timeoutzeiten usw. konfiguriert werden.

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: Verbindungs-Handle [▶ 413] zum FB_IEC870_SerialLineCtrl [▶ 403]-Funktionsbaustein. Über diese Variable werden mit dem FB_IEC870_SerialLineCtrl-Funktionsbaustein die zu sendenden und empfangenen Daten ausgetauscht.

buffer: TX/RX Datenpuffer [▶ 436].

VAR_INPUT

```
VAR_INPUT
    protPara    : ST_IEC870_5_101ProtocolParam; (* IEC60870-5-101/103 protocol parameters *)
    bOutDbg     : BOOL;       (* Enable/disable debug output *)
END_VAR
```

protPara: IEC60870-5-101/103-Protokollparameter [▶ 406].

bOutDbg:: Aktiviert/deaktiviert die Debug-Ausgabe der Frames in der TwinCAT System Manager-Loggeransicht.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_101SerialLinkState := eSERIALLINK_DISCONNECTED; (* Data transfer state
*)
ND_VAR
```

bError: Dieser Ausgang wird auf TRUE gesetzt, sobald ein Fehler bei der Datenübertragung aufgetreten ist.

nErrID: Liefert bei einem gesetzten bError-Ausgang einen Fehlercode;

eState: Verbindungsstatus [▶ 411].

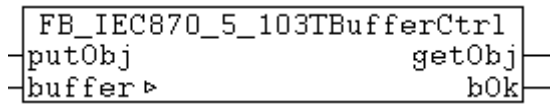
Beispiel:

Beispielprojekte: IEC60870-5-103 Serial Link Interface [▶ 441]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1313	PC oder 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; werden automatisch eingebunden)

10.1.2 FB_IEC870_5_103TBufferCtrl



Mit diesem Funktionsbaustein kann der Inhalt des TX/RX-Datenpuffers manipuliert werden, der bei der Kommunikation über das IEC60870-5-103 Serial Link Interface benutzt wird.

Der Funktionsbaustein besitzt folgende Aktionen:

- **RxRemoveObj** (entfernt den ältesten Fifoeintrag aus dem RX-Fifo);
- **RxReset** (löscht alle RX-Fifoeinträge, setzt den RX-Fifo zurück);
- **TxAddObj** (fügt einen neuen Fifoeintrag in den TX-Fifo);
- **TxReset** (löscht alle TX-Fifoeinträge, setzt den TX-Fifo zurück)

Durch den Aufruf der oben aufgelisteten Aktionen kann der Inhalt des TX/RX-Datenpuffers verändert werden.

VAR_IN_OUT

```
VAR_IN_OUT
    buffer  : ST_IEC870_5_103TBuffer; (* TX/RX data buffer *)
END_VAR
```

buffer: TX/RX-Datenpuffer [▶ 436]. Die TX/RX-Pufferparameter (wie z.B. asduSize) müssen vor der Benutzung konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    putObj   : ST_IEC870_5_103AOGen; (* ASDU to send *)
END_VAR
```

putObj: Dateneinheit [▶ 437] (ASDU), die gesendet werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    getObj   : ST_IEC870_5_103AOGen; (* received ASDU *)
    bOk      : BOOL; (* TRUE = action succesfully, FALSE=Fifo overflow or fifo empty *)
END_VAR
```

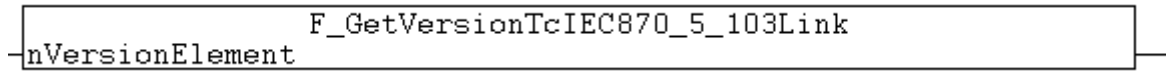
getObj: Empfangene Dateneinheit [▶ 437] (ASDU).

bOk: Diese Variable wird TRUE, wenn ein neuer Eintrag erfolgreich hinzugefügt oder aus dem Fifo entfernt wurde. Diese Variable wird FALSE beim Pufferüberlauf und wenn kein Eintrag entfernt werden konnte weil der Fifo bereits leer ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1313	PC oder 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; werden automatisch eingebunden)

10.1.3 F_GetVersionTcIEC870_5_103Link



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_103Link: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

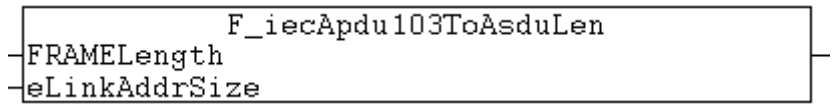
nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 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; werden automatisch eingebunden)

10.1.4 F_iecApdu103ToAsduLen



Die Funktion berechnet für das IEC 60870-5-103 Protokoll die maximal verfügbare ASDU-Octetlänge anhand der konfigurierten APDU-Telegrammlänge und der Adressfeldlänge der Verbindungsschicht. Die maximal verfügbare ASDU-Länge wird z.B. bei der Konfiguration der ST_IEC870_5_103TBuffer-Variablen benötigt. Diese Datenstruktur (TX/RX-Datenpuffer) wird beim Datenaustausch über das IEC60870-5-103 Serial Link Interface benutzt.

FUNCTION F_iecApdu103ToAsduLen: BYTE

```
VAR_INPUT
    FRAMELength     : BYTE;
    eLinkAddrSize   : E_IEC870_5_101LinkAddrSize;
END_VAR
```

FRAMELength : Die max. verfügbare APDU-Telegrammlänge (siehe Kompatibilitätsliste).

eLinkAddrSize: Adressfeldlänge [▶ 330] der Verbindungsschicht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 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; werden automatisch eingebunden)

10.1.5 ST_IEC870_5_103TBuffer

Diese Datenstruktur wird beim Datenaustausch (TX/RX-Datenpuffer) über das IEC60870-5-103 Serial Link Interface benutzt.

```
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-Ausgabe-Parameter [▶ 334].

asduSize: Maximale Bytelänge der ASDU.

mode: Reserviert, wird zurzeit nicht benutzt. Dieser Wert sollte Null sein.

dataLink: Auf die Elemente dieser Datenstruktur sollte nicht direkt, sondern nur mit einer Instanz des FB_IEC870_5_103TBufferCtrl [▶ 434]-Funktionsbausteins zugegriffen werden.

Der TX/RX-Datenpuffer verwendet intern zwei Sende-Fifos und einen Empfangs-Fifo:

1. Class 1 Sende-Fifo mit (hochprioren) Daten;
2. Class 2 Sende-Fifo mit (niederprioren) Daten;
3. Receive-Fifo (für Class 1 und Class 2 Daten);

Die unteren Transportfunktionen der Bibliothek leeren zuerst den Class 1-Fifo und dann den Class 2-Fifo. Die Class 2 Daten werden nur dann versendet, wenn der Class 1-Fifo keine zu versendenden Daten enthält.

Jeder der internen Fifos hat eine feste Größe von 10000 Bytes. Erfahrungsgemäß können in jedem Fifo ca. 200 ASDUs mit einem Informations-Element (Objekt) mit der maximalen Grösse oder ca. 20 ASDUs mit einer Sequence von 100 Informations-Elementen (Objekten) abgelegt werden.

Wenn eine größere Anzahl der zu versendenden oder zu empfangenen Frames zwischengespeichert werden soll (z.B. ~20000), so können diese in externen, vom SPS-Programmierer festgelegten Puffern/Fifos zwischengehalten werden. Die SPS-Applikation kann dann zur Laufzeit die TwinCAT-Sende-Fifos mit den eigenen Fifo-Einträgen nachfüllen oder bei vielen empfangenen Frames den TwinCAT-Receive-Fifo leeren. Eine andere Möglichkeit ist z.B. zwei Puffer zu benutzen und diese abwechselnd zu füllen/lesen und an den Kommunikationsbaustein übergeben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1313	PC oder 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 werden automatisch eingebunden)

10.1.6 ST_IEC870_5_103AOGen

Variablen von diesem Typ repräsentieren ein ASDU-Objekt.

```
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 (reserviert).

ident:Identifikationsfelder [▶ 438] der Dateneinheit (ASDU).

info: Informationsobjekt [▶ 437]-/Informationselement-Datenfeld.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1313	PC oder 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 werden automatisch eingebunden)

10.1.7 ST_IEC870_5_103AOInfoObj

Informationsobjektbeschreibung.

```
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: Funktionsnummer.

n: Informationsnummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1313	PC oder 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 werden automatisch eingebunden)

10.1.8 ST_IEC870_5_103DataUnit_Ident

Identifikationsfeld der Dateneinheit (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-Typ. Zulässiger Bereich: 0..255.

bSQ: Sequence-Flag.

nObj: Anzahl der Informationsobjekte oder Informationselemente. Zulässiger Bereich: 0..127.

eCOT: Übertragungsursache. Zulässiger Bereich: 0..255.

asduAddr: Gemeinsame ASDU-Adresse (one octet).

eClass: Prioritätsklasse [▶ 332].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build >= 1313	PC oder 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 werden automatisch eingebunden)

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-Typkennungen in Überwachungsrichtung (slave -> master).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

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-Typkennungen in Steuerungsrichtung (master -> slave).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

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
```

Übertragungsursachen in Überwachungsrichtung (slave -> master).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

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
```

Übertragungsursachen in Steuerungsrichtung (master -> slave).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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; werden automatisch eingebunden)

10.2 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE Image-Version usw.);
2. Vergleichen/überprüfen Sie die Kompatibilitätsliste der Leitstation mit der Kompatibilitätsliste der Unterstation;

3. Überprüfen Sie die IO-Konfiguration und das Mapping der SPS-Variablen in TwinCAT System Manager (Konfiguration der seriellen Schnittstellen, Baudrate, Parity, Stopbits usw.). Vergleichen Sie die Parameter mit den Parametern der Unterstation;
4. Überprüfen Sie ob der FB_IEC870_5_103TProtocol [▶ 433]() Funktionsbaustein einen Fehlercode ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes;
5. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 406] (Link-Adresse, Länge der Link-Adresse, FRAMELength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern der Unterstation;
6. Überprüfen Sie die am TX/RX-Datenpuffer (instanz von ST_IEC870_5_103TBuffer [▶ 436]) konfigurierte max. ASDU-Länge. Vergleichen Sie die Länge mit der Konfiguration der Unterstation;
7. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, ASDU-Adresse, Funktionsnummer, Informationsnummer usw.);
8. Überprüfen Sie ob der andere Kommunikationspartner einen Fehlercode ausgibt;
9. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben;

10.3 Beispiele

In den Beispielprojekten sind folgende Stationsparameter für die Zentralstation eingestellt:

- 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)**

Voraussetzungen

SPS Project	TwinCAT System Manager Konfiguration	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11724892683.zip	https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11724892683.zip	Einfache **Master-Applikation** die das TwinCAT IEC60870-5-103 Serial Link Interface nutzt.

11 TcIEC870_5_104: IEC 60870-5-104 Transport Interface (master/slave)

Die Bibliothek **TcIEC870_5_104.Lib** implementiert eine Schnittstelle über die einzelne ASDUs (Dienstdateneinheiten der Anwendungsschicht) verschickt und empfangen werden können. Innerhalb der Protokollstruktur liegt diese Schnittstelle oberhalb der Transportschicht (4) und implementiert bereits die APCI-Funktionen (Protokollsteuerinformationen der Anwendungsschicht, siehe untere Tabelle). Anwendungsfunktionen wie z.B. Generalabfrage und Zählerabfrage sind in der Schnittstelle nicht implementiert, der Anwender kann aber mit Hilfe der Schnittstelle diese Anwendungsfunktionen selber implementieren.

"Low level"-Schnittstelle: IEC 60870-5-104 Transport Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können manipuliert werden (z.B. ein eigener Zeitstempel, Select/Execute oder eine besondere Befehlsausführung usw); Weil nur die benötigten Dienste implementiert werden kann eine hohe Performance erreicht werden; Hohe Performace bei vielen Datenpunkten;

Contra: Grösserer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Diese Schnittstelle empfielt sich wenn Sie:

- Mit der Protokollnorm vertraut sind;
- Eine Protokollkonverter-Applikation implementieren möchten;
- In der Applikation die verfügbaren Norm-Funktionalitäten fast vollständig implementieren müssen;
- Besondere Funktionalitäten wie z.B. das weiterleiten der Zeitstempel von einem Modbusgerät oder die Kontrolle über die Befehlsausführung haben möchten;
- Funktionalitäten benötigen die laut Kompatibilitätsliste nicht unterstützt werden;
- Viele Datenpunte haben (>1000) und eine hohe Performace benötigen;

Protokollstruktur des Endsystems:

Voraussetzungen

Auswahl von Anwendungsfunktionen aus IEC 60870-5-5 nach IEC 60870-5-101	Initialisierung	Anwenderprozess
Auswahl von ASDU (Dienstdateneinheiten der Anwendungsschicht) aus IEC 60870-5-101 und IEC 60870-5-104		Anwendungsschicht (7)
TwinCAT PLC Library: IEC 60870-5-104 (low level) Transport Interface		
APCI (Protokollsteuerinformation der Anwendungsschicht) Transportschnittstelle (Anwender zur TCP-Schnittstelle)		
Auswahl aus der TCP/IP-Protokollsammlung (RFC 2200)		Transportschicht (4) Vermittlungsschicht (3) Verbindungsschicht (2) Physikalische Schicht (1)

Anmerkung: Die Schichten 5 und 6 werden nicht benutzt.

11.1 SPS-API

Einfache Projekte mit kompletten Sourcen finden Sie hier: Transport-Interface-Beispiele.Transport-Interface-Beispiele [▶ 449]

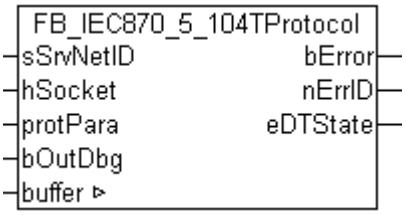
Eine SPS-Applikation, die über das Transport Interface mit einer Zentralstation oder Unterstation kommunizieren soll, benötigt folgende Resourcen:

- Eine Instanz des Kommunikationsbausteins: FB_IEC870_5_104TProtocol [▶ 443];
- Eine Instanz des Funktionsbausteins zur Herstellung der TCP/IP-Verbindung: FB_ServerClientConnection oder FB_ClientServerConnection;
- Eine Instanz des TX/RX-Datenpuffers: ST_IEC870_5_101TBuffer;
- Eine Instanz des Funktionsbausteins zur Manipulation des TX/RX-Datenpuffers: FB_IEC870_5_101TBufferCtrl;

.....

11.1.1 FB_IEC870_5_104TProtocol

Ab der Produktversion: TwinCAT PLC Library IEC870-5-104 Unterstation v2.0.6 und höher.



Der Kommunikationsbaustein FB_IEC870_5_104TProtocol implementiert die ACPI-Funktionen der IEC60870-5-104-Norm (Start/Stop-Data Transfer, Test-Frames, Send/Receive Frame-Counter usw.). Beim Protokollfehler wird ein entsprechender Fehlercode am Ausgang des Funktionsbausteins ausgegeben und die Datenübertragung unterbrochen. Um den Datenaustausch erneut aktivieren zu können, muss die Aktion INIT aufgerufen werden. Es werden dabei z.B. die Framezähler, Sende- und der TX/RX-Datenpuffer zurückgesetzt. Der Kommunikationsbaustein erwartet eine TX/RX-Datenpuffervariable. Diese Variable muss per VAR_IN_OUT an den Baustein übergeben werden.

Der Funktionsbaustein besitzt folgende Aktionen:

- **INIT** (Führt eine Initialisierung des Funktionsbausteins durch);
- **STARTDT** (Sendet ein Start-Data-Transfer-Frame an den Kommunikationspartner);
- **STOPDT** (Sendet Stop-Data-Transfer-Frame an den Kommunikationspartner);

Verbindungsaufbau

Mit einem gesonderten Baustein z.B. FB_ServerClientConnection muss die TCP/IP-Verbindung auf- und abgebaut werden. Die SPS-Applikation kann dadurch selber auf mögliche Protokollfehler reagieren und die Verbindung schließen oder z.B. den Dienst Process-Reset implementieren. Dieser Baustein liefert am Ausgang ein Verbindungshandle, den Verbindungsstatus und Informationen über Fehler die beim Verbindungsaufbau/-abbau aufgetreten sind.
Das Verbindungshandle wird von dem Kommunikationsbaustein benötigt.

Protokollkonfiguration

Der Kommunikationsbaustein besitzt eine protPara-Variable vom strukturierten Typ. Über diese Variable können Protokollparameter z.B. iK, iW, Start/Stop-Datentransfer-Verhalten usw. konfiguriert werden.

VAR_IN_OUT

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

buffer: TX/RX Datenpuffer.

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 mit der Netzwerkadresse des TwinCAT TCP/IP Connection Servers. Für den lokalen Rechner (default) kann auch ein Leerstring angegeben werden.

hSocket: Das TCP/IP-Verbindungshandle des Kommunikationspartners dessen Daten empfangen oder gesendet werden sollen.

protPara: IEC60870-5-104-Protokollparameter.

bOutDbg:: Aktiviert/deaktiviert die Debug-Ausgabe der TCP/IP-Frames in der TwinCAT System Manager-Loggeransicht.

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    nErrID      : UDINT;
    eDTState    : E_IEC870_5_104DataTransferState := eIEC870_STOPDT; (* Data transfer state *)
ND_VAR
```

bError: Dieser Ausgang wird auf TRUE gesetzt, sobald ein Fehler bei der Datenübertragung aufgetreten ist.

nErrID: Liefert bei einem gesetzten bError-Ausgang einen Fehlercode;

eDTState: Status des IEC60870-5-104-Datenaustauschs (STARTDT, STOPDT).

Beispiel:

Beispielprojekte: IEC60870-5-104 Transport Interface [▶ 449]

Voraussetzungen

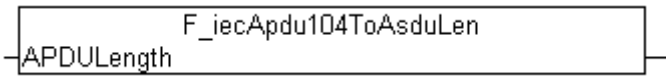
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder CX (x86) CX (ARM)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib; TcIEC870_5_101.Lib; werden automatisch eingebunden)

Sehen Sie dazu auch

- ST_IEC870_5_104ProtocolParams [▶ 446]
- E_IEC870_5_104DataTransferState [▶ 448]

11.1.2 F_iecApdu104ToAsduLen

Ab der Produktversion: TwinCAT PLC Library IEC60870-5-104 Unterstation v2.0.6 und höher.



Die Funktion berechnet für das IEC 60870-5-104 Protokoll die maximal verfügbare ASDU-Octetlänge anhand der konfigurierten APDU-Länge. Die maximal verfügbare ASDU-Länge wird z.B. bei der Konfiguration der ST_IEC870_5_101TBuffer-Variablen benötigt. Diese Datenstruktur (TX/RX-Datenpuffer) wird beim Datenaustausch über das IEC60870-5-104 Transport Interface benutzt.

FUNCTION F_iecApdu104ToAsduLen: BYTE

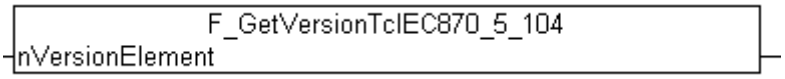
```
VAR_INPUT
    APDULength : BYTE;
END_VAR
```

APDULength : Die verfügbare APDU-Länge (siehe Kompatibilitätsliste).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

11.1.3 F_GetVersionTcIEC870_5_104



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_104: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib werden automatisch eingebunden)
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) Inter
net 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_AP
DULEN;(* 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 frame 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 CO
T 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 : Nicht benutzt, reserviert.

t1 : Max. Timeout-Zeit für eine Antwort auf ein STARTDTAct-, STOPDTAct- oder TESTFRAct-Frame.

t2 : Spätestens nach dieser Zeit wird ein S-Frame versendet.

t3 : Spätestens nach dieser Zeit wird ein Test-Frame versendet.

iK : Spätestens nach diesem gesendeten APDUs im I-Format, welches nicht quittiert wurde, wird die Verbindung geschlossen.

iW : Spätestens nach dem Empfang von w APDUs im I-Format quittiert der Empfänger den Empfang.

bSFrameACK : Sende S-Frames.

bTESTFRAct : Sende Test-Frames.

bDTControlled : Warte auf STARTDT-, STOPDT-Frame vom Master.

bControlDT : Nur bei der Masterkonfiguration: Sende STARTDT zum Slave.

bSTARTDTCon : Sende STARTDT Bestätigung

bSTOPDTCon : Sende STOPDT Bestätigung.

bTESTFRCon : Sende TESTFR Bestätigung.

eAcceptMode : Legt fest ob Verbindungen zu allen Remote-Clients, oder nur zu Clients mit bestimmter Host- und Port-Adresse zugelassen werden sollen. Default: Akzeptiere alle ankommenden Verbindungen.

sRemoteHost : Host-Adresse des Remote-Clients. Bei eAcceptMode = eACCEPT_ALL wird dieser Parameter ignoriert.

nRemotePort : Port-Adresse des Remote-Clients. Bei eAcceptMode = eACCEPT_ALL wird dieser Parameter ignoriert.

APDULength : Maximale Länge der APDU.

bThrottleMode: Implementiert ab IEC870-5-104 slave library **v2.0.0** und höher. Die TCP/IP sockets werden pollend aus der SPS abgefragt. Mit diesem Parameter kann die Anzahl der pollenden Lesezugriffe und damit auch die Systemauslastung reduziert werden, besonders dann wenn nur selten Daten empfangen werden (z.B. Generalabfrage oder Uhrzeitsynchronisationsbefehle).

- FALSE: Nach jedem Lesezugriff folgt der nächste auch wenn keine neuen Daten empfangen wurden (höhere Systemauslastung);
- TRUE: Nach jedem Lesezugriff ,der keine neuen Daten liefert wird eine Verzögerung eingefügt. Der nächste Lesezugriff wird danach verzögert ausgeführt (kleinere Systemauslastung). Nach jedem Lesezugriff, der neue Daten liefert, wird der nächste ohne Verzögerung ausgeführt. Die max. Verzögerungszeit <= 2 Sekunden.

bPackFrames: Implementiert ab IEC870-5-104 slave library **v2.0.4** und höher. Per Default wird mit einem TCP/IP-Send-Aufruf nur eine einzelne APDU verschickt. Wenn Sie diesen Parameter auf TRUE setzen, können Sie die Sende-Performance wesentlich erhöhen und so die Sende-Buffer-Überläufe verringern.

- FALSE: Für eine APDU wird ein TCP/IP-Send-Aufruf benötigt. Ca. alle 3 SPS-Zyklen kann max. eine APDU verschickt werden;
- TRUE: Mehrere APDUs werden zu einem größeren TCP/IP-Block (maximal aber iK-Frames) zusammengefasst und mit einem TCP/IP-Send-Aufruf verschickt;

bRetainBuffer: Implementiert ab IEC870-5-104 slave library **v3.0.14** und höher. In der Standardeinstellung (FALSE) werden die internen Tc/Rx-Puffer beim Verbindungsabbruch gelöscht. Wenn dieses Flag auf TRUE gesetzt ist dann werden die noch nicht gesendeten ASDU's im internen Sendepuffer nicht gelöscht. Somit ist eine Offline-Pufferung von von ca. 100-200 Messwerten (abhängig von der ASDU-Größe) im RAM-Speicher möglich. Die Station entfernt immer dann die ASDUs aus dem Sendepuffer wenn deren Empfang bestätigt wurde. Bitte beachten Sie, dass die bereits gesendeten aber noch nicht bestätigten ASDUs auch im Puffer verbleiben und nächstes mal erneut gesendet werden. Möglicherweise empfängt die andere Station dann die Werte zwei Mal. Sie können dieses Verhalten über zwei weitere Parameter: **bCOTFilter** und **cotFilter** konfigurieren.

bCOTFilter: Implementiert ab IEC870-5-104 slave library **v3.0.14** und höher. Aktiviert/deaktiviert die Filtermaske mit den Übertragungsursachen. Dieser Parameter ist nur dann gültig wenn auch der *bRetainBuffer*-Parameter auf TRUE gesetzt wurde. Bei jedem neuen Verbindungsaufbau wird intern eine SessionID hochgezählt. Diese SessionID wird immer an die empfangenen und gesendeten ASDUs angehängt. Dadurch können die ASDUs die noch nicht gesendet wurden und im Offline-Puffer verbleiben der alten Verbindung zugeordnet werden. Diese ASDUs können dann mit Hilfe der COT-Maske (**cotFilter**) gefiltert und verworfen werden (COT = cause of transfer). Dies ist manchmal notwendig wenn der andere Kommunikationspartner die wiederholten ASDUs nicht akzeptiert.

cotFilter: Implementiert ab IEC870-5-104 slave library **v3.0.14** und höher. Filtermaske mit Übertragungsursachen (COT = Cause of transfer). Dieser Parameter ist nur dann gültig wenn auch der *bRetainBuffer*-Parameter und *bCOTFilter*-Parameter gesetzt wurde. Die Übertragungsursache der zu sendenden ASDUs wird nur dann geprüft wenn deren SessionID nicht zu der aktuellen SessionID passt (d.h. die ASDUs die aus der vorherigen Verbindung stammen). Jedes Bit entspricht einer Übertragungsursache. Die Übertragungsursache wird nur dann überprüft wenn das entsprechende Bit gesetzt wurde.

Die Übertragungsursachen sind auf folgende Weise in den Bits kodiert:

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

... usw.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib werden automatisch eingebunden)
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 (master only)*)
    eIEC870_STARTDT_PENDING (* waiting for STARTDT confirmation (master only)*)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9.0 Build >= 1030	PC oder CX (x86)	TcIEC870_5_104.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcpIp.Lib; TcUtilities.Lib; TcSocketHelper.Lib; TcIEC870_5_101.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

11.2 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE Image-Version usw.).

2. Überprüfen Sie die Installationshinweise (z.B. Installation der CAB-Files auf einem CE-System).

3. Bei Verbindungsproblemen kann der PING-Befehl dazu benutzt werden, um festzustellen, ob der Kommunikationspartner über die Netzwerkverbindung erreichbar ist. Wenn dies nicht der Fall ist überprüfen Sie die Netzwerkkonfiguration und die Firewall-Einstellungen [▶ 510].

4. Überprüfen Sie ob die Netzwerkadresse, Portnummer die Sie an die Funktion F_CreateServerHnd() bzw. an den Funktionsbaustein FB_ClientServerConnection()/FB_ServerClientConnection() übergeben richtig sind.

5. Überprüfen Sie ob der Funktionsbaustein einen Fehlercode ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes.

6. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 446] (iK, iW, t0, t1, t2, t3, APDULength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern des Kommunikationspartners.

7. Überprüfen Sie die am TX/RX-Datenpuffer (instanz von ST_IEC870_5_101TBuffer) konfigurierten Adresslängen: Länge der ASDU-Adresse, Länge der Informationsobjekt-Adresse, Länge der Übertragungsursache, max. ASDU-Länge. Vergleichen Sie die Adresslängen mit der Konfiguration des Kommunikationspartners.

8. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, Adresse des Informationsobjektes usw.).

9. Überprüfen Sie, ob der andere Kommunikationspartner einen Fehlercode ausgibt.

10. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU-Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben.

11. Überprüfen Sie die Verwendung des FB_SocketCloseAll()-Funktionsbausteins und des LISTEN_MODE_CLOSEALL-Parameters wenn Sie über mehrere TCP/IP Verbindungen (Server/Clients) in einem Laufzeitsystem kommunizieren.

Bei mehreren Verbindungen aktivieren Sie nur eine Instanz des FB_SocketCloseAll()-Funktionsbausteins einmalig im Initialisierungsschritt beim Programmstart. Der LISTEN_MODE_CLOSEALL-Parameter darf in diesem Fall nicht mehr verwendet werden.

12. Eine komplette Aufzeichnung der Netzwerkkommunikation kann mit Sniffer-Tools wie Wireshark durchgeführt werden. Die Aufnahme kann dann vom Beckhoff-Supportpersonal analysiert werden.

11.3 Beispiele

In Beispielprojekten sind folgende Stationsparameter für die Unterstation oder Zentralstation eingestellt:

- Server host address: **127.0.0.1** (Sie müssen mindestens diesen Parameter an Ihre Zielplattform anpassen!)
- 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**

Voraussetzungen

SPS Project	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11729046539.zip	Einfache **Slave-Applikation** die das TwinCAT IEC60870-5-104 Transport Interface nutzt. MAIN implementiert den Verbindungsaufbau/-abbau und einen einfachen TX/RX-Datenaustausch: • Datenpunkte in Überwachungsrichtung: M_SP_NA_1(IOA:=100) und M_BO_TB_1(IOA:=400). Datenpunkte in Kontrollrichtung: C_SC_NA_1(IOA:=10); • Einfache Implementierung einer Uhrzeitsynchronisation: C_CS_NA_1(IOA:=0); • Einfache Implementierung einer Generalabfrage: C_IC_NA_1(IOA:=0);

SPS Project	Beschreibung
	• Einfache Implementierung eines Einzehlbefehls: C_SC_NA_1(IOA:=10); • Einfache Implementierung einer Zählerabfrage: C_CI_NA_1(IOA:=0);
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11729047947.zip	Einfache **Master-Applikation** die das TwinCAT IEC60870-5-104 Transport Interface nutzt. MAIN implementiert den Verbindungsaufbau/-abbau und in einen einfachen TX/RX-Datenaustausch: • Die Beispielapplikation sendet alle 20 Sekunden eine Generalabfrage: C_IC_NA_1(IOA:=0); • Eine steigenden Flanke an der *bSCmd*-Variablen sendet ein einfaches Befehl: C_SC_NA_1(IOA:=10); • Eine steigenden Flanke an der *bClockSyn*-Variablen sendet ein Uhrzeitsynchronisationsbefehl: C_CS_NA_1(IOA:=0);

12 TcIEC870_5_104Slave: IEC 60870-5-104 Unterstation (slave)

Mit den SPS-Funktionen und -Funktionsbausteinen können Unterstationen (Slaves) nach der IEC 60870-5-104 Norm in TwinCAT SPS realisiert werden.

Die SPS-Bibliothek verfügt über zwei Software-Schnittstellen. Die Endapplikation setzt auf einer der Schnittstellen auf. Die Wahl der Schnittstelle hängt von den Anforderungen an die Endapplikation ab. Im Folgenden werden die Eigenschaften beider Schnittstellen kurz beschrieben.

"High level"-Schnittstelle: IEC 60870-5-104 Unterstation

Bei dieser Schnittstelle handelt es sich um eine sogenannte "Ein-Baustein-Lösung". Alle Funktionalitäten sind in einem SPS-Baustein gekapselt. Der Baustein implementiert die wichtigsten Dienste und Funktionen. Diese Implementierung ist für über 90% der Anwendungen ausreichend.

Pro: Sehr kleiner SPS-Programmieraufwand um eine laufende Applikation zu erhalten; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw sind bereits in dem Baustein implementiert und werden automatisch ausgeführt; Das Mapping der IEC<->SPS Prozessdaten und das der Datenpunkte wird über Funktionsaufrufe konfiguriert; Der SPS-Programmierer muss nicht sehr gut mit der Protokollnorm vertraut sein;

Contra: Die SPS-Applikation hat nur einen geringen Einfluss auf die Protokollausführung; Kein Einfluss auf die Ausführung der Dienste, diese werden intern automatisch ausgeführt; Zeitstempel werden von dem Baustein automatisch generiert und können nicht verändert (von extern übergeben) werden; Es ist z.B. nur die direkte Befehlsausführung möglich; Schlechtere Performance bei vielen Datenpunkten.

Diese Schnittstelle empfiehlt sich wenn Sie:

- Mit der Protokollnorm nicht vertraut sind;
- Eine einfache Applikation mit einer handvoll Datenpunkten implementieren möchten (<1000);
- Keine grossen Performace-Anforderungen an die Applikation stellen;
- Keine besondere Befehlsausführung wie Select/Execute oder Daten + Zeistempel von externen Geräten versenden möchten;
- Keine Funktionalitäten benötigen die laut Kompatibilitätsliste nicht unterstützt werden;

"Low level"-Schnittstelle: IEC 60870-5-104 Transport Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können manipuliert werden (z.B. ein eigener Zeitstempel, Select/Execute oder eine besondere Befehlsausführung usw); Weil nur die benötigten Dienste implementiert werden kann eine hohe Performance erreicht werden; Hohe Performace bei vielen Datenpunkten;

Contra: Grösserer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Diese Schnittstelle empfielt sich wenn Sie:

- Mit der Protokollnorm vertraut sind;
- Eine Protokollkonverter-Applikation implementieren möchten;
- In der Applikation die verfügbaren Norm-Funktionalitäten fast vollständig implementieren müssen;

- Besondere Funktionalitäten wie z.B. das weiterleiten der Zeitstempel von einem Modbusgerät oder die Kontrolle über die Befehlsausführung haben möchten;
- Funktionalitäten benötigen die laut Kompatibilitätsliste nicht unterstützt werden;
- Viele Datenpunte haben (>1000) und eine hohe Performace benötigen;

Kompatibilitätsliste

für TwinCAT SPS Bibliothek: IEC 60870-5-104 Unterstation (bezieht sich auf die "High level"-Schnittstelle). Hier können Sie die https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11764746379.zip

Systemvoraussetzungen

Programmierumgebung:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT Installation Level: TwinCAT PLC oder höher;
- TwinCAT System Version 2.9.0 Build >= 1030 (CX (ARM) Build >= 1301) oder höher;

Zielplattform:

- Industrie PC oder Embedded PC/CX (x86, ARM);
- Betriebssystem:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86) (image v1.75 oder höher);
  - Windows CE (ARM) (image v2.13 oder höher);
- TwinCAT SPS-Laufzeitsystem Version 2.9.0 oder höher;

Produktkomponenten

- **TcIEC870_5_104Slave.Lib** (implementiert die Beckhoff IEC60870-5-104 Unterstation). Diese Bibliothek muss in dem SPS-Projekt eingebunden werden. Alle anderen Bibliotheken werden automatisch hinzugefügt.
- TcIEC870_5_104.Lib (implementiert das Übertragungsprotokoll nach der IEC60870-5-104 Norm);
- TcIEC870_5_101.Lib (implementiert die Verbindungsfunktionen und gemeinsame Datentypen);
- TcSocketHelper.Lib (TCP/IP Hilfsfunktionen);
- TcpIp.Lib (TCP/IP Basisfunktionen);
- TwinCAT TCP/IP Connection Server;

Installation auf Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

Die SPS-Bibliotheken werden während der Installation in den ..\TwinCAT\Plc\Lib-Ordner hineinkopiert. Der TwinCAT TCP/IP Connection Server wird in die Liste der TwinCAT Server eingetragen. Beim TwinCAT Start wird der TCP/IP Connection Server automatisch gestartet und beim TwinCAT Stop gestoppt.

Installation auf Windows CE

Produktversion für das Laufzeitsystem unter Windows CE ist als separates Produkt verfügbar. Führen Sie folgende Schritte aus wenn Sie eine Produktversion für Windows CE erworben haben:

- Installieren Sie das Produkt zuerst wie gewohnt auf Ihrem Programmier-PC. Die SPS-Bibliotheken werden während der Installation in den ..\TwinCAT\Plc\Lib-Ordner hineinkopiert.
- X86 CPU (CX1000, CX1020, IPC):
  - Nach der Installation finden Sie im Ordner: **...\TwinCAT\CE\TCPIP\Install** ein Cabinet-File für das CE-Laufzeitsystem.

  - Kopieren Sie die darin befindliche Datei: **TcTCPIPSvrCe.I586.CAB** in einen Ordner auf dem CE-Laufzeitsystem.
- ARM CPU (CX9000), TwinCAT PLC Library: IEC 60870-5-104 Unterstation (slave) **v2.0.4** und höher:
  - Nach der Installation finden Sie im Ordner: **...\TwinCAT\CE\TCPIP\Install** ein Cabinet-File für das CE-Laufzeitsystem.
  - Kopieren Sie die darin befindliche Datei: **TcTCPIPSvrCe.ARMV4I.CAB** in einen Ordner auf dem CE-Laufzeitsystem.
- Auf dem CE-System: Installieren Sie (durch einen Doppelklick auf das das Cabinet-File) die CE-Komponenten.
- Rebooten Sie das CE-Gerät. Der TwinCAT TCP/IP Connection Server wird mit dem CE-Betriebssystem automatisch gestartet.

Dies ist nur eine kurze Produktinformation (kein vollständiges Handbuch). Bitte installieren Sie sich die vollständige Ausgabe des Beckhoff Information System.

Sie finden es

- auf sämtlichen Beckhoff-Produkt-DVDs
- auf unserem Web-Server http://www.beckhoff.com unter Download.

Beispiele

Beispiele befinden sich in der Beckhoff Information System Dokumentation der SPS-Bibliotheken.

Link zu "High level" Beispiel-Übersichtsseite: IEC 60870-5-104 Unterstation [▶ 479];

Link zu "Low level" Beispiel-Übersichtsseite: IEC 60870-5-104 Transport Interface [▶ 449];

Weiterführende Dokumentation

- Dokumentation zur TwinCAT PLC Library ("Low level"-Schnittstelle): IEC 60870-5-104 Transport Interface [▶ 442];
- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268];
- Dokumentation: 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 Einführung (Tutorial)

Die Einführung ist eine Anleitung wie Sie in der TwinCAT SPS eine IEC60870-5-104 Unterstation (slave) implementieren und konfigurieren können.

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

12.1.1 SPS-Projekt anlegen, SPS-Bibliotheken einbinden

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

- Starten Sie TwinCAT PLC Control.

- Mit Datei -> Neu legen Sie ein neues SPS-Projekt an. Wählen Sie als Zielsystem PC or CX (x86, ARM).

- Als nächstes wird automatisch ein neuer Programmbaustein MAIN angelegt. Wählen Sie als Sprache des Bausteins ST (Strukturierter Text). Bestätigen Sie dies.

- Aus dem Menü wählen Sie Fenster -> Bibliotheksverwaltung und dann Einfügen -> Weitere Bibliothek...

- Aus der Liste der TwinCAT Bibliotheken wählen Sie **TcIEC870_5_104Slave.Lib** aus und bestätigen dies.

12.1.2 Applikationsobjekt-Datenbank der Unterstation definieren und konfigurieren

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Applikationsobjekte = Single Points, Double Points, Measured Values, Short Floating Point Values usw.

In diesem Beispiel wurden die Befehle so konfiguriert, dass die Prozessdaten der Befehle im gleichen Speicherbereich aber auf einem anderen Byte-/Bit-Offset wie die Daten der Information in Überwachungsrichtung liegen. Sie können aber auch die Befehle auf den gleichen Byte- Bit-Offset wie die Information in Überwachungsrichtung legen.

Beispiel:

C_SC_NA_1 mit IOA = 10 auf den gleichen Byte- und Bit-Offset wie M_SP_NA_1 mit IOA = 100 (beide Byte-Offset = 100 und Bit-Offset = 0). In diesem Fall wird eine Wertänderung durch ein Kommando von der Leitstation eine Übertragung des M_SP_NA_1 mit der Objektadresse 100 und Übertragunsursache <11> (returned by remote command) zur Folge haben.

Als Beispiel konfigurieren wir in dem Einführungsprojekt folgende Applikationsobjekte:

Voraussetzungen

Arrayelement	ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
0	M_SP_NA_1	100	Generalabfrage	0	Merker	100	0	1 Bit
1	M_SP_NA_1	101	Generalabfrage	0	Merker	100	1	1 Bit
2	M_SP_TB_1	102	Generalabfrage	0	Merker	100	2	1 Bit
3	M_DP_NA_1	200	Generalabfrage	0	Merker	200	0	2 Bits
4	M_DP_NA_1	201	Generalabfrage	0	Merker	200	2	2 Bits
5	M_DP_TB_1	202	Generalabfrage	0	Merker	200	4	2 Bits
6	M_ST_NA_1	300	Generalabfrage	0	Merker	300	0	1 Byte

Arrayelement	ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
7	M_ST_NA_1	301	Generalabfrage	0	Merker	301	0	1 Byte
8	M_ST_TB_1	302	Generalabfrage	0	Merker	302	0	1 Byte
9	M_BO_NA_1	400	Generalabfrage	0	Merker	400	0	4 Byte
10	M_BO_NA_1	401	Generalabfrage	0	Merker	404	0	4 Byte
11	M_BO_TB_1	402	Generalabfrage	0	Merker	408	0	4 Byte
12	M_ME_NA_1	500	Generalabfrage	0	Merker	500	0	2 Byte
13	M_ME_NA_1	501	Generalabfrage	0	Merker	502	0	2 Byte
14	M_ME_TD_1	502	Generalabfrage	0	Merker	504	0	2 Byte
15	M_ME_NB_1	600	Generalabfrage	0	Merker	600	0	2 Byte
16	M_ME_NB_1	601	Generalabfrage	0	Merker	602	0	2 Byte
17	M_ME_TE_1	602	Generalabfrage	0	Merker	604	0	2 Byte
18	M_ME_NC_1	700	Generalabfrage	0	Merker	700	0	4 Byte
19	M_ME_NC_1	701	Generalabfrage	0	Merker	704	0	4 Byte
20	M_ME_TF_1	702	Generalabfrage	0	Merker	708	0	4 Byte
21	M_IT_NA_1	800	Generalzählerabfrage und Mode A (lokal Umspeichern mit Spontanübertragung alle 15s)	0	Merker	800	0	4 Byte
22	M_IT_NA_1	801	Generalzählerabfrage und Mode A (lokal Umspeichern mit Spontanübertragung alle 15s)	0	Merker	804	0	4 Byte

Arrayelement	ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
23	M_IT_TB_1	802	Generalzählerabfrage und Mode A (lokal Umspeichern mit Spontanübertragung alle 15s)	0	Merker	808	0	4 Byte
Commands								
24	C_SC_NA_1	10	-	0	Merker	2100	0	1 Bit
25	C_SC_NA_1	11	-	0	Merker	2100	1	1 Bit
26	C_SC_TA_1	12	-	0	Merker	2100	2	1 Bit
27	C_DC_NA_1	20	-	0	Merker	2200	0	2 Bit
28	C_DC_NA_1	21	-	0	Merker	2200	2	2 Bit
29	C_DC_TA_1	22	-	0	Merker	2200	4	2 Bit
30	C_RC_NA_1	30	-	0	Merker	2300	0	1 Byte
31	C_RC_NA_1	31	-	0	Merker	2301	0	1 Byte
32	C_RC_TA_1	32	-	0	Merker	2302	0	1 Byte
33	C_BO_NA_1	40	-	0	Merker	2400	0	4 Byte
34	C_BO_NA_1	41	-	0	Merker	2404	0	4 Byte
35	C_BO_TA_1	42	-	0	Merker	2408	0	4 Byte
36	C_SE_NA_1	50	-	0	Merker	2500	0	2 Byte
37	C_SE_NA_1	51	-	0	Merker	2502	0	2 Byte
38	C_SE_TA_1	52	-	0	Merker	2504	0	2 Byte
39	C_SE_NB_1	60	-	0	Merker	2600	0	2 Byte
40	C_SE_NB_1	61	-	0	Merker	2602	0	2 Byte
41	C_SE_TB_1	62	-	0	Merker	2604	0	2 Byte
42	C_SE_NC_1	70	-	0	Merker	2700	0	4 Byte

Arrayelement	ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
43	C_SE_NC_1	71	-	0	Merker	2704	0	4 Byte
44	C_SE_TC_1	72	-	0	Merker	2708	0	4 Byte

Datenbankvariable deklarieren

Die Applikationsobjekt-Datenbank ist eine Array-Variable vom Typ ST_IEC870_5_101AODBEntry [▶ 312]. Jedes Array-Element entspricht einem Applikationsobjekt. Die maximale Anzahl der Applikationsobjekte ist frei wählbar und nur durch den verfügbaren Speicher begrenzt. Sie müssen sich auf eine konstante maximale Anzahl während der SPS-Programmierung festlegen. Zur Laufzeit kann die maximale Anzahl der Applikationsobjekte nicht mehr verändert werden.

In unserem Beispiel werden 50 Applikationsobjekte deklariert. Diese Anzahl reicht für die meisten Anwendungen aus. Beachten Sie, dass sehr viele Applikationsobjekte auch entsprechend viel Speicher und Laufzeit benötigen.

Definieren Sie folgende Variable in MAIN:

```
PROGRAM MAIN
VAR
     AODB : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
END_VAR
```

Applikationsobjekte konfigurieren

Während der Konfiguration der einzelnen Applikationsobjekte werden unter anderem der Objekt-Typ (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 usw.), die Objekt-Adresse und weitere Objekt-Parameter festgelegt.

Die Konfiguration der gewünschten Applikationsobjekte wird zur Programmlaufzeit durchgeführt. Jedes Applikationsobjekt (Datenbank-Array-Element) wird durch einen einmaligen Aufruf der F_iecInitAOEntry [▶ 277]-Funktion konfiguriert. Das zu konfigurierende Array-Element wird an die Funktion per VAR_IN_OUT übergeben. Im Regelfall wird die Konfiguration beim SPS-Programmstart einmalig in einer Init-Routine durchgeführt. Die Funktion F_iecInitAOEntry erwartet folgende Funktionsparameter (von links nach rechts):

```
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: Applikationsobjekt [▶ 326]-Typ (ASDU identifier, z.B.: M_SP_NA_1 für Single Point oder M_DP_NA_1 für Double Point). Beachten Sie, dass nur die in der Kompatibilitätsliste aufgeführten ASDU-Typen verwendet werden können. Unzulässige Typen werden ignoriert.

objAddr : Objektadresse, z.B. 100. Jedes Applikationsobjekt sollte mit einer eindeutigen Adresse konfiguriert werden.

group: Group-Konfigurationsparameter. Die verfügbaren Group-Parameter sind als Konstanten definiert und können mit ODER-Operator kombiniert werden. Z.B.: IEC870_GRP_INROGEN OR IEC870_GRP_PERCYC.

Hier finden Sie die Beschreibung aller Group-Konfigurationsparameter [▶ 347].

mutiplier: Basiszeitmultiplikator für die zyklische/periodische Datenübertragung (0=Deaktiviert). Die Basiszeit wird über die Systemparameter konfiguriert. Wurde die Basiszeit z.B. auf T#10s gesetzt und der Multiplikator auf den Wert 2, dann werden die periodischen/zyklischen Daten des Applikationsobjekts alle 20 Sekunden gesendet.

ioMapType: Dieser Parameter [▶ 329] legt fest, aus oder in welchen Prozessdatenbereich der TwinCAT SPS die IEC-Prozessdaten zur Laufzeit gemappt werden sollen (inputs, outputs, memory, data).

byteOffs: Prozessdatenbereich Byteoffset;

bitOffs: Prozessdatenbereich Bitoffset;

dbEntry: Applikationsobjekt das konfiguriert werden soll (ein Datenbankvariable-Arrayelement, das an die Funktion per VAR_IN_OUT übergeben wird).

Um die Applikationsobjekte beim Programmstart zu konfigurieren wird in MAIN folgender SPS-Code hinzugefügt:

```
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
```

12.1.3 Mapping der SPS- und IEC-Prozessdaten

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Die TwinCAT SPS-Prozessdaten werden zur Programmlaufzeit zyklisch in die IEC-Prozessdaten (Applikationsobjekte) und umgekehrt gemappt (kopiert). Für das Mapping der IEC<->SPS Prozessdaten können bis zu 4 Prozessdatenbereiche (IO-Eingänge, IO-Ausgänge, Merkerbereich, Datenbereich) als Puffervariablen im SPS-Programm deklariert werden. Die Bytegröße der Puffer ist frei wählbar und kann für jeden Bereich unterschiedlich gewählt werden. Unbenutzte Bereiche müssen nicht unbedingt deklariert werden.

In unserem Einführungsbeispiel deklarieren wir 4 SPS-Prozessdatenbereiche mit jeweils 3000 Bytes:

```
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
```

Die Zuordnung, wie die Prozessdaten zur Laufzeit gemappt werden sollen, wird während der Konfiguration der Applikationsobjekte mit der F_iecInitAOEntry [▶ 277]-Funktion festgelegt.
Siehe auch in: Applikationsobjekte definieren und konfigurieren [▶ 454].

Die Puffervariablen wurden nun als Byte-Arrays deklariert. Um auf die gewünschten Daten besser zugreifen zu können definieren wir die einzelnen Variablen ein zweites Mal und legen diese auf die entsprechenden Byte/Bit-Offsetadressen. Bei einer Änderung im Byte-Array wird die entsprechende einzelne Variable gleichzeitig geändert und umgekehrt. Dies ist aber nicht zwingend notwendig. Sie können direkt auf die Bytes/Bits der Byte-Array-Puffervariablen zugreifen.

```
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 der IEC<->SPS Prozessdaten in der Unterstation

Prozessdaten in Überwachungsrichtung (Slave->Master information)

Beispiel 1

Single point information (M_SP_NA_1) mit der IOA = 100, SPS Merkerbereich, Byteoffset = 100, Bitoffset = 0.

msgSingle_0 == memory[100].0 -> Unterstation FB -> ... -> Leitstation

Beispiel 2

Measured value, short floating point value (M_ME_NC_1) mit der IOA = 700, SPS Merkerbereich, Byteoffset = 700, Bitoffset = 0 (bedeutungslos).

msgFloating_0 == memory[700..703] -> Unterstation FB -> ... -> Leitstation

Prozessdaten in Steuerungsrichtung (Master->Slave commands)

Beispiel 1

Single command state (C_SC_NA_1) mit der IOA = 10, SPS Merkerbereich, Byteoffset = 2100, Bitoffset = 0.

Leitstation -> ... -> Unterstation FB -> memory[2100].0 == cmdSingle_0

Beispiel 2

Set point, short floating point value (C_SE_NC_1) mit der IOA = 70, SPS Merkerbereich, Byteoffset = 2700, Bitoffset = 0 (bedeutungslos).

Leitstation -> ... -> Unterstation FB -> memory[2700..2703] == cmdFloating_0

12.1.4 Instanz der IEC60870-5-104 Unterstation deklarieren und aufrufen

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Die gesamte Funktionalität einer Unterstation ist im Funktionsbaustein FB_IEC870_5_104Slave gekapselt. Mit einer Instanz kann eine Verbindung zum Master aufgebaut werden. Für eine weitere Verbindung deklarieren Sie eine weitere Instanz und übergeben an diese zweite Instanz das gleiche Server-Handle (*hServer*-Variable), oder Sie verwenden den FB_IEC870_5-104SlaveGrp [▶ 470]-Funktionsbaustein (empfohen). Die IP-Addresse müssen Sie passend zu der IP-Adresse Ihres Zielsystems setzen.

Fügen Sie im Deklarationsteil von MAIN folgenden SPS-Code ein:

```
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_HSERVER;
    server      : FB_IEC870_5_104Slave;
END_VAR
```

und im Programmteil wird die Instanz aufgerufen:

```
IF init THEN
    init := FALSE;
...



F_CreateServerHnd( '', '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-Protokollparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Über die IEC60870-5-104-Protokollparameter kann das Verhalten der Unterstation an die Anforderungen des Masters angepasst werden. Die meisten Parameter sind mit Defaultwerten vorbelegt, so dass diese nicht verändert werden müssen.

Im unserem Beispiel verändern wir die Werte der iK- und iW-Parameter:

```
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
```

Die Dokumentation aller Übertragungsprotokoll-Parameter finden Sie hier: ST_IEC870_5_104PotocolParams [▶ 446].

12.1.6 Systemparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Über die Systemparameter wird z.B. die gemeinsame ASDU-Adresse und die Anwenderfunktionen der Unterstation konfiguriert.

In unserer Einführung konfigurieren wir folgende Systemparameter:

- Die gemeinsame ASDU-Adresse wird auf 7 gesetzt. (*asduAddr*)

- Die Systemzeit der Unterstation wird während der Initialisierung mit der Systemzeit des lokalen TwinCAT PC's synchronisiert (*bUsePCTime*).
- Die Synchronisierung der Systemzeit der Unterstation mit dem Uhrzeitsynchronisationsbefehl wird aktiviert (*bSyncTime*).
- Während der Synchronisierung der Systemzeit in der Unterstation soll die Systemzeit des TwinCAT PC's nicht synchronisiert werden (*bSyncPCTime*).
- Das Senden von M_EI_NA_1 (End of init) an die Zentralstation wird aktiviert (*bEndOfInit*).
- Das Senden der periodischen/zyklischen Daten wird deaktiviert (*bPerCyclic*). Die Basiszeit fürs Senden dieser Daten wird auf 5s gesetzt.
- Hintergrundabfrage wird deaktiviert (*bBackScan*). Die Zykluszeit für Hintergrundabfrage wird auf 30s gesetzt.
- Das lokale Umspeichern und Reset der Zählerstände wird aktiviert (*bPerFRZ*) und die Zykluszeit fürs Umspeichern und Reset auf 15s gesetzt.
- Das loggen der Debugmeldungen im Application-Log wird aktiviert (*dbgMode*). Es werden Änderungen im Gerätestatus gelogt.

Fügen Sie folgenden SPS-Code in Ihr SPS-Projekt ein:

```
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
```

Die Dokumentation aller Systemparameter finden Sie hier: ST_IEC870_5_101SystemParams [▶ 315].

12.1.7 Stationsabfrage

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Der Stationsabfragebefehl wird von der Zentralstation eingeleitet. Im Kennungsfeld des Befehls ist auch die Gruppe (1 bis 16 oder allgemein) festgelegt. Die Unterstation überträgt die zu dieser Gruppe dazugehörigen Applikationsobjekte mit der Übertragungsursache *<20>* bis *<36>* an die Zentralstation. Applikationsobjekte mit Zeitmarken werden ohne Zeitmarken übertragen.

Konfiguration der Systemparameter

- Es müssen keine speziellen Systemparameter gesetzt werden;

Konfiguration der Applikationsobjekte

- Der Datenpunkt muss einer oder mehreren Groupen zugeordnet werden. Der Groupparameter muss gesetzt werden. Eine Übersicht aller verfügbaren Grouppen finden Sie hier: Group configuration flags [▶ 347].

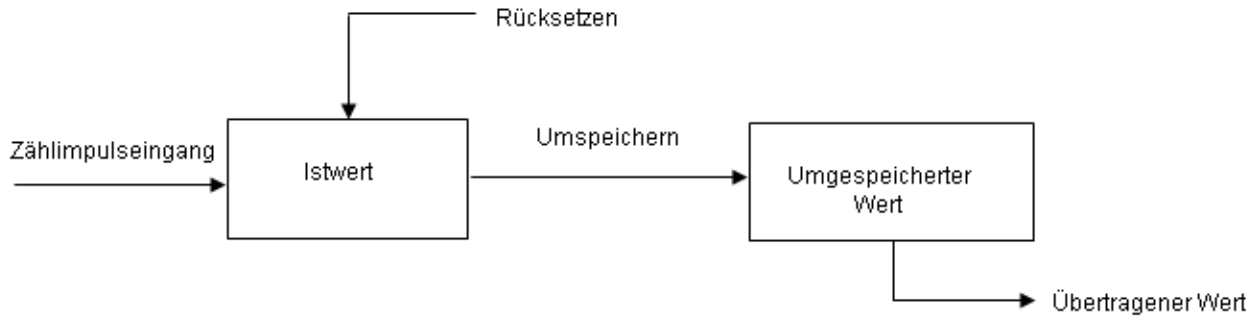
Beispielkonfiguration für einen Datenpunkt der der Gruppe: 1 und der Gruppe: Allgemein zugerodnet wurde.

```
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 Zählwertübertragung (counter interrogation)

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Allgemeines Modell für die Zählwertübertragung:



Die Istwerte werden durch Zähler aufsummiert. Die Istwerte können durch einen Umspeicherbefehl, der entweder von der Zentralstation empfangen oder örtlich (lokal in der Unterstation) erzeugt wird, periodisch in umgespeicherte Werte umgespeichert (kopiert) werden. Nach dem Umspeichern wird der erfasste Wert entweder auf Null zurückgesetzt (Erfassen von Inkrementalwerten) oder der Zähler fährt mit seinem Betrieb fort (Erfassen von Zählerständen).

Applikationsobjekte mit Zählwerten werden Gruppen zugeordnet. Die Gruppen werden einzeln umgespeichert (frozen), zurückgesetzt (reset) oder übertragen. Die Zentralstation sendet Zählwertabfragebefehle an die Unterstation. In einem Kennungsfeld des Befehls (QCC) wird die durchzuführende Aktion (FRZ) und Gruppe (RQT) festgelegt.

Die Zuordnung der Applikationsobjekte zu den einzelnen Gruppen (1 bis 4 oder allgemein) wird während der Konfiguration durch den Group-Flagparameter festgelegt. Es gibt vier Betriebsarten für die Erfassung von Zählerständen und Inkrementalwerten. Zu jeder Betriebsart sind einige Hinweise zur Konfiguration der Systemparameter oder der Applikationsobjekte aufgeführt.

Betriebsart A: Örtlich Umspeichern mit Spontanübertragung

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden spontan übertragen, nachdem die Funktion Umspeichern oder Umspeichern mit Rücksetzen durchgeführt wurde. Die Zentralstation gibt in dieser Betriebsart keine Zählwertabfragebefehle aus.

Konfiguration der Systemparameter:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

Der erste Parameter aktiviert das örtliche Umspeichern oder Umspeichern mit Rücksetzen. Der zweite Parameter gibt die Zykluszeit an in der das Umspeichern oder Umspeichern mit Rücksetzen durchgeführt wird (z.B.: alle 60 Sekunden).

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF-Groupparameter darf nicht gesetzt werden. Dieser würde die spontane Datenübertragung der Zählwerte verhindern.
- Der Zählwert wird umgespeichert, wenn IEC870_GRP_LOCFREEZE-Groupparameter gesetzt wurde.
- Der Zählwert wird zurückgesetzt, wenn IEC870_GRP_LOCRESET-Groupparameter gesetzt wurde.
- Das Örtliche Umspeichern oder Umspeichern mit Rücksetzen wird gleichzeitig für alle Gruppen (1 bis 4 oder allgemein) durchgeführt.

Betriebsart B: Örtliches Umspeichern mit Zählerabfrage

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden durch Zählwertabfragebefehle von der Zentralstation abgefragt. In diesem Fall darf die Zentralstation im Befehlskennungsfeld das Umspeichern oder Umspeichern mit Rücksetzen nicht benutzen (FRZ=0). Die Zählwerte werden allgemein oder in Gruppen (groups) 1 bis 4 abgefragt.

Konfiguration der Systemparameter:

```
bPerFRZ  := TRUE
tPerFRZCycle := T#60s
```

Der erste Parameter aktiviert das örtliche Umspeichern oder (und) Rücksetzen. Der zweite Parameter gibt die Zykluszeit an in der das umspeichern oder Umspeichern mit Rücksetzen durchgeführt wird (z.B.: alle 60 Sekunden).

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF-Groupparameter muss gesetzt werden. Die Zählwerte sollen nicht spontan zur Zentralstation übertragen werden.
- Der Zählwert wird umgespeichert wenn IEC870_GRP_LOCFREEZE-Groupparameter gesetzt wurde.
- Der Zählwert wird zurückgesetzt wenn IEC870_GRP_LOCRESET-Groupparameter gesetzt wurde.
- Das örtliche Umspeichern oder Umspeichern mit Rücksetzen wird gleichzeitig für alle Gruppen (1 bis 4 oder allgemein) durchgeführt.

Betriebsart C: Zentralstation leitet das Umspeichern, Umspeichern mit Rücksetzen oder Rücksetzen ein

Ein Zählwertabfragebefehl wird periodisch von der Zentralstation an die Unterstation ausgegeben, um das Umspeichern oder (und) Rücksetzen zu steuern. Dieser Befehl hat aber noch keine Übertragung der Zählwerte zur Folge. Erst ein nachfolgender Zählwertabfragebefehl wird von der Zentralstation gesendet um die umgespeicherten Zählwerte einzusammeln. Ähnlich, wie bei der Betriebsart B.

Konfiguration der Systemparameter:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Das örtliche Umspeichern oder (und) Rücksetzen muss deaktiviert werden. Der zweite Parameter wird ignoriert.

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF muss gesetzt werden. Die Zählwerte sollen nicht spontan zur Zentralstation übertragen werden.
- IEC870_GRP_LOCFREEZE- und IEC870_GRP_LOCRESET-Groupparameter dürfen nicht gesetzt werden. Die Zentralstation leitet das Umspeichern oder (und) Rücksetzen ein.
- Die Zählwerte können einzelnen Gruppen (1 bis 4 oder allgemein) zugeordnet und abgefragt werden (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1,IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

Betriebsart D: Zentralstation leitet das Umspeichern und (oder) Rücksetzen ein und die umgespeicherten Werte werden spontan übertragen

Diese Betriebsart ist eine Kombination des Zählwertbefehls von der Zentralstation wie für Betriebsart C mit einer spontanen Übertragung der Zählwerte wie bei der Betriebsart A.

Konfiguration der Systemparameter:

```
bPerFRZ  := FALSE
tPerFRZCycle := T#60s
```

Das örtliche Umspeichern oder (und) Rücksetzen muss deaktiviert werden. Der zweite Parameter wird ignoriert.

Konfiguration der Applikationsobjekte:

- IEC870_GRP_SPONTOFF-Groupparameter darf nicht gesetzt werden. Dieser würde die spontane Datenübertragung der Zählwerte verhindern.
- IEC870_GRP_LOCFREEZE- und IEC870_GRP_LOCRESET-Groupparameter dürfen nicht gesetzt werden. Die Zentralstation leitet das Umspeichern oder (und) Rücksetzen ein.
- Die Zählwerte können einzelnen Gruppen (1 bis 4 oder allgemein) zugeordnet und abgefragt werden (IEC870_GRP_REQCOGEN, IEC870_GRP_REQCO1,IEC870_GRP_REQCO2, IEC870_GRP_REQCO3, IEC870_GRP_REQCO4).

12.1.9 Uhrzeitsynchronisation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Das Verhalten, wie die Systemzeit der Unterstation synchronisiert werden soll, ist über die Systemparameter konfigurierbar.

- Die Systemzeit der Unterstation kann während der Initialisierung mit der Systemzeit des lokalen TwinCAT PC's synchronisiert werden;
- Beim Empfang eines Uhrzeit-Synchronisationsbefehls von der Zentralstation kann die Systemzeit der Unterstation ebenfalls synchronisiert werden;
- Die Systemzeit des lokalen TwinCAT PC's kann beim Empfang eines Uhrzeit-Synchronisationsbefehls auch synchronisiert werden.

12.1.10 Hintergrundabfrage

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Die Hintergrundabfrage wird zum Auffrischen der Prozessinformationen von der Unterstation zur Zentralstation als zusätzlicher Sicherheitsbeitrag zur Stationsabfrage und spontanen Übertragung angewendet.
Applikationsobjekte mit Typkennungen wie für die Stationsabfrage, dürfen mit der Übertragungsursache <2> Hintergrundabfrage stetig mit niedriger Priorität übertragen werden. Die zulässigen ASDU-Typkennungen sind in der Kompatibilitätsliste der Station aufgelistet (Tabelle Typkennung <-> Übertragungsursache). Die Hintergrundabfrage wird von der Unterstation eingeleitet und ist unabhängig von den Stationsabfragebefehlen.

Konfiguration der Systemparameter

Der Übertragungszyklus wird durch Systemparameter [▶ 315] in der Unterstation festgelegt.

```
bBackScan := TRUE;
tBackScanCycle := T#30s;
```

Konfiguration der Applikationsobjekte

Applikationsobjekte, deren Prozessdaten als Hintergrundabfrage übertragen werden sollen, müssen mit dem Group-Flag: IEC870_GRP_BACKGROUND konfiguriert werden.

Beispiel:

```
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 Zyklische Datenübertragung

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Die zyklische Datenübertragung wird ähnlich wie die Hintergrundabfrage von der Unterstation eingeleitet und ist unabhängig von anderen Befehlen aus der Zentralstation. Durch die zyklische Datenübertragung werden die Prozessdaten der Zentralstation kontinuierlich aufgefrischt. Bei den Prozessdaten handelt es sich meistens um Messwerte, die in regulären Zeitabständen erfasst werden. Die zyklische Datenübertragung

wird oft benutzt um nicht-zeitkritische, oder sich nicht so schnell ändernde Prozessdaten zu überwachen (z.B. ein Temperatursensor). Die zyklischen/periodischen Daten werden zur Zentralstation mit der Übertragungsursache *<1> Periodic/Cyclic* übertragen. Die zulässigen ASDU-Typkennungen sind in der Kompatibilitätsliste der Station aufgeführt (Tabelle Typkennung <-> Übertragungsursache). Die zyklische Datenübertragung kann über die Systemparameter und die Konfigurationsparameter der Applikationsobjekte konfiguriert werden.

Konfiguration der Systemparameter:

```
bPerCyclic       : BOOL := TRUE;
tPerCyclicBase   : TIME := T#60s;
```

Der erste Parameter aktiviert die zyklische Übertragung. Der zweite Parameter ist die Basiszeit der zyklischen/periodischen Datenübertragung (hier z.B. 60 Sekunden).

Konfiguration der Applikationsobjekte:

- IEC870_GRP_PERCYC-Groupparameter muss gesetzt werden;
- Der Multiplikator-Parameter *(multiplier)* der F_iecInitAOEntry-Funktion muss auf einen Wert <> Null gesetzt werden. Beispiel: Bei einem Multiplikator = 2 und Basiszeit von 60 Sekunden werden die die Prozessdaten des Applikationsobjekts alle 120 Sekunden zur Zentralstation gesendet;

Beispielkonfiguration für einen Messwert der zyklisch alle 120 Sekunden zur Zentralstation übertragen werden soll (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 Befehlsübertragung

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Befehle können von der Zentralstation in Steuerungsrichtung (zur Unterstation) gesendet werden.
Ein Einzelbefehl mit der Typkennung 45 (C_SC_NA_1) wird zur Steuerung eines Applikationsobjekts benutzt, das in Überwachungsrichtung als Einzelmeldung (M_SP_NA_1, M_SP_TA_1 oder M_SP_TB_1) übertragen wird. Ein Doppelbefehl (C_DC_NA_1) wird zur Steuerung eines Applikationsobjekts benutzt, das in Überwachungsrichtung als Doppelmeldung (M_DP_NA1, M_DP_TA_1 oder M_DP_TB_1) übertragen wird, usw.

Konfiguration der Systemparameter:

- Es müssen keine speziellen Systemparameter gesetzt werden;

Konfiguration der Applikationsobjekte:

- Die Applikationsobjekte müssen als Befehle (Datentypen in Steuerungsrichtung) konfiguriert werden;
- Die Addressen der Informationsobjekte (IOA's) müssen den Adressen in der Leitstation entsprechen;

Beispiele:

Single command mit der IOA = 10. Der empfangene Wert wird in den Merkerbereichspuffer, Byteoffset = 100, Bitoffset = 0 kopiert.

```
F_iecInitAOEntry( C_SC_NA_1, 10, 0, 0, MAP_AREA_MEMORY, 100, 0, AODB[24] );
```

Double command mit der IOA = 20. Der empfangene Wert wird in den Merkerbereichspuffer, Byteoffset = 200, Bitoffset = 0..1 kopiert.

```
F_iecInitAOEntry( C_DC_NA_1, 20, 0, 0, MAP_AREA_MEMORY, 200, 0, AODB[27] );
```

Set point, scaled value mit der IOA = 60. Der empfangene Wert wird in den Merkerbereichspuffer, Byteoffset = 600..601, Bitoffset = 0 kopiert.

```
F_iecInitAOEntry( C_SE_NB_1, 60, 0, 0, MAP_AREA_MEMORY, 600, 0, AODB[39] )
```

12.1.13 Abfrage-/Lese-Prozedur

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Die Zentralstation sendet Abfragebefehle an die Unterstation.
In dem Abfragebefehl wird die Adresse des abzufragenden Applikationsobjekts übertragen. Die Daten dieses Applikationsobjekts sollen an die Zentralstation gesendet werden. Die Unterstation sendet die Daten mit der Übertragungsursache *<5> Abfrage oder abgefragt*. Die zulässigen ASDU-Typkennungen sind in der Kompatibilitätsliste der Station aufgelistet (Tabelle Typkennung <-> Übertragungsursache).

Konfiguration der Systemparameter

- Es müssen keine speziellen Systemparameter gesetzt werden;

Konfiguration der Applikationsobjekte

- Es müssen keine speziellen Parameter gesetzt werden;

12.1.14 Doppelübertragung

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Alle Applikationsobjekte (Informationsobjekte), die mit der Übertragungsursache *<3> Spontan* übertragen werden, dürfen zweimal übertragen werden, mit und ohne Zeitmarke. Diese Betriebsart wird "Doppelübertragung" genannt.
Doppelübertragung wird von der Unterstation zur Zeit nicht unterstützt.

12.1.15 Quality-Flags

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Die Quality-Flags (Quality-Descriptor) liefern der Zentralstation zusätzliche Informationen zur Qualität eines Applikationsobjekts. Die Quality-Flags können aus der SPS-Applikation mit Hilfe der F_iecSetAOQuality [▶ 279]-Funktion unabhängig voneinander gesetzt/zurückgesetzt werden. Mit der F_iecGetAOQuality [▶ 280]-Funktion kann der Status der Quality-Flags abgefragt werden. Jede Änderung der Quality-Flags führt zu einer spontanen Übertragung der Daten zur Zentralstation.

Folgende Quality-Flags werden intern von der Unterstation zur Laufzeit ausgewertet:

- IECQ_BL_ON (Blocked). Wurden die Prozessdaten des Applikationsobjekts für die Übertragung blockiert, dann wird das Mapping der SPS- und IEC-Prozessdaten für dieses Applikationsobjekt nicht ausgeführt;

Folgende Quality-Flags werden intern von der Unterstation zur Laufzeit gesetzt/zurückgesetzt:

- IECQ_IV_ON (Invalid). Die Unterstation setzt das Invalid-Flag wenn das Mapping der SPS- und IEC-Prozessdaten nicht durchgeführt werden konnte (z.B. wegen fehlerhafter Konfiguration des Applikationsobjekts). Dieses Verhalten kann durch einen gesetzten Group-Parameter: IEC870_GRP_IV_OFF deaktiviert werden.

Alle anderen Quality-Flags werden unverändert zur Zentralstation gesendet.

12.1.16 Test der Kommunikation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Durch das Setzen der *bChangeIO*-Variable auf TRUE wird eine einfache Simulation der Datenpunkte in Überwachungsrichtung aktiviert und mit FALSE deaktiviert. Bei einer aktiven Verbindung werden die Werte Zyklisch alle 3 Sekunden zur Leitstation übertragen.

```
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 Übertragungs- und Kommunikationsfehler

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip

Die Stationsfehlermeldungen werden in einem FIFO abgelegt. Es können bis zu 10 Fehlermeldungen zwischengespeichert werden. Bei fatalen Kommunikationsfehlern (z.B. Fehler der Verbindungsschicht, die Checksumme des Frames passt nicht) wird die Verbindung unterbrochen und muss neu aufgebaut werden. Fehler in der Applikationsschicht (z.B. der ASDU-Sendepuffer ist wegen zu vieler Frames übergelaufen) werden nur geloggt und führen nicht zum Verbindungsabbruch. Es immer noch möglich auch bei diesen Fehlern die Verbindung aus der Applikation zu unterbrechen. Neben dem Fehler-Code wird auch die Fehlerquelle in der Fehlermeldung abgelegt. Dieses erleichtert die Lokalisierung des Fehlers.

Beispiel

Die anfallenden Fehlermeldungen einer IEC 60870-5-104 Unterstation können durch folgenden Aufruf ausgelesen werden:

```
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 SPS-API

Einfache Projekte mit kompletten Sourcen finden Sie hier: IEC60870-5-104 Unterstation [▶ 479].

Kompatibilitätsliste finden Sie hier: Interoperability check list

Übersicht der Fehlercodes finden Sie hier: Fehlercodes

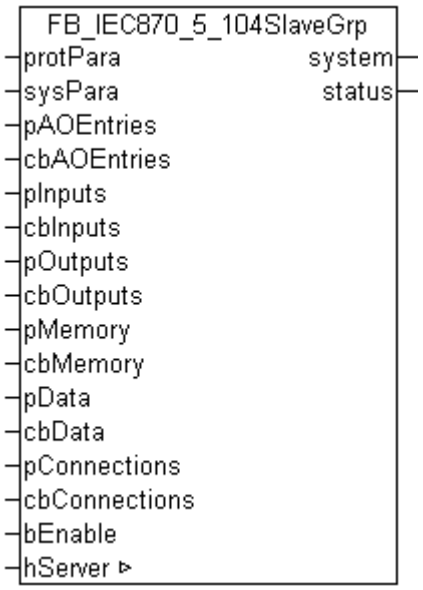
Eine ausführliche Anleitung zur Implementierung der Unterstation in der SPS finden Sie hier: TUTORIAL [▶ 453]

Kurzanleitung

1. Erstellen Sie ein neues SPS-Projekt erstellen binden Sie die SPS-Bibliothek **TcIEC870_5_104Slave.Lib** ein.
2. Legen Sie im Hauptprogramm eine Instanz der T_HSERVER-Variablen an (Verbindungs-Handle) und initialisieren Sie im Programmteil einmalig durch den Aufruf der F_CreateServerHnd-Funktion mit passenden Parametern.
3. Konfiguration der Datenpunkte: Legen Sie eine Array-Variable vom Typ ST_IEC870_5_101AODBEntry [▶ 312] an. Jedes Arrayelement entspricht einem Datenpunkt. Konfigurieren Sie die Datenpunkte mit Hilfe der Funktion F_iecInitAOEntry [▶ 277] zur Laufzeit (z.B. in einem Init-Schritt).
4. Legen Sie im Hauptprogramm eine Instanz des Protokoll-Bausteins FB_IEC870_5_104Slave [▶ 474] an, konfigurieren Sie diese und rufen Sie diese auf.
   Konfigurieren Sie die System- und Protokoll-Parameter passend zu den Parametern der Leitstation.

12.2.1 FB_IEC870_5_104SlaveGrp

Ab Produktversion: TwinCAT PLC Library: IEC60870-5-104 controlled station (slave) v2.0.2 und höher.



Mit einer Instanz des Funktionsbausteins FB_IEC870_5_104SlaveGrp kann in der TwinCAT SPS eine IEC60870-5-104 Unterstation (Slave) implementiert werden. Mit einer Instanz können einfache Systeme mit einer Verbindung oder Systeme mit mehreren Verbindungen zum Master realisiert werden (redundantes System).

Die maximale Anzahl der Verbindungen kann durch die Anzahl der ST_IEC870_5_104ServerConnection [▶ 477]-Arrayelemente festgelegt werden. Die Adresse der Arrayvariablen und die Bytegröße der Arrayvariablen müssen dann an die Instanz des FB_IEC870_5_104SlaveGrp-Funktionsbausteins übergeben werden.

VAR_IN_OUT

```
VAR_IN_OUT
    hServer : T_HSERVER;
END_VAR
```

hServer : TCP/IP Server-Handle. Die internen Parameter der Server-Handle-Variable müssen vorher mit der Funktion F_CreateServerHnd initialisiert werden.

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-Protokolparameter [▶ 446].

sysPara: Systemparameter [▶ 315].

pAOEntries: Adresse der Applikationsobjekt-Datenbankvariablen.

cbAOEntries: Bytegröße der Applikationsobjekt [▶ 312]-Datenbankvariablen.

pInputs: Adresse des SPS-Prozessdatenbereichs der Eingänge.

cbInputs: Bytegröße des SPS-Prozessdatenbereichs der Eingänge.

pOutputs: Adresse des SPS-Prozessdatenbereichs der Ausgänge.

cbOutputs: Bytegröße des SPS-Prozessdatenbereichs der Ausgänge.

pMamory: Adresse des SPS-Prozessdatenbereichs der Merker.

cbMamory: Bytegröße des SPS-Prozessdatenbereichs der Merker.

pData: Adresse des SPS-Datenbereichs.

cbData: Bytegröße des SPS-Datenbereichs.

pConnections: Adresse der ST_IEC870_5_104ServerConnection [▶ 477]-Arrayvariablen.

cbConnections: Bytegröße der ST_IEC870_5_104ServerConnection-Arrayvariablen.

bEnable : Aktiviert/Deaktiviert den Funktionsbaustein (Kommunikation und Verbindungen).

Die Adressen können mit dem ADR- und die Bytegrößen mit dem SIZEOF-Operator ermittelt werden.

VAR_OUTPUT

```
VAR_OUTPUT
    system      : ST_IEC870_5_104SystemInterface;
    status      : ST_IEC870_5_104GrpStatus;
ND_VAR
```

system: System-Interface [▶ 478]. Diese Variable dient anderen IEC-Applikationsfunktionen als Kommunikationsschnittstelle zum IEC-Gerät (hier: Unterstation).

- Membervariable *system.device* wird z.B. von der F_iecSetAOQuality [▶ 279]-Funktion als VAR_IN_OUT-Parameter erwartet.
- Membervariable *system.device.errors* ist ein Gerätefehler-Fifo. Die registrierten Fehler können von der SPS-Applikation ausgelesen und ausgewertet werden.

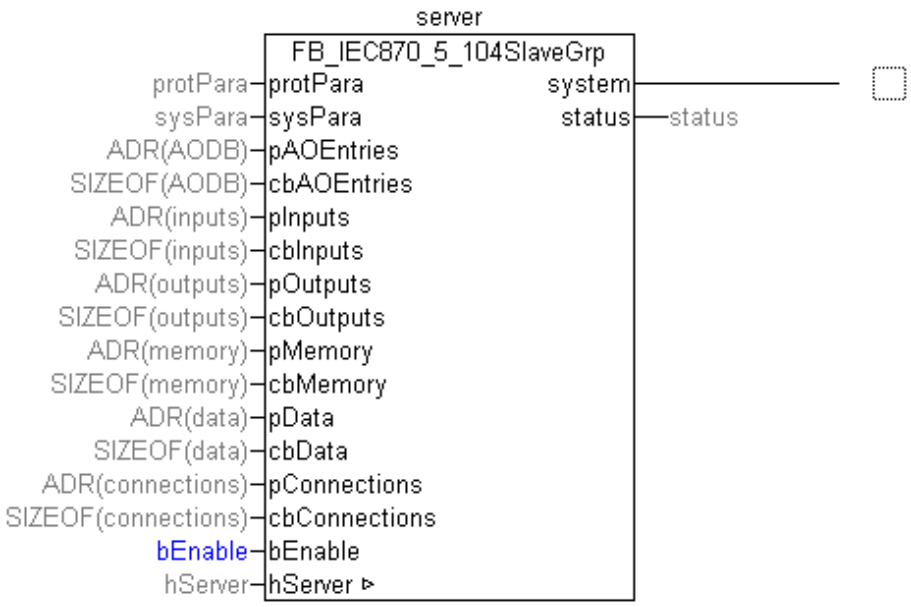
status: Verbindungs- und Datentransfer-Statusinformationen [▶ 477].

Beispiel:

Beispielprojekte: IEC60870-5-104 Unterstation [▶ 479]

Aufruf in FUP mit max. 2 Master-Verbindungen:

```
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
```



Im folgenden ST-Beispiel wird der Gerätefehler-Fifo zyklisch ausgelesen und die registrierten Fehler ins Windows Application Log geschrieben.

```
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden

12.2.2 FB_IEC870_5_104Slave

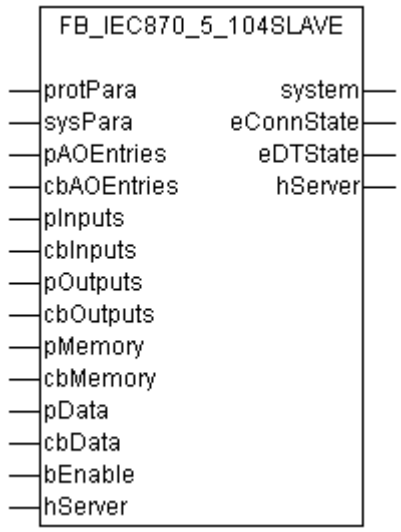


Abb. 7: FB_IEC870_5_104Slave

Mit einer Instanz des Funktionsbausteins FB_IEC870_5_104Slave kann in der TwinCAT SPS eine IEC60870-5-104 Unterstation (Slave) implementiert werden. Eine Instanz des Funktionsbausteins kann nur eine Verbindung zu einem Master aufbauen.

Erstellen von redundanten Systemen

Wenn Sie redundante Systeme mit zwei oder mehr Verbindungen zum Master realisieren wollen, benutzen Sie bitte den FB_IEC870_5_104SlaveGrp [▶ 470]-Funktionsbaustein.

VAR_IN_OUT

```
VAR_IN_OUT
    hServer : T_HSERVER;
ND_VAR
```

hServer : TCP/IP Server-Handle. Die internen Parameter der Server-Handle-Variable müssen vorher mit der Funktion F_CreateServerHnd initialisiert werden.

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;    (* P
ointer to the first element of application database object array *)
    cbAOEntries     : UDINT;                              (* Byte size (length) of application datab
ase 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-Protokolparameter [▶ 446].

sysPara: Systemparameter [▶ 315].

pAOEntries: Adresse der Applikationsobjekt [▶ 312]-Datenbankvariablen.

cbAOEntries: Bytegröße der Applikationsobjekt-Datenbankvariablen.

pInputs: Adresse des SPS-Prozessdatenbereichs der Eingänge.

cbInputs: Bytegröße des SPS-Prozessdatenbereichs der Eingänge.

pOutputs: Adresse des SPS-Prozessdatenbereichs der Ausgänge.

cbOutputs: Bytegröße des SPS-Prozessdatenbereichs der Ausgänge.

pMamory: Adresse des SPS-Prozessdatenbereichs der Merker.

cbMamory: Bytegröße des SPS-Prozessdatenbereichs der Merker.

pData: Adresse des SPS-Datenbereichs.

cbData: Bytegröße des SPS-Datenbereichs.

bEnable : Aktiviert/Deaktiviert den Funktionsbaustein (Kommunikation und Verbindung).

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]. Diese Variable dient anderen IEC-Applikationsfunktionen als Kommunikationsschnittstelle zum IEC-Gerät (hier: Unterstation).

- Membervariable *system.device* wird z.B. von der F_iecSetAOQuality [▶ 279]-Funktion als VAR_IN_OUT-Parameter erwartet.
- Membervariable *system.device.errors* ist ein Gerätefehler-Fifo. Die registrieten Fehler können von der SPS-Applikation ausgelesen und ausgewertet werden.

eConnState: Status der TCP/IP-Verbindung zum Master.

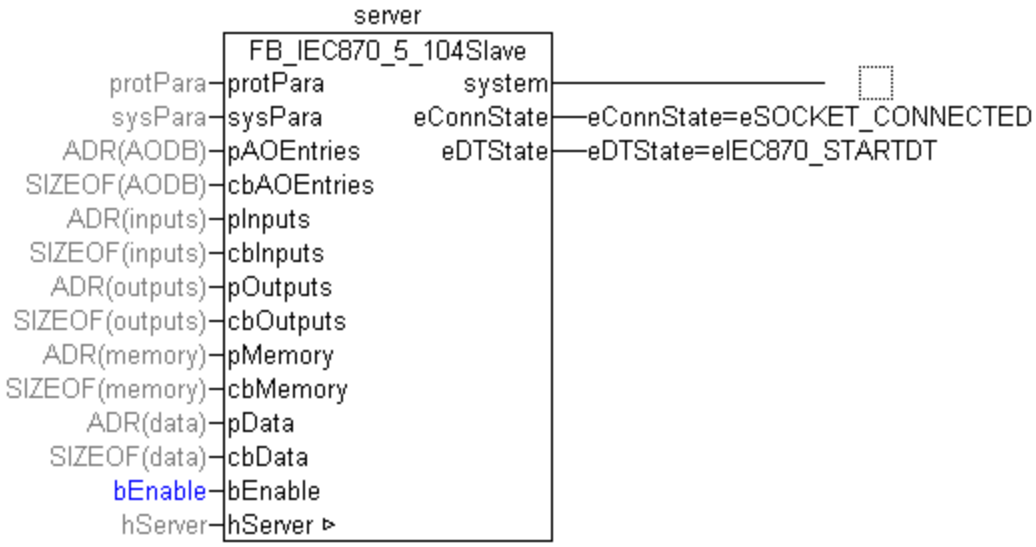
eDTState: Status [▶ 448] des IEC60870-5-104-Datenaustauschs (STARTDT, STOPDT)

Beispiel:

Beispielprojekte: IEC60870-5-104 Unterstation [▶ 479]

Aufruf in FUP:

```
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
```



Im folgenden ST-Beispiel wird der Gerätefehler-Fifo zyklisch ausgelesen und die registrierten Fehler ins Windows Application Log geschrieben.

```
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
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden

12.2.3 F_GetVersionTcIEC870_5_104Slave

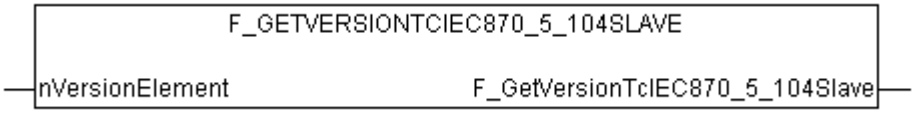


Abb. 8: F_GetVersionTcIEC870_5_104Slave

Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_104Slave: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

12.2.4 ST_IEC870_5_104ServerConnection

Ab Produktversion: TwinCAT PLC Library: IEC60870-5-104 controlled station (slave) v2.0.2 und höher.

Eine Variable von diesem Typ repräsentiert eine IEC870-5-104 Server-Verbindung.

Beispiel einer Deklaration für eine dreifache Verbindung:

```
connections : ARRAY[0..3] OF ST_IEC870_5_104ServerConnection;
```

Beispiel einer Deklaration für eine einfache Verbindung:

```
connections : ARRAY[0..0] OF ST_IEC870_5_104ServerConnection;
```

HINWEIS

Die Strukturelemente sollen nicht direkt beschrieben oder verändert werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

12.2.5 ST_IEC870_5_104GrpStatus

Ab Produktversion: TwinCAT PLC Library: IEC60870-5-104 controlled station (slave) v2.0.2 und höher,

Status einer IEC870-5-104 Slave-Gruppe.

```
TYPE ST_IEC870_5_104GrpStatus:
STRUCT
    nConnected : DWORD := 0;
    nSuspended : DWORD := 0;
```

```
    nDTStarted : DWORD := 0;
END_STRUCT
END_TYPE
```

nConnected: Anzahl der hergestellten TCP/IP-Verbindungen (ESTABLISHED).

nSuspended: Anzahl der Verbindungen deren Zustand gerade wechselt (CONNECTED->DISCONNECTED oder DISCONNECTED->CONNECTED).

nDTStarted: Anzahl der Verbindungen mit aktivem Datentransfer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden)
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: Kommunikationsschnittstelle [▶ 318] des IEC-Gerätes.

service: IEC-Gerätedienst;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
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 werden automatisch eingebunden
TwinCAT v2.10.0 Build >= 1301	CX (ARM)	

12.3 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE-Image-Version usw.).

2. Vergleichen/Überprüfen Sie die Kompatibilitätsliste der Unterstation mit der Kompatibilitätsliste der Leitstation.

3. Überprüfen Sie die Installationshinweise (z.B. Installation der CAB-Files auf einem CE-System).

4. Bei Verbindungsproblemen kann der PING-Befehl dazu benutzt werden, um festzustellen, ob der Kommunikationspartner über die Netzwerkverbindung erreichbar ist. Wenn dies nicht der Fall ist, überprüfen Sie die Netzwerkkonfiguration und die Firewall-Einstellungen.

5. Überprüfen Sie, ob die Eingangsparameter, die Sie an die F_CreateServerHnd() Funktion übergeben richtig sind (Netzwerkadresse, Portnummer, usw.).

6. Überprüfen Sie, ob der Funktionsbaustein einen Fehlercode ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes.

7. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 446] (iK, iW, t0, t1, t2, t3, APDULength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern in der Leitstation.

8. Überprüfen Sie die am Funktionsbaustein eingestellten Systemparameter [▶ 315] (ASDU-Adresse, Länge der ASDU-Adresse, Länge der Informationsobjekt-Adresse, Länge der Übertragungsursache COT, usw.). Vergleichen Sie die Systemparameter mit den Parametern in der Leitstation.

9. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, Adresse des Informationsobjektes usw.).

10. Überprüfen Sie, ob die Leitstation einen Fehlercode ausgibt.

11. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU-Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben.

12. Überprüfen Sie die Verwendung des FB_SocketCloseAll()-Funktionsbausteins und des LISTEN_MODE_CLOSEALL-Parameters wenn Sie über mehrere TCP/IP Verbindungen (Server/Clients) in einem Laufzeitsystem kommunizieren.

Bei mehreren Verbindungen aktivieren Sie nur eine Instanz des FB_SocketCloseAll()-Funktionsbausteins einmalig im Initialisierungsschritt beim Programmstart. Der LISTEN_MODE_CLOSEALL-Parameter darf in diesem Fall nicht mehr verwendet werden.

13. Eine komplette Aufzeichnung der Netzwerkkommunikation kann mit Sniffer-Tools wie Wireshark durchgeführt werden. Die Aufnahme kann dann vom Beckhoff-Supportpersonal analysiert werden.

12.4 Beispiele

In Beispielprojekten sind folgende Stationsparameter für die Unterstation eingestellt:

- Local (server) host address: **127.0.0.1** (Sie müssen mindestens diesen Parameter an Ihre Zielplattform anpassen!)
- 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**

Voraussetzungen

SPS Project	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737010443.zip	TwinCAT IEC60870-5-104 Unterstation mit einer Verbindung zum Master. Ein **minimalistisches** SPS-Projekt mit einem Single-Point in Überwachungsrichtung (IOA = 100) und einem Single-Command in Steuerungsrichtung (IOA = 10).

SPS Project	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737009035.zip	TwinCAT IEC60870-5-104 Unterstation mit einer Verbindung zum Master. Ein komplexeres SPS-Projekt mit unterschiedlichen Datenpunkten in beide Richtungen.
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11737011851.zip	TwinCAT IEC60870-5-104 Unterstation mit zwei Verbindungen zum Master. Ein komplexeres SPS-Projekt mit unterschiedlichen Datenpunkten in beide Richtungen.

13 TcIEC870_5_104Master: IEC 60870-5-104 Zentralstation (master)

Mit den SPS-Funktionen und -Funktionsbausteinen können Zentralstationen (Master) nach der IEC 60870-5-104 Norm in TwinCAT SPS realisiert werden.

Die SPS-Bibliothek verfügt über zwei Software-Schnittstellen. Die Endapplikation setzt auf einer der Schnittstellen auf. Die Wahl der Schnittstelle hängt von den Anforderungen an die Endapplikation ab. Im Folgenden werden die Eigenschaften beider Schnittstellen kurz beschrieben.

"High level"-Schnittstelle: IEC 60870-5-104 Zentralstation

Bei dieser Schnittstelle handelt es sich um eine sogenannte "Ein-Baustein-Lösung". Alle Funktionalitäten sind in einem SPS-Baustein gekapselt. Der Baustein implementiert die wichtigsten Dienste und Funktionen. Diese Implementierung ist für über 90% der Anwendungen ausreichend.

Pro: Sehr kleiner SPS-Programmieraufwand um eine laufende Applikation zu erhalten; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw sind bereits in dem Baustein implementiert und werden automatisch ausgeführt; Das Mapping der IEC<->SPS Prozessdaten und das der Datenpunkte wird über Funktionsaufrufe konfiguriert; Der SPS-Programmierer muss nicht sehr gut mit der Protokollnorm vertraut sein;

Contra: Die SPS-Applikation hat nur einen geringen Einfluss auf die Protokollausführung; Kein Einfluss auf die Ausführung der Dienste, diese werden intern automatisch ausgeführt; Zeitstempel werden von dem Baustein automatisch generiert und können nicht verändert (von extern übergeben) werden; Es ist z.B. nur die direkte Befehlsausführung möglich; Schlechtere Performance bei vielen Datenpunkten.

Diese Schnittstelle empfiehlt sich wenn Sie:

- Mit der Protokollnorm nicht vertraut sind;
- Eine einfache Applikation mit einer handvoll Datenpunkten implementieren möchten (<1000);
- Keine grossen Performace-Anforderungen an die Applikation stellen;
- Keine besondere Befehlsausführung wie Select/Execute oder Daten + Zeistempel von externen Geräten versenden möchten;
- Keine Funktionalitäten benötigen die laut Kompatibilitätsliste nicht unterstützt werden;

"Low level"-Schnittstelle: IEC 60870-5-104 Transport Interface

Diese Schnittstelle setzt bei dem Protokollstack tiefer an und ermöglicht das Versenden und/oder Empfangen einzelner Frames (ASDU's).

Pro: Sehr flexibel; Alle Eigenschaften im ASDU-Frame können manipuliert werden (z.B. ein eigener Zeitstempel, Select/Execute oder eine besondere Befehlsausführung usw); Weil nur die benötigten Dienste implementiert werden kann eine hohe Performance erreicht werden; Hohe Performace bei vielen Datenpunkten;

Contra: Grösserer Programmieraufwand; Alle Dienste wie Generalabfrage, Zählerabfrage, Uhrzeitsynchronisation, Befehlsausführung, spontane Datenübertragung usw. müssen von dem SPS-Programmierer selbst implementiert (ausprogrammiert) werden; Der SPS-Programmierer muss mit der Protokollnorm vertraut sein.

Diese Schnittstelle empfielt sich wenn Sie:

- Mit der Protokollnorm vertraut sind;
- Eine Protokollkonverter-Applikation implementieren möchten;

- In der Applikation die verfügbaren Norm-Funktionalitäten fast vollständig implementieren müssen;
- Besondere Funktionalitäten wie z.B. das weiterleiten der Zeitstempel von einem Modbusgerät oder die Kontrolle über die Befehlsausführung haben möchten;
- Funktionalitäten benötigen die laut Kompatibilitätsliste nicht unterstützt werden;
- Viele Datenpunte haben (>1000) und eine hohe Performace benötigen;

Kompatibilitätsliste

für TwinCAT SPS Bibliothek: IEC 60870-5-104 Zentralstation (bezieht sich auf die "High level"-Schnittstelle). Hier können Sie die https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11764581515.zip

Systemvoraussetzungen

Wenn Sie auf einer Programmierumgebung gleichzeitig die TwinCAT SPS-Bibliothek: IEC60870-5-104 Zentralstation und die TwinCAT SPS-Bibliothek: IEC60870-5-104 oder 101 Unterstation einsetzen:

Die Zentralstation ist nur mit der Unterstation v3.0.0 oder höher kompatibel. Der Grund: Einige SPS-Bibliotheken werden von der Zentralstation und Unterstation gemeinsam benutzt.

Programmierumgebung:

- Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
- TwinCAT Installation Level: TwinCAT PLC oder höher;
- TwinCAT System Version 2.10.0 Build >= 1301 oder höher;

Zielplattform:

- Industrie PC oder Embedded PC/CX (x86, ARM);
- Betriebssystem:
  - Windows NT (XP, XPe/WES, Vista, W7, WES7, ...);
  - Windows CE (x86) (image v1.75 oder höher);
  - Windows CE (ARM) (image v2.13 oder höher);
- TwinCAT SPS-Laufzeitsystem Version 2.9.0 oder höher;

Produktkomponenten

- **TcIEC870_5_104Master.Lib** (implementiert die Beckhoff IEC60870-5-104 Zentralstation). Diese Bibliothek muss in dem SPS-Projekt eingebunden werden. Alle anderen Bibliotheken werden automatisch hinzugefügt.
- TcIEC870_5_104.Lib (implementiert das Übertragungsprotokoll nach der IEC60870-5-104 Norm);
- TcIEC870_5_101.Lib (implementiert die Verbindungsfunktionen und gemeinsame Datentypen);
- TcSocketHelper.Lib (TCP/IP Hilfsfunktionen);
- TcpIp.Lib (TCP/IP Basisfunktionen);
- TwinCAT TCP/IP Connection Server;

Installation auf Windows NT (XP, XPe/WES, Vista, W7, WES7, ...)

Die SPS-Bibliotheken werden während der Installation in den ..\TwinCAT\Plc\Lib-Ordner hineinkopiert. Der TwinCAT TCP/IP Connection Server wird in die Liste der TwinCAT Server eingetragen. Beim TwinCAT Start wird der TCP/IP Connection Server automatisch gestartet und beim TwinCAT Stop gestoppt.

Installation auf Windows CE

Produktversion für das Laufzeitsystem unter Windows CE ist als separates Produkt verfügbar. Führen Sie folgende Schritte aus wenn Sie eine Produktversion für Windows CE erworben haben:

- Installieren Sie das Produkt zuerst wie gewohnt auf Ihrem Programmier-PC. Die SPS-Bibliotheken werden während der Installation in den ..\TwinCAT\Plc\Lib-Ordner hineinkopiert.
- X86 CPU (CX1000, CX1020, IPC):
  - Nach der Installation finden Sie im Ordner: **...\TwinCAT\CE\TCPIP\Install** ein Cabinet-File für das CE-Laufzeitsystem.
  - Kopieren Sie die darin befindliche Datei: **TcTCPIPSvrCe.I586.CAB** in einen Ordner auf dem CE-Laufzeitsystem.
- ARM CPU (CX9000):
  - Nach der Installation finden Sie im Ordner: **...\TwinCAT\CE\TCPIP\Install** ein Cabinet-File für das CE-Laufzeitsystem.
  - Kopieren Sie die darin befindliche Datei: **TcTCPIPSvrCe.ARMV4I.CAB** in einen Ordner auf dem CE-Laufzeitsystem.
- Auf dem CE-System: Installieren Sie (durch einen Doppelklick auf das das Cabinet-File) die CE-Komponenten.
- Rebooten Sie das CE-Gerät. Der TwinCAT TCP/IP Connection Server wird mit dem CE-Betriebssystem automatisch gestartet.

Dies ist nur eine kurze Produktinformation (kein vollständiges Handbuch). Bitte installieren Sie sich die vollständige Ausgabe des Beckhoff Information System.

Sie finden es

- auf sämtlichen Beckhoff-Produkt-DVDs
- auf unserem Web-Server http://www.beckhoff.com unter Download.

Beispiele

Beispiele befinden sich in der Beckhoff Information System Dokumentation der SPS-Bibliotheken.

Link zu "High level" Beispiel-Übersichtsseite: IEC 60870-5-104 Zentralstation [▶ 500];

Link zu "Low level" Beispiel-Übersichtsseite: IEC 60870-5-104 Transport Interface [▶ 449];

Weiterführende Dokumentation

- Dokumentation zur TwinCAT PLC Library ("Low level"-Schnittstelle): IEC 60870-5-104 Transport Interface [▶ 442];
- Dokumentation zur TwinCAT PLC Library: IEC 60870-5-101 Common Data Types [▶ 268];
- Dokumentation: 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 Einführung (Tutorial)

Die Einführung ist eine Anleitung wie Sie in der TwinCAT SPS eine IEC60870-5-104 Zentralstation (master) implementieren und konfigurieren können.

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

13.1.1 SPS-Projekt anlegen, SPS-Bibliotheken einbinden

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

- Starten Sie TwinCAT PLC Control.

- Mit Datei -> Neu legen Sie ein neues SPS-Projekt an. Wählen Sie als Zielsystem PC or CX (x86, ARM).

- Als nächstes wird automatisch ein neuer Programmbaustein MAIN angelegt. Wählen Sie als Sprache des Bausteins ST (Strukturierter Text). Bestätigen Sie dies.

- Aus dem Menü wählen Sie Fenster -> Bibliotheksverwaltung und dann Einfügen -> Weitere Bibliothek...

- Aus der Liste der TwinCAT Bibliotheken wählen Sie **TcIEC870_5_104Master.Lib** aus und bestätigen dies.

13.1.2 Applikationsobjekt-Datenbank der Zentralstation definieren und konfigurieren

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Applikationsobjekte = Single Points, Double Points, Measured Values, Short Floating Point Values usw.

In diesem Beispiel wurden die Befehle so konfiguriert dass die Prozessdaten der Befehle im gleichen Speicherbereich aber auf einem anderen Byte-, Bit-Offset wie die Daten der Information in Überwachungsrichtung liegen. Sie können aber auch die Befehle auf den gleichen Byte- Bit-Offset wie die Information in Überwachungsrichtung legen.

Beispiel:
C_SC_NA_1 mit IOA = 10 auf den gleichen Byte- und Bit-Offset wie M_SP_NA_1 mit IOA = 100 (beide Byte-Offset = 100 und Bit-Offset = 0). In diesem Fall würde eine Wertänderung des Single-Points in der Unterstation (M_SP_NA_1 mit der Objektadresse 100) eine Datenübertragung mit der Übertragungsursache <3> (Spontan) zur Leitstation auslösen. In der Leitstation wird der neue Wert auf die Bit- und Byte-Offsetadresse des Befehls (C_SC_NA_1 mit der Objektadresse 10) kopiert und löst eine Befehlsübertragung zur Unterstation aus.

Als Beispiel konfigurieren wir in dem Einführungsprojekt folgende Applikationsobjekte:

ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
M_SP_NA_1	100	Generalabfrage	0	Merker	100	0	1 Bit
M_SP_NA_1	101	Generalabfrage	0	Merker	100	1	1 Bit
M_SP_TB_1	102	Generalabfrage	0	Merker	100	2	1 Bit
M_DP_NA_1	200	Generalabfrage	0	Merker	200	0	2 Bits
M_DP_NA_1	201	Generalabfrage	0	Merker	200	2	2 Bits

ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
M_DP_TB_1	202	Generalabfrage	0	Merker	200	4	2 Bits
M_ST_NA_1	300	Generalabfrage	0	Merker	300	0	1 Byte
M_ST_NA_1	301	Generalabfrage	0	Merker	301	0	1 Byte
M_ST_TB_1	302	Generalabfrage	0	Merker	302	0	1 Byte
M_BO_NA_1	400	Generalabfrage	0	Merker	400	0	4 Byte
M_BO_NA_1	401	Generalabfrage	0	Merker	404	0	4 Byte
M_BO_TB_1	402	Generalabfrage	0	Merker	408	0	4 Byte
M_ME_NA_1	500	Generalabfrage	0	Merker	500	0	2 Byte
M_ME_NA_1	501	Generalabfrage	0	Merker	502	0	2 Byte
M_ME_TD_1	502	Generalabfrage	0	Merker	504	0	2 Byte
M_ME_NB_1	600	Generalabfrage	0	Merker	600	0	2 Byte
M_ME_NB_1	601	Generalabfrage	0	Merker	602	0	2 Byte
M_ME_TE_1	602	Generalabfrage	0	Merker	604	0	2 Byte
M_ME_NC_1	700	Generalabfrage	0	Merker	700	0	4 Byte
M_ME_NC_1	701	Generalabfrage	0	Merker	704	0	4 Byte
M_ME_TF_1	702	Generalabfrage	0	Merker	708	0	4 Byte
M_IT_NA_1	800	Generalzählerabfrage	0	Merker	800	0	4 Byte
M_IT_NA_1	801	Generalzählerabfrage	0	Merker	804	0	4 Byte
M_IT_TB_1	802	Generalzählerabfrage	0	Merker	808	0	4 Byte
Commands							
C_SC_NA_1	10	-	0	Merker	2100	0	1 Bit
C_SC_NA_1	11	-	0	Merker	2100	1	1 Bit
C_SC_TA_1	12	-	0	Merker	2100	2	1 Bit
C_DC_NA_1	20	-	0	Merker	2200	0	2 Bit
C_DC_NA_1	21	-	0	Merker	2200	2	2 Bit
C_DC_TA_1	22	-	0	Merker	2200	4	2 Bit

ASDU identifier	Objektadresse IOA	Group-Konfigurationsparameter	Basiszeitmultiplikator	SPS-Prozessdatenbereich	Offset Byte	Offset Bit	Prozessdatenbreite in der TwinCAT SPS
C_BO_NA_1	40	-	0	Merker	2400	0	4 Byte
C_BO_NA_1	41	-	0	Merker	2404	0	4 Byte
C_BO_TA_1	42	-	0	Merker	2408	0	4 Byte
C_SE_NA_1	50	-	0	Merker	2500	0	2 Byte
C_SE_NA_1	51	-	0	Merker	2502	0	2 Byte
C_SE_TA_1	52	-	0	Merker	2504	0	2 Byte
C_SE_NB_1	60	-	0	Merker	2600	0	2 Byte
C_SE_NB_1	61	-	0	Merker	2602	0	2 Byte
C_SE_TB_1	62	-	0	Merker	2604	0	2 Byte
C_SE_NC_1	70	-	0	Merker	2700	0	4 Byte
C_SE_NC_1	71	-	0	Merker	2704	0	4 Byte
C_SE_TC_1	72	-	0	Merker	2708	0	4 Byte

Datenbankvariable deklarieren

Die Applikationsobjekt-Datenbank ist eine Array-Variable vom Typ ST_IEC870_5_101AODBEntry [▶ 312]. Jedes Array-Element entspricht einem Applikationsobjekt. Die Array-Elemente werden aber nicht direkt, sondern nur mit Hilfe der speziell dafür zur Verfügung gestellten Funktionen und ein Datenbank-Handle (Tabellen-Handle) manipuliert. Das Datenbank-Handle muss vor der Benutzung durch einen einmaligen F_iecCreateTableHnd [▶ 293]-Funktionsaufruf initialisiert werden. Dabei werden auch die Array-Elemente miteinander als Hash-Tabelle verknüpft. Bei einer größeren Anzahl der Datenpunkte ermöglicht die Hash-Tabelle einen schnelleren Zugriff auf einen einzelnen Datenpunkt.

Die maximale Anzahl der Applikationsobjekte ist frei wählbar und nur durch den verfügbaren Speicher begrenzt. Sie müssen sich auf eine konstante maximale Anzahl während der SPS-Programmierung festlegen. Zur Laufzeit kann die maximale Anzahl der Applikationsobjekte nicht mehr verändert werden. In unserem Beispiel werden 50 Applikationsobjekte deklariert. Diese Anzahl reicht für die meisten Anwendungen aus. Beachten Sie, dass sehr viele Applikationsobjekte auch entsprechend viel Speicher und Laufzeit benötigen.

Definieren Sie folgende Variablen in MAIN:

```
PROGRAM MAIN
VAR
    AODB        : ARRAY[0..49] OF ST_IEC870_5_101AODBEntry;
    hTable      : T_HAODBTable;
END_VAR
```

Applikationsobjekte konfigurieren

Die Konfiguration der gewünschten Applikationsobjekte wird zur Programmlaufzeit durchgeführt. Während der Konfiguration werden unter anderem der Objekt-Typ (M_SP_NA_1, M_DP_NA_1, M_ST_NA_1 usw.), die gemeinsame ASDU-Adresse, die Objekt-Adresse und weitere Objekt-Parameter festgelegt.

Nach der Initialisierung des Datenbank-Handles ist die Applikationsobjekt-Datenbank (Datenbank-Array) leer und muss mit den gewünschten Daten (Datenpunkten) gefüllt werden. Die Konfiguration der Datenpunkte der Zentralstation muss der Konfiguration der Datenpunkte in der Unterstation entsprechen! D.h. in der Zentralstation müssen Datenpunkte vom gleichen Typ, gleicher gemeinsamen ASDU-Adresse und mit der gleichen Informationsobjekt-Adresse wie in der Unterstation konfiguriert werden. Andere Parameter wie z.B. das Mapping-Bereich, Byte-, Bit-Offset können beliebig konfiguriert werden.

Es stehen folgende Funktionen zur Manipulation der Applikationsdatenbank zur Verfügung:

Function	Description
F_iecCreateTableHnd [▶ 293]	Initialisiert das Hash-Tabellenhandle
F_iecAddTableEntry [▶ 294]	Konfiguriert und fügt einen einen neuen Hash-Tabelleneintrag
F_iecRemoveTableEntry [▶ 299]	Entfernt einen Hash-Tabelleneintrag
F_iecLookupTableEntry [▶ 298]	Prüft ob ein bestimmter Hash-Tabelleneintrag existiert
F_iecGetPosOfTableEntry [▶ 296]	Ermittelt die lineare Position eines Hash-Tabelleneientrags

Das Datenbank-Handle muss an die Funktion per VAR_IN_OUT übergeben werden. Im Regelfall wird die Konfiguration beim SPS-Programmstart einmalig in einer Init-Routine durchgeführt.

Um die Applikationsobjekte beim Programmstart zu konfigurieren wird in MAIN folgender SPS-Code hinzugefügt:

```
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 der SPS- und IEC-Prozessdaten

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Die TwinCAT SPS-Prozessdaten werden zur Programmlaufzeit zyklisch in die IEC-Prozessdaten (Applikationsobjekte) und umgekehrt gemappt (kopiert). Für das Mapping der IEC<->SPS Prozessdaten können bis zu 4 Prozessdatenbereiche (IO-Eingänge, IO-Ausgänge, Merkerbereich, Datenbereich) als Puffervariablen im SPS-Programm deklariert werden. Die Bytegröße der Puffer ist frei wählbar und kann für jeden Bereich unterschiedlich gewählt werden. Unbenutzte Bereiche müssen nicht unbedingt deklariert werden.

In unserem Einführungsbeispiel deklarieren wir 4 SPS-Prozessdatenbereiche mit jeweils 3000 Bytes:

```
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
```

Die Zuordnung, wie die Prozessdaten zur Laufzeit gemappt werden sollen, wird während der Konfiguration der Applikationsobjekte mit der F_iecAddTableEntry [▶ 294]-Funktion festgelegt.

Siehe auch in: Applikationsobjekte definieren und konfigurieren [▶ 484].

Die Puffervariablen wurden nun als Byte-Arrays deklariert. Um auf die gewünschten Daten besser zugreifen zu können definieren wir die einzelnen Variablen ein zweites Mal und legen diese auf die entsprechenden Byte/Bit-Offsetadressen. Bei einer Änderung im Byte-Array wird die entsprechende einzelne Variable gleichzeitig geändert und umgekehrt. Dies ist aber nicht zwingend notwendig. Sie können direkt auf die Bytes/Bits der Byte-Array-Puffervariablen zugreifen.

```
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 der IEC<->SPS Prozessdaten in der Zentralstation

Prozessdaten in Überwachungsrichtung (Slave->Master information)

Beispiel 1

Single point information (M_SP_NA_1) mit der IOA = 100, SPS Merkerbereich, Byteoffset = 100, Bitoffset = 0.

Unterstation -> ... -> Zentralstation FB -> memory[100].0 == msgSingle_0

Beispiel 2

Measured value, short floating point value (M_ME_NC_1) mit der IOA = 700, SPS Merkerbereich, Byteoffset = 700, Bitoffset = 0 (bedeutungslos).

Unterstation -> ... -> Zentralstation FB -> memory[700..703] == msgFloating_0

Prozessdaten in Steuerungsrichtung (Master->Slave commands)

Beispiel 1

Single command state (C_SC_NA_1) mit der IOA = 10, SPS Merkerbereich, Byteoffset = 2100, Bitoffset = 0.

cmdSingle_0 == memory[2100].0 -> Zentralstation FB -> ... -> Unterstation

Beispiel 2

Set point, short floating point value (C_SE_NC_1) mit der IOA = 70, SPS Merkerbereich, Byteoffset = 2700, Bitoffset = 0 (bedeutungslos).

cmdFloating_0 == memory[2700..2703] -> Zentralstation FB -> ... -> Unterstation

13.1.4 Instanz der IEC60870-5-104 Zentralstation deklarieren und aufrufen

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Die gesamte Funktionalität einer Zentralstation ist im Funktionsbaustein FB_IEC870_5_104Master gekapselt. Mit einer Instanz kann eine Verbindung zur Unterstation aufgebaut werden. Für eine weitere Verbindung deklarieren Sie eine weitere Instanz des Funktionsbausteins.

Fügen Sie im Deklarationsteil von MAIN folgenden SPS-Code ein:

```
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;

END_VAR
```

und im Programmteil wird die Instanz aufgerufen:

```
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-Protokollparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Über die IEC60870-5-104-Protokollparameter kann das Verhalten der Zentralstation an die Anforderungen des Unterstation angepasst werden. Die meisten Parameter sind mit Defaultwerten vorbelegt, so dass diese nicht verändert werden müssen.

Im unserem Beispiel verändern wir die Werte der iK- und iW-Parameter und configurieren die IP-Adresse und Portnummer der Unterstation zu der die Verbindung aufgebaut werden soll.

```
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
```

Die Dokumentation aller Übertragungsprotokoll-Parameter finden Sie hier: ST_IEC870_5_104PotocolParams [▶ 446].

13.1.6 Systemparameter

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Über die Systemparameter wird z.B. die gemeinsame ASDU-Adresse und die Anwenderfunktionen der Zentralstation konfiguriert.

In unserer Einführung konfigurieren wir folgende Systemparameter:

- Die gemeinsame ASDU-Adresse wird auf 7 gesetzt. (*asduAddr*)
- Das loggen der Debugmeldungen im Application-Log wird aktiviert (*dbgMode*). Es werden Änderungen im Gerätestatus gelogt.

Fügen Sie folgenden SPS-Code in Ihr SPS-Projekt ein:

```
IF init THEN
    init := FALSE;
...

     client.sysPara.asduAddr := 7;
    client.sysPara.dbgMode := IEC870_DEBUGMODE_DEVSTATE(* OR IEC870_DEBUGMODE_LINKERROR OR IEC870_DE
BUGMODE_ASDU OR IEC870_DEBUGMODE_LINKLAYER *);

...
ELSE
    client( pInputs := ADR( inputs ),
        cbInputs := SIZEOF( inputs ),
        pOutputs := ADR( outputs ),
...

END_IF
```

Die Dokumentation aller Systemparameter finden Sie hier: ST_IEC870_5_101SystemParams [▶ 315].

13.1.7 Initialisierungssequence

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/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 Stationsabfrage

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

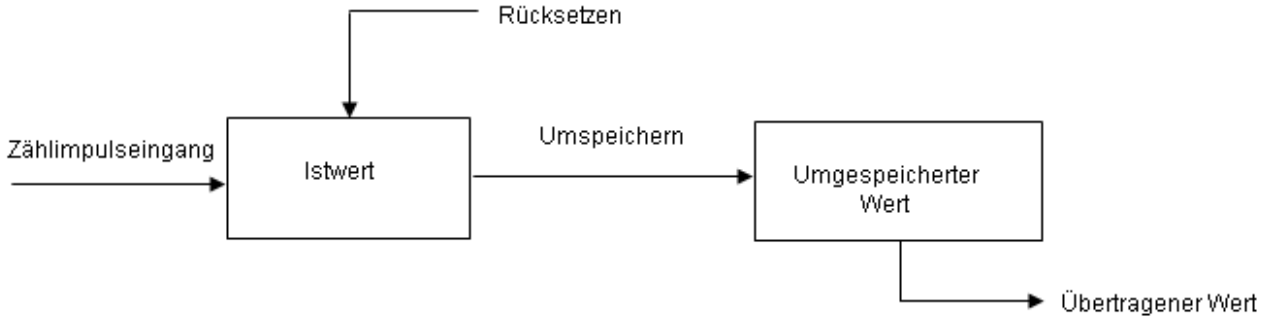
Der Stationsabfragebefehl wird von der Zentralstation eingeleitet. Im Kennungsfeld des Befehls ist auch die Gruppe (1 bis 16 oder allgemein) festgelegt. Die Unterstation überträgt die zu dieser Gruppe dazugehörigen Applikationsobjekte mit der Übertragungsursache <*20*> bis <*36*> an die Zentralstation. Applikationsobjekte mit Zeitmarken werden ohne Zeitmarken übertragen.

```
client.acqPara.arrGenro[0].tPollCycle := T#60s;
client.acqPara.arrGenro[0].eQOI := eIEC870_QOI_INROGEN;
client.acqPara.arrGenro[0].bEnable := TRUE;
```

13.1.9 Zählwertübertragung (counter interrogation)

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Allgemeines Modell für die Zählwertübertragung:



Die Istwerte werden durch Zähler aufsummiert. Die Istwerte können durch einen Umspeicherbefehl, der entweder von der Zentralstation empfangen oder örtlich (lokal in der Unterstation) erzeugt wird, periodisch in umgespeicherte Werte umgespeichert (kopiert) werden. Nach dem Umspeichern wird der erfasste Wert entweder auf Null zurückgesetzt (Erfassen von Inkrementalwerten) oder der Zähler fährt mit seinem Betrieb fort (Erfassen von Zählerständen).

Applikationsobjekte mit Zählwerten werden Gruppen zugeordnet. Die Gruppen werden einzeln umgespeichert (frozen), zurückgesetzt (reset) oder übertragen. Die Zentralstation sendet Zählwertabfragebefehle an die Unterstation. In einem Kennungsfeld des Befehls (QCC) wird die durchzuführende Aktion (FRZ) und Gruppe (RQT) festgelegt.

Die Zuordnung der Applikationsobjekte zu den einzelnen Gruppen (1 bis 4 oder allgemein) wird während der Konfiguration durch den Group-Flagparameter festgelegt. Es gibt vier Betriebsarten für die Erfassung von Zählerständen und Inkrementalwerten. Zu jeder Betriebsart sind einige Hinweise zur Konfiguration der Systemparameter oder der Applikationsobjekte aufgeführt.

Betriebsart A: Örtlich Umspeichern mit Spontanübertragung

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden spontan übertragen, nachdem die Funktion Umspeichern oder Umspeichern mit Rücksetzen durchgeführt wurde. Die Zentralstation gibt in dieser Betriebsart keine Zählwertabfragebefehle aus.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

Betriebsart B: Örtliches Umspeichern mit Zählerabfrage

Die Unterstation initiiert intern das Umspeichern oder Umspeichern mit Rücksetzen. Die umgespeicherten Zählwerte werden durch Zählwertabfragebefehle von der Zentralstation abgefragt. In diesem Fall darf die Zentralstation im Befehlskennungsfeld das Umspeichern oder Umspeichern mit Rücksetzen nicht benutzen (FRZ=0). Die Zählwerte werden allgemein oder in Gruppen (groups) 1 bis 4 abgefragt.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

Betriebsart C: Zentralstation leitet das Umspeichern, Umspeichern mit Rücksetzen oder Rücksetzen ein

Ein Zählwertabfragebefehl wird periodisch von der Zentralstation an die Unterstation ausgegeben, um das Umspeichern oder (und) Rücksetzen zu steuern. Dieser Befehl hat aber noch keine Übertragung der Zählwerte zur Folge. Erst ein nachfolgender Zählwertabfragebefehl wird von der Zentralstation gesendet um die umgespeicherten Zählwerte einzusammeln. Ähnlich, wie bei der Betriebsart B.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

Betriebsart D: Zentralstation leitet das Umspeichern und (oder) Rücksetzen ein und die umgespeicherten Werte werden spontan übertragen

Diese Betriebsart ist eine Kombination des Zählwertbefehls von der Zentralstation wie für Betriebsart C mit einer spontanen Übertragung der Zählwerte wie bei der Betriebsart A.

Konfiguration der Systemparameter:

Konfiguration der Applikationsobjekte:

13.1.10 Uhrzeitsynchronisation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

In Vorbereitung....

13.1.11 Test der Kommunikation

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Durch das Setzen der *bExecuteCmd*-Variable auf TRUE wird eine einfache Simulation der Befehle in Steuerungsrichtung aktiviert und mit FALSE deaktiviert. Bei einer aktiven Verbindung wird in unsem Beispiel ein Einzeln-Befehl (C_SC_NA_1, IOA = 10) zyklisch alle 10 Sekunden zur Unterstation übertragen.

```
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 Übertragungs- und Kommunikationsfehler

Hier können Sie die kompletten SPS-Sourcen entpacken: https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip

Die Stationsfehlermeldungen werden in einem FIFO abgelegt. Es können bis zu 10 Fehlermeldungen zwischengespeichert werden. Bei fatalen Kommunikationsfehlern (z.B. Fehler der Verbindungsschicht, die Checksumme des Frames passt nicht) wird die Verbindung unterbrochen und muss neu aufgebaut werden.

Fehler in der Applikationsschicht (z.B. der ASDU-Sendepuffer ist wegen zu vieler Frames übergelaufen) werden nur geloggt und führen nicht zum Verbindungsabbruch. Es immer noch möglich auch bei diesen Fehlern die Verbindung aus der Applikation zu unterbrechen. Neben dem Fehler-Code wird auch die Fehlerquelle in der Fehlermeldung abgelegt. Dieses erleichtert die Lokalisierung des Fehlers.

Beispiel

Die anfallenden Fehlermeldungen einer 60870-5-104 Zentralstation können durch folgenden Aufruf ausgelesen werden:

```
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 SPS-API

Einfache Projekte mit kompletten Sourcen finden Sie hier: IEC60870-5-104 Zentralstation [▶ 500].

Kompatibilitätsliste finden Sie hier: Interoperability check list

Übersicht der Fehlercodes finden Sie hier: Fehlercodes

Eine ausführliche Anleitung zur Implementierung der Unterstation in der SPS finden Sie hier: TUTORIAL [▶ 484]

Kurzanleitung

Applikationsobjekt-Datenbank

Die Applikationsobjekt-Datenbank der Zentralstation muss mit der Funktion: F_iecCreateTableHnd [▶ 293] als Hash-Tabelle konfiguriert werden. Die einzelnen Arrayelemente werden dabei als Hash-Tabelle untereinander verlinkt. Dies ermöglicht u. a. einen schnelleren Zugriff auf die einzelnen Datenpunkte, bringt aber auch einige Nachteile mit sich, die beachtet werden müssen:

- Die Größe der Applikationsdatenbank (Arraygröße) darf nicht zur Laufzeit (z.B. durch Online-Change) verändert werden. Die Zentralstation stoppt sofort die Ausführung und meldet einen Fehler. Der Grund: Die Verlinkung der Hash-Tabelle passt nicht mehr. Bei Programmänderungen laden Sie am besten das komplette Projekt ins Laufzeitsystem.
- Auf die einzelnen Array-Elemente darf nicht per Index sondern nur mit Hilfe der speziellen Funktionen zugegriffen werden (z.B.: F_iecAddTableEntry [▶ 294] usw.).
- Bei einem indizierten Zugriff auf die Tabellenelemente dürfen die internen Konfigurationsparameter nicht beschrieben oder verändert werden. Bei einer Änderung des Typs, der ASDU-Adresse oder der Objektadresse kann der Datenpunkt nicht mehr gefunden werden. Ein Datenpunkt, der umkonfiguriert werden soll wird aus der Tabelle durch den Funktionsaufruf F_iecRemoveTableEntry [▶ 299] zuerst entfernt. Danach kann der neue Datenpunkt hinzugefügt werden.

Bei einer Implementierung als lineare Tabelle müsste die Zentralstation bei jeder empfangenen ASDU (Dateneinheit) das komplette Array nach dem passenden Element durchsuchen. Dies würde bei vielen Datenpunkten sehr lange Ausführungszeiten generieren.

Protokollparameter

Über die Protokollparameter wird das Verhalten der TCP/IP-Transportschicht konfiguriert. Die meisten Protokollparameter sind bereits mit Defaultwerten vorbelegt und müssen nicht explizit gesetzt werden. Die SPS-Applikation muss aber einmalig die IP-Adresse (*sRemoteHost*) und die Port-Adresse (*nRemotePort*) der Unterstation konfigurieren.

Systemparameter

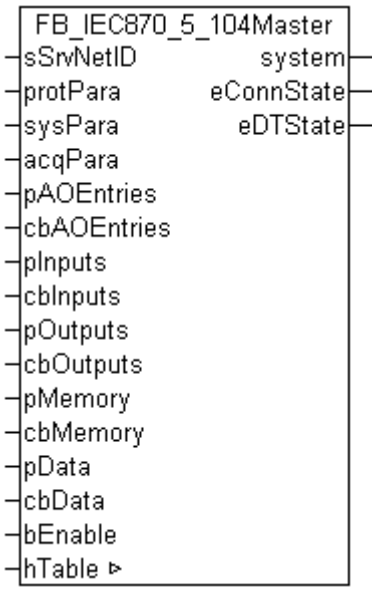
Die Systemparameter sind ebenfalls mit Defaultwerten vorbelegt. Während der Inbetriebnahme ist es nützlich die Debug-Ausgabe zu aktivieren (*dbgMode*) um mögliche Fehler lokalisieren zu können.

Parameter für die zyklische Datenerfassung

Folgende Parameter sind bereits mit Defaultwerten vorkonfiguriert:

- Initialisierungssequence (besteht aus einem Testbefehl, Uhrzeitsynchronisation, Stationsabfrage und Zählerabfrage);
- Zyklische Befehle:
  - Testbefehl alle 60s;
  - Uhrzeitsynchronisation alle 60s;
  - Stationsabfrage der Gruppe: Allgemein alle 60s;
  - Zählerabfrage der Gruppe: Allgemein alle 60s;

13.2.1 FB_IEC870_5_104Master



Mit einer Instanz des Funktionsbausteins FB_IEC870_5_104Master kann in der TwinCAT SPS eine IEC60870-5-104 Zentralstation (Master) implementiert werden. Pro Instanz des Funktionsbausteins wird eine Verbindung zum Slave aufgebaut.

Der Funktionsbaustein besitzt folgende Aktionen:

- **STARTDT** (startet den Datenaustausch);
- **STOPDT** (stoppt den Datenaustausch);

Im Normalfall wird der Datenaustausch automatisch gestartet, nachdem die Verbindung hergestellt wurde. Standardmäßig ist der Funktionsbaustein auch so konfiguriert. Bei Bedarf kann der Datenaustausch durch einen Aufruf der Aktionen gestoppt, bzw. gestartet werden.

VAR_IN_OUT

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

hTable: Applikationsobjekt-Datenbankhandle [▶ 342] (Hash-Tabellenhandle). Das Tabellenhandle muss vor der Benutzung einmalig mit der Funktion F_iecCreateTableHnd [▶ 293] initialisiert werden.

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 mit der Netzwerkadresse des TwinCAT TCP/IP Connection Servers. Für den lokalen Rechner (default) kann auch ein Leerstring angegeben werden.

protPara: IEC60870-5-104-Protokolparameter [▶ 446].

sysPara: Systemparameter [▶ 315].

acqPara: Parameter für die zyklische Datenerfassung.

pAOEntries: Adresse der Applikationsobjekt [▶ 312]-Datenbankvariablen.

cbAOEntries: Bytegröße der Applikationsobjekt-Datenbankvariablen.

pInputs: Adresse des SPS-Prozessdatenbereichs der Eingänge.

cbInputs: Bytegröße des SPS-Prozessdatenbereichs der Eingänge.

pOutputs: Adresse des SPS-Prozessdatenbereichs der Ausgänge.

cbOutputs: Bytegröße des SPS-Prozessdatenbereichs der Ausgänge.

pMamory: Adresse des SPS-Prozessdatenbereichs der Merker.

cbMamory: Bytegröße des SPS-Prozessdatenbereichs der Merker.

pData: Adresse des SPS-Datenbereichs.

cbData: Bytegröße des SPS-Datenbereichs.

bEnable : Aktiviert/Deaktiviert den Funktionsbaustein (Kommunikation und Verbindungen).

Die Adressen können mit dem ADR- und die Bytegrößen mit dem SIZEOF-Operator ermittelt werden.

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]. Diese Variable dient anderen IEC-Applikationsfunktionen als Kommunikationsschnittstelle zum IEC-Gerät (hier: Zentralstation).

- Membervariable *system.device* wird z.B. von der F_iecSetAOQuality [▶ 279]-Funktion als VAR_IN_OUT-Parameter erwartet.
- Membervariable *system.device.errors* ist ein Gerätefehler-Fifo. Die registrierten Fehler können von der SPS-Applikation ausgelesen und ausgewertet werden.

eConnState: Status der TCP/IP-Verbindung zum Slave.

eDTState: Status [▶ 448] des IEC60870-5-104-Datenaustauschs (STARTDT, STOPDT)

Beispiel in ST: IEC60870-5-104 Zentralstation [▶ 500]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder 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; werden automatisch eingebunden)

13.2.2 F_GetVersionTcIEC870_5_104Master

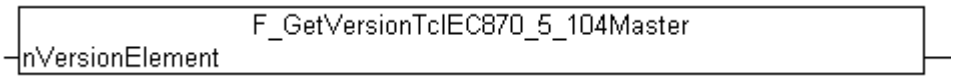


Abb. 9: F_GetVersionTcIEC870_5_104Master

Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC870_5_104Master: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder 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 werden automatisch eingebunden)

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: Kommunikationsschnittstelle [▶ 318] des IEC-Gerätes.

service: IEC-Gerätedienste;

hSOTable : Systemobjekt Datenbank Handle [▶ 342];

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1301	PC oder 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 werden automatisch eingebunden)

13.3 Fehlersuche/Diagnose

1. Überprüfen Sie die in dieser Dokumentation beschriebenen Hardware- und Softwareanforderungen (TwinCAT-Version, CE-Image-Version usw.).
2. Vergleichen/Überprüfen Sie die Kompatibilitätsliste der Unterstation mit der Kompatibilitätsliste der Leitstation.
3. Überprüfen Sie die Installationshinweise (z.B. Installation der CAB-Files auf einem CE-System).
4. Bei Verbindungsproblemen kann der PING-Befehl dazu benutzt werden, um festzustellen, ob der Kommunikationspartner über die Netzwerkverbindung erreichbar ist. Wenn dies nicht der Fall ist überprüfen Sie die Netzwerkkonfiguration und die Firewall-Einstellungen.
5. Überprüfen Sie ob die Netzwerkadresse, Portnummer die Sie an den Funktionsbaustein übergeben richtig sind.
6. Überprüfen Sie ob der Funktionsbaustein einen Fehlercode/Fehlerquelle [▶ 494] ausgibt. Die Dokumentation zu den Fehlercodes finden Sie hier: Übersicht der Fehlercodes.
7. Überprüfen Sie die am Funktionsbaustein eingestellten Protokolparameter [▶ 446] (iK, iW, t0, t1, t2, t3, APDULength, usw.). Vergleichen Sie die Protokolparameter mit den Parametern in der Unterstation.
8. Überprüfen Sie die am Funktionsbaustein eingestellten Systemparameter [▶ 315] (ASDU-Adresse, Länge der ASDU-Adresse, Länge der Informationsobjekt-Adresse, Länge der Übertragungsursache COT, usw.). Vergleichen Sie die Systemparameter mit den Parametern in der Unterstation.
9. Überprüfen Sie die am Funktionsbaustein eingestellten Parameter für die zyklische Datenerfassung [▶ 319] (Akquisition) (Initialisierungssequenz, zyklische Generalabfrage, zyklische Zählerabfrage, zyklische Testkommandos, usw.).
10. Überprüfen Sie die Konfiguration der Datenpunkte (Typ, Adresse des Informationsobjektes usw.).
11. Überprüfen Sie ob die Unterstation einen Fehlercode ausgibt.
12. Aktivieren Sie die Debugausgaben beim Aufbauen und Abbauen der Verbindung und/oder der ASDU Daten. Öffnen Sie den TwinCAT System Manager und aktivieren das LogView-Fenster. Prüfen Sie die Debugausgaben.
13. Überprüfen Sie die Verwendung des FB_SocketCloseAll()-Funktionsbausteins und des LISTEN_MODE_CLOSEALL-Parameters wenn Sie über mehrere TCP/IP Verbindungen (Server/ Clients) in einem Laufzeitsystem kommunizieren. Bei mehreren Verbindungen aktivieren Sie nur eine Instanz des FB_SocketCloseAll()-Funktionsbausteins einmalig im Initialisierungsschritt beim Programmstart. Der LISTEN_MODE_CLOSEALL-Parameter darf in diesem Fall nicht mehr verwendet werden.

14. Eine komplette Aufzeichnung der Netzwerkkommunikation kann mit Sniffer-Tools wie Wireshark durchgeführt werden. Die Aufnahme kann dann vom Beckhoff-Supportpersonal analysiert werden.

13.4 Beispiele

Im Beispielprojekten sind folgende Stationsparameter für die Zentralstation eingestellt:

- 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**

Voraussetzungen

SPS Project	Beschreibung
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751839627.zip	Einfache TwinCAT IEC60870-5-104 Zentralstation. Ein **minimalistisches** SPS-Projekt mit einem Single-Point in Überwachungsrichtung (IOA = 100) und einem Single-Command in Steuerungsrichtung (IOA = 10).
https://infosys.beckhoff.com/content/1031/TS650x_tcplc_iec60870-5-10x/Resources/11751838219.zip	TwinCAT IEC60870-5-104 Zentralstation

14 Fehlercodes

Übersicht

Fehlercodes (Hex)	Fehlercodes (Dez)	Fehlerquelle	Beschreibung
0x00000000-0x00002000	0-8192	Serial communication errors	Serielle Kommunikation Fehler (aber nur wenn die Fehlerquelle [▶ 331]=eIEC870_ESRC_IEC60870_5_101LINK)
0x00000000-0x00007800	0-30720	TwinCAT System Fehler	TwinCAT System Fehler (ADS-Fehlercodes inklusive wenn Fehlerquelle <> eIEC870_ESRC_IEC60870_5_101LINK
0x80070000-0x8007FFFF	2147942400-2148007935	Fehlerquelle = Fehlercode - 0x80070000 = Win32 system error code	0x80070000-0x8007FFFF
0x00008100-0x000081FF	32768-33023	IEC 60870-5-10x Fehler	Interne IEC 60870-5-10x Fehler

IEC 60870-5-10x-Fehler

Fehlercode (Hex)	Fehlercode (Dez)	Symbolische Konstante	Beschreibung
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

Fehlercode (Hex)	Fehlercode (Dez)	Symbolische Konstante	Beschreibung
0x00008112	33042	IEC870_COMMERR_STARTCHAR2	Invalid second stat character
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 timeout 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

Fehlercode (Hex)	Fehlercode (Dez)	Symbolische Konstante	Beschreibung
0x00008134	33076	IEC870_COMMERR_SCANTIMEOUT	Station scan cycle timeout error

15 Glossar

Begriff	Beschreibung
Unterstation, Slave, Server	Synonyme für eine untergeordnete Station (sie wird überwacht)
Zentralstation, Leitstation, Master, Client	Synonyme für eine übergeordnete Station (Leitstation, sie überwacht andere Stationen)
Steuerungsrichtung	Datenübertragungsrichtung von der Zentralstation zur Unterstation
Überwachungsrichtung	Datenübertragungsrichtung von der Unterstation zur Zentralstation
Applikationsobjekte	IEC-Informationsobjekte in der TwinCAT SPS-Applikation (Single Points, Double Points, Measured Values, Short Floating Point Values usw.)
APDU	Protokolldateneinheit der Anwendungsschicht (application protocol data unit)
APCI	Protokolsteuerinformation der Anwendungsschicht (application protocol control information)
ASDU	Dienstdateneinheit der Anwendungsschicht (application service data unit)
IOA, Addresse des Informationsobjekts	Adresse des Single Points, Double Points usw. (information object address)
Primärstation	Die Primärstation sendet Befehle (Anfragen) an die Sekundärstation und steuert/kontrolliert so die Datenübertragung der Sekundärstation.
Sekundärstation	Die Sekundärstation antwortet auf die Anfragen der Primärstation.
Kombinierte Station	Kombinierte Stationen können die Rolle der Primärstation und Sekundärstation annehmen (symetrische Übertragung, balanced mode).
Symetrische Übertragung (balanced mode)	Beide Stationen können als Primärstation oder Sekundärstation agieren und die Datenübertragung initialisieren.
Unsymetrische Übertragung (unbalanced mode)	Die Datenübetragung wird immer durch die Primärstation initiert und gesteuert. Die Zentralstation agiert immer als Primärstation und die Unterstation als Sekundärstation.

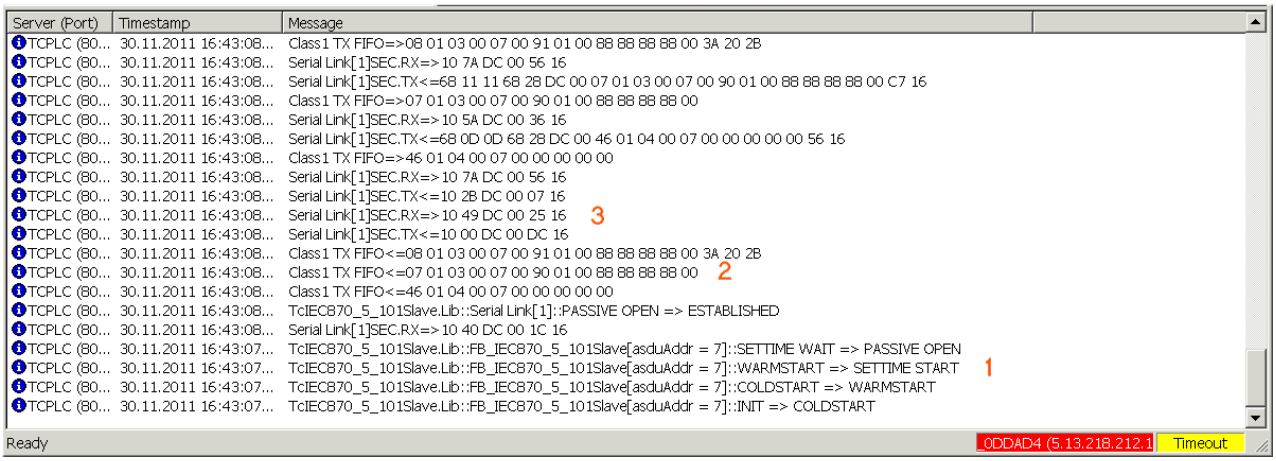
16 Anhang

16.1 Fehlersuche und Diagnose mit serieller Verbindung

Debug-Meldungen, die ins Application-Log geschrieben werden erleichtern eine Fehlersuche im System. Zur Zeit können drei Stufen der Debugmeldungen in einer IEC-Applikation aktiviert werden. Diese Meldungen können durch den dbgMode-Systemparameter der Leitstation aktiviert werden (ST_IEC870_5_101SystemParams [▶ 315]).

1. Stationsstatus-Meldungen (dbgMode: IEC870_DEBUGMODE_DEVSTATE);
2. Hexadezimale Ausgabe der ASDU's (ohne Link-Layer-Control-Header, dbgMode: IEC870_DEBUGMODE_ASDU). Pro Zeile werden 32-ASDU-Datenbytes als hexadezimale Zahlen ausgegeben. Längere ASDU's werden auf mehrere Zeilen verteilt;
3. Hexadezimale Ausgabe der APDU's (Serial-Port-Telegramme, dbgMode: IEC870_DEBUGMODE_LINKLAYER). Pro Zeile werden 32-APDU-Datenbytes als hexadezimale Zahlen ausgegeben. Ähnlich wie bei 2. werden längere APDU's auf mehrere Zeilen verteilt;

Optional können auch Link-Layer-Fehler ausgegeben werden (dbgMode: IEC870_DEBUGMODE_LINKERROR). Um die aktivierten Debugmeldungen zu sehen starten Sie TwinCAT System Manager und aktivieren Sie die Loggeransicht. Die nachfolgende Grafik zeigt eine Debugausgabe. Die drei unterschiedlichen Typen der Meldungen wurden mit entsprechenden Zahlen markiert.



Weitere Diagnosetools:

- Portmon for Windows (v3.02, Windows Sysinternals);
- Diverse Protokoll-Test-Suit-Produkte;

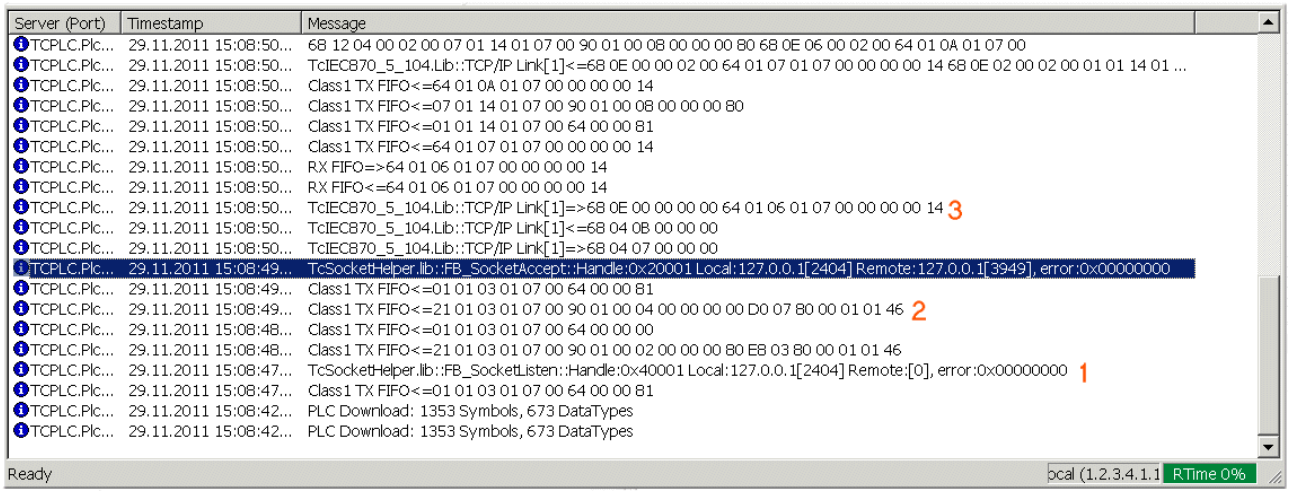
16.2 Fehlersuche und Diagnose mit TCP/IP-Verbindung

Debugmeldungen, die ins Application-Log geschrieben werden erleichtern eine Fehlersuche im System. Zur Zeit können drei Stufen der Debugmeldungen bei der Benutzung der "Low Level"-Schnittstelle aktiviert werden:

1. Debugmeldungen, die beim Aufbauen oder Abbauen der TCP/IP-Verbindung geloggt werden (Meldungen der TcSocketHelper.Lib). Diese Meldungen können durch den nMode-Parameter: CONNECT_MODE_ENABLEDBG beim Aufruf der F_CreateServerHnd-Funktion (Unterstation) bzw. beim Aufruf des FB_ClientServerConnection (Leitstation) aktiviert werden;
2. Hexadezimale Ausgabe der ASDU's (ohne Link-Layer-Control-Header). Pro Zeile werden 32-ASDU-Datenbytes als hexadezimale Zahlen ausgegeben. Längere ASDU's werden auf mehrere Zeilen verteilt. Die ST_IEC870_5_101TBuffer -Membervariable eDbg aktiviert die Debugausgabe;

3. Hexadezimale Ausgabe der APDU's (TCP/IP Telegramme). Pro Zeile werden 32-APDU-Datenbytes als hexadezimale Zahlen ausgegeben. Ähnlich wie bei 2. werden längere APDU's auf mehrere Zeilen verteilt. Die FB_IEC870_5_104TProtocol [▶ 443]-Eingangsvariable bOutDbg aktiviert die Debugausgabe;

Um die aktivierten Debugmeldungen zu sehen starten Sie TwinCAT System Manager und aktivieren Sie die Loggeransicht. Die nachfolgende Grafik zeigt eine Debugausgabe. Die drei unterschiedlichen Typen der Meldungen wurden mit entsprechenden Zahlen markiert.



Weitere Diagnosetools:

- TwinCAT ADS Monitor;
- Netzwerkmonitor;
- Wireshark;
- Ethereal;
- Diverse Protokoll-Test-Suit-Produkte;

16.3 Konfiguration der seriellen Schnittstellen

- Konfiguration der standard PC COMx-Schnittstellen [▶ 506]
- Konfiguration der seriellen Busklemmen KL6xxx [▶ 508]
- Konfiguration der seriellen Busklemmen EL6xxx [▶ 508]

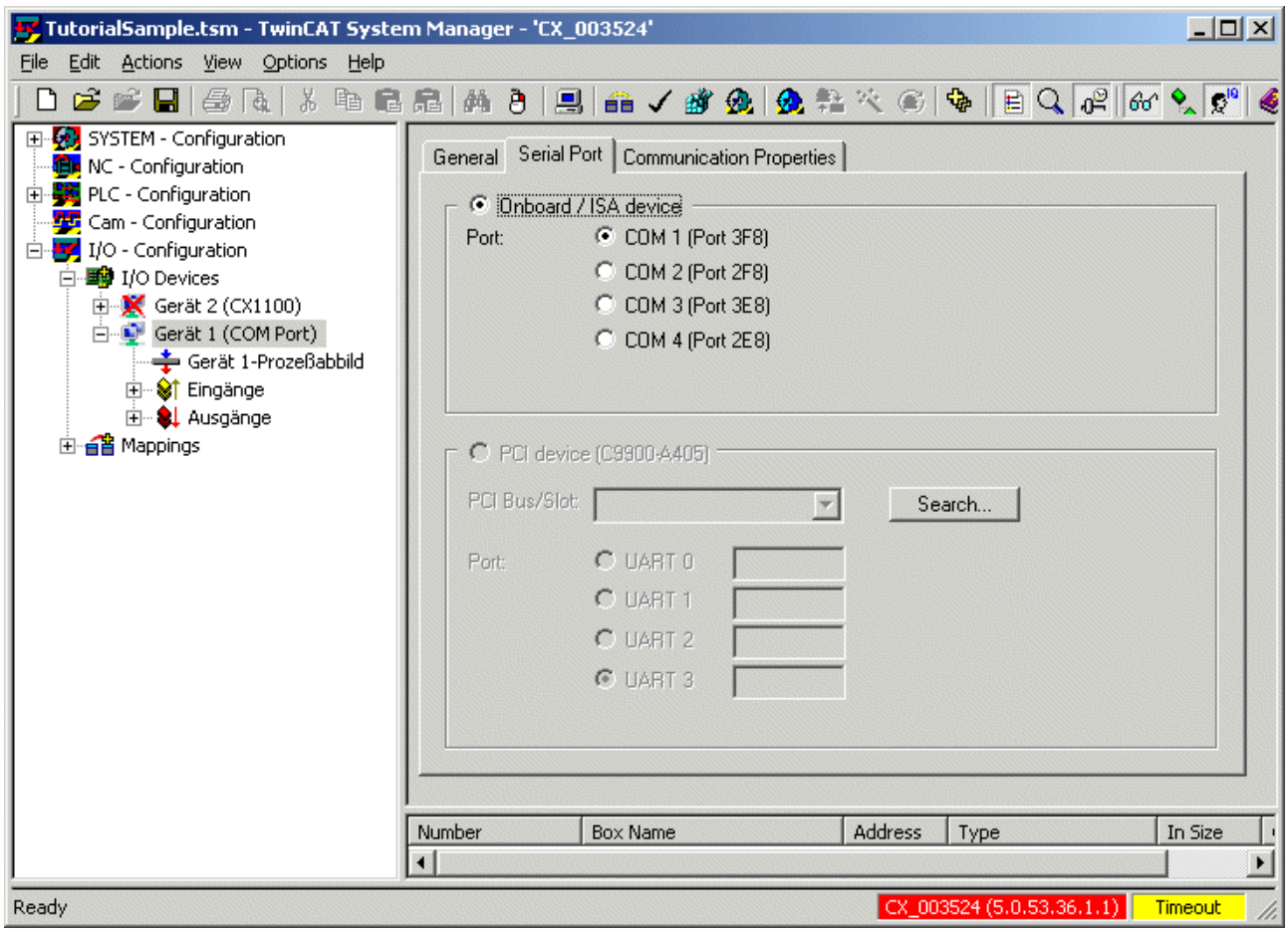
1. Konfiguration der standard PC COMx-Schnittstellen

Die serielle Schnittstelle kann nur in PC/CX-Systemen eingefügt werden.

Mit der rechten Maustaste "E/A Geräte" anklicken. "Gerät anfügen" auswählen. Unter "Verschiedenes" die "Serielle Schnittstelle" auswählen. Anschließend nachfolgende Einstellungen vornehmen.

1.1. Karteireiter Serial Port

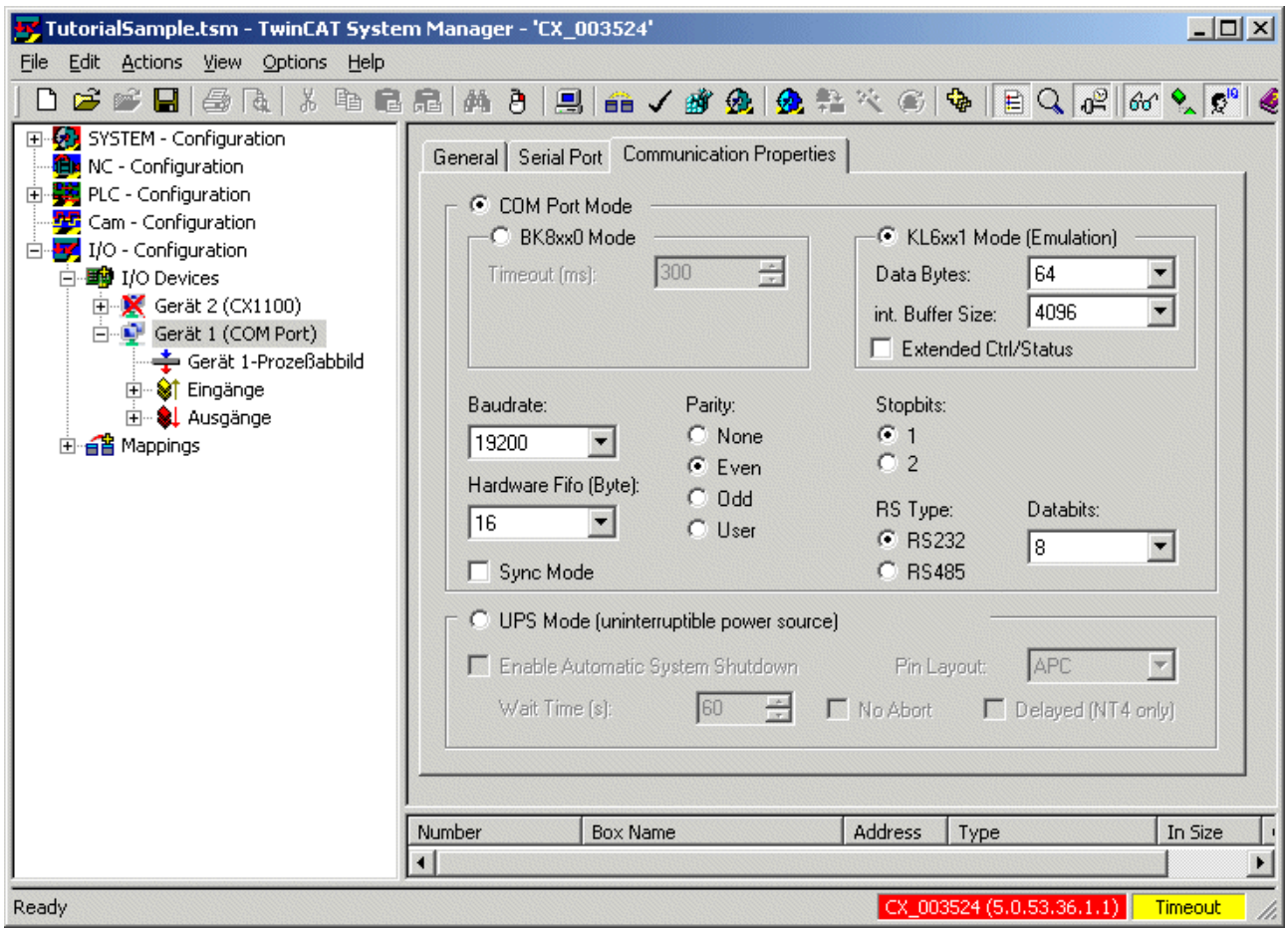
Serielle Schnittstelle COM1



1.2. Karteireiter Eigenschaften

Die Option KL6xxx1 Mode (Emulation) anwählen und dann die Kommunikationsparameter einstellen:

Baudrate, hier 19200 Baud, 8 Datenbits, Parity=even, 1 Stoppbit



2. Konfiguration der seriellen Busklemmen KL6xxx

Die Schnittstelle wird in der TwinCAT SPS durch die Instanz des FB_IEC870_SerialLineCtrl-Funktionsbausteins konfiguriert. Die Kommunikationsparameter wie Baudrate, Parity usw. sind an diesem Baustein einzustellen.

2.1 RS485 Betriebsmodus

Bei der RS485 Betriebsart (z.B. KL6041) werden die Daten in halbduplex Übertragung ausgetauscht. Im RS485 Betriebsmodus werden die Sende- und Empfangsleitungen miteinander verbunden. Dadurch empfängt die Klemme nicht nur die Daten anderer Teilnehmer, sondern auch ihre eigenen Sendedaten. Diese Daten können die Kommunikation stören. Für den Halbduplex-Modus muss der *Handshake*-Eingang des FB_IEC870_SerialLineCtrl-Funktionsbausteins auf den Wert: RS485_HALFDUPLEX gesetzt werden.

3. Konfiguration der seriellen Busklemmen EL6xxx

Die serielle Schnittstelle wird im TwinCAT System Manager Karteireiter CoE-Online (CoE=**C**anOpen **o**ver **E**therCAT) eingestellt. Der Karteireiter steht nur bei Online-Zugriff auf die Klemme zur Verfügung, d.h., ist der System Manger nicht mit der Hardware verbunden, können keine Einstellungen verändert werden. Die Einstellungen werden in der Klemme Spannungsausfallsicher abgespeichert. Nach einem Scannen der Klemmen müssen diese Daten nicht neu eingegeben werden. Bei einem Tausch der Klemme werden die Daten aus der neuen Klemme aktiv und müssen demzufolge wieder angepasst werden. Um das zu Verhindern, müssenim Karteireiter *Startup* die geänderten Daten eingetragen werden. Beim Systemstart werden die Daten aus dem Reiter *Startup* in die Klemme übertragen. Ein Tausch der Klemme ist somit Problemlos möglich. Nach einem Scannen der Klemmen müssen die Daten im Reiter *Startup* neu eingegeben werden.

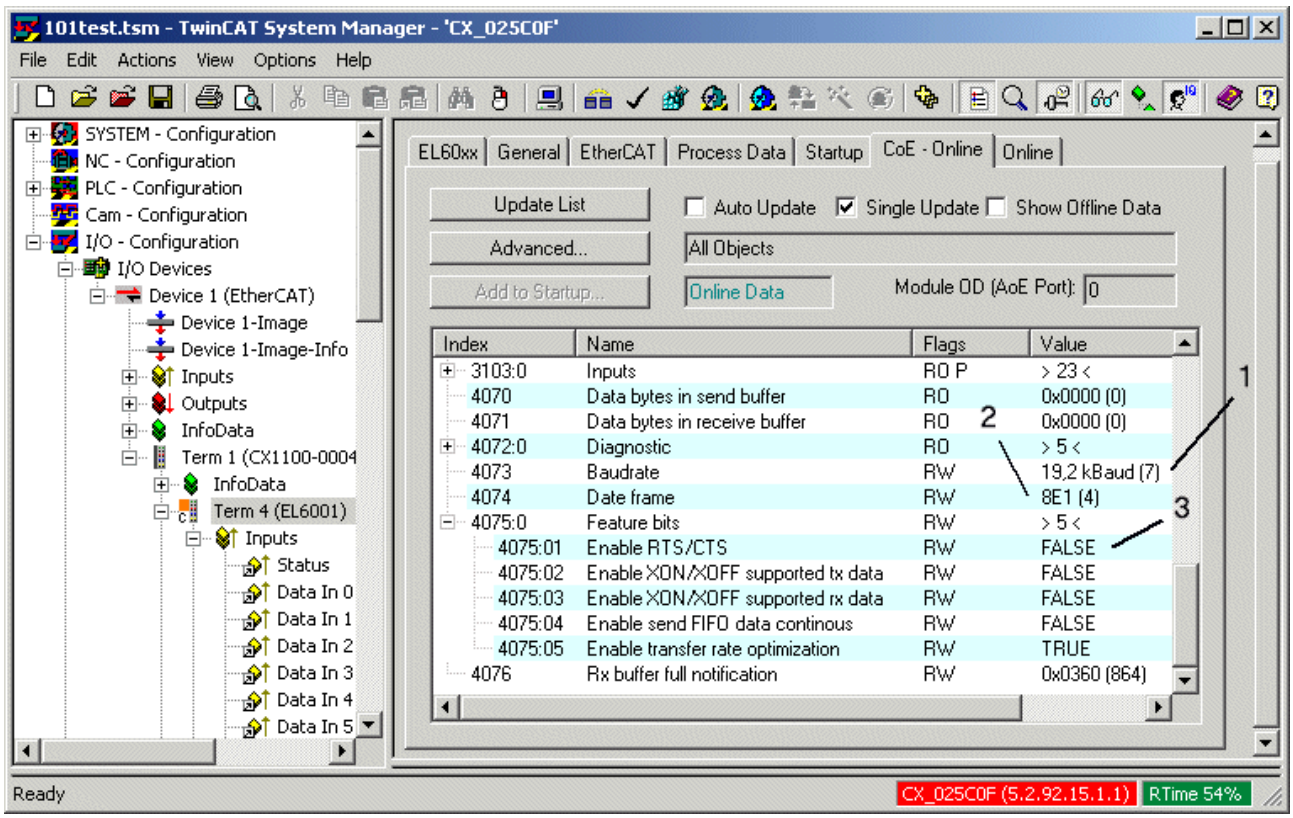
3.1. Karteireiter CoE

(1) Baudrate, hier 19200 Baud

(2) Date frame, 8E1 entspricht 8 Datenbits, Parity=even, 1 Stoppbit

(3) Feature bits -> Enable RTS/CTS = *FALSE* bei EL6001

Mit einem Doppelklick auf die entsprechende Zeile wird ein Menü geöffnet, in dem die Einstellungen geändert werden können.



3.1.1 RS485 Betriebsmodus

Bei der RS485 Betriebsart werden die Daten in halbduplex Übertragung ausgetauscht. Im RS485 Betriebsmodus werden die Sende- und Empfangsleitungen miteinander verbunden. Dadurch empfängt die Klemme nicht nur die Daten anderer Teilnehmer, sondern auch ihre eigenen Sendedaten. Diese Daten können die Kommunikation stören. Mit der Option "Enable half duplex" im "COM Settings"-Objekt kann dies unterdrückt werden (Index 8000:06).

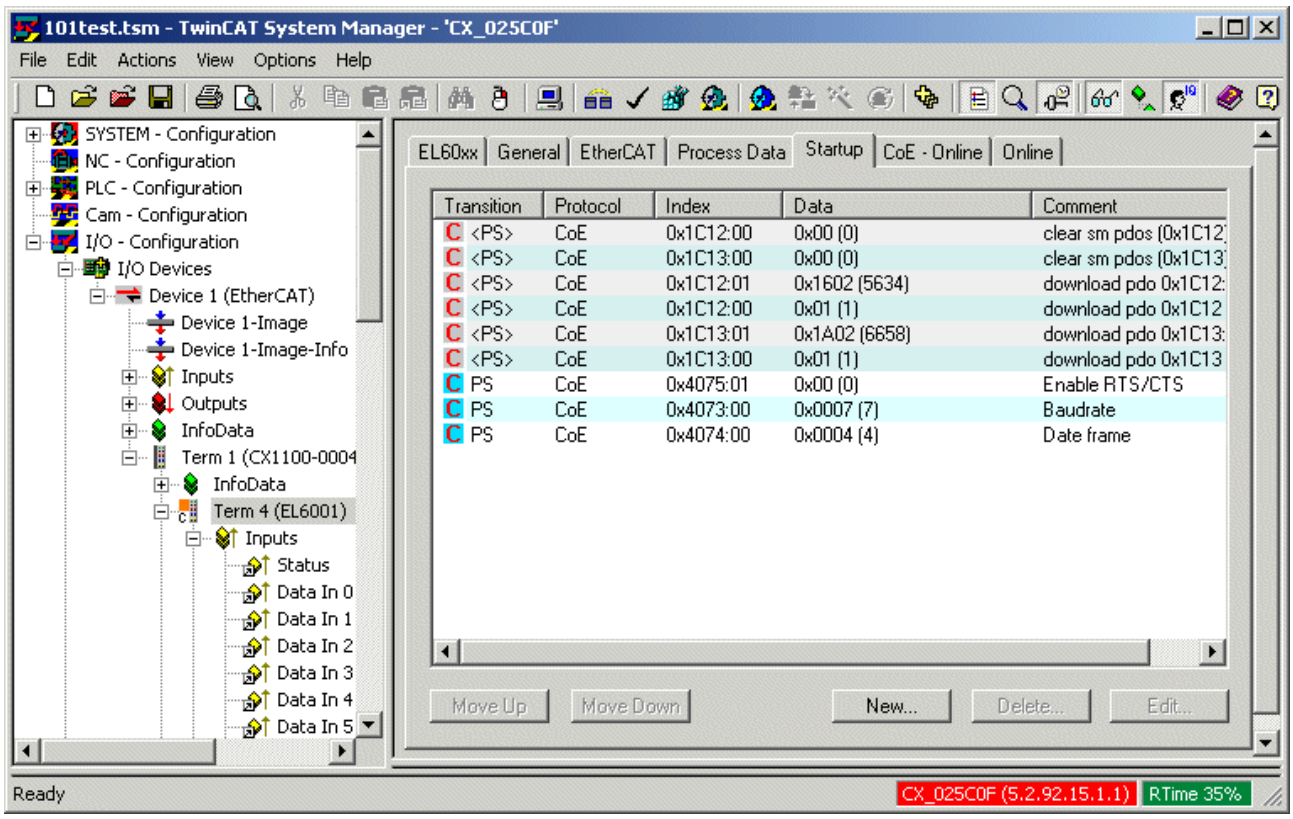
3.2. Karteireiter Startup

Alle vom Standard abweichende Einstellungen müssen in die Startup Liste eingetragen werden. Beim Systemstart werden die Daten aus dem Reiter *Startup* in die Klemme übertragen. Ein Tausch der Klemme ist somit Problemlos möglich. Nach einem Scannen der Klemmen müssen die Daten im Reiter *Startup* neu eingegeben werden.

Baudrate, hier 19200 Baud

Date frame, 8E1 entspricht 8 Datenbits, Parity=even, 1 Stoppbit

Feature bits -> Enable RTS/CTS = *FALSE* bei EL6001



16.4 Firewall Einstellungen

Die TwinCAT SPS Bibliothek: IEC 60870-5-104 Unterstation (slave/server) verwendet während der Kommunikation das TCP/IP als Transportprotokoll. Es ist daher darauf zu achten, daß der entsprechende TCP-Port bei Benutzung einer Firewall frei geschaltet wird. Die untere Tabelle listet standard Ports auf, die bei der Benutzung einer Firewall zu berücksichtigen sind.

Beschreibung	Typ	Protokoll	Port
IEC 60870-5-104 telecontrol protocol	Protocol	TCP	2404

Die Konfiguration der Windows Firewall wird über den entsprechenden Dialog in der Systemsteuerung vorgenommen. Weitere Informationen zur Konfiguration finden Sie in der Windows bzw. Firewall Dokumentation.

HINWEIS

Achten Sie bei einem Embedded Controller ohne Monitoranschluss und USB darauf, dass Sie den Port für Remote Display (Windows CE) oder Remote Desktop (Windows XP / Windows Vista) in der Firewall frei schalten. Ansonsten haben Sie keine Möglichkeit mehr, den Rechner über das Netzwerk zu administrieren.

Mehr Informationen:
www.beckhoff.com/ts650x

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com