Titel

Abkürzungsverzeichnis

PLC: Programmable Logic Controller
TCP: Transmission Control Protocol
IP: Internet Protocol
TCP/IP: Transmission Control Protocol/Internet Protocol (Internet Protocol Suite)
RTU: Remote Terminal Unit
ASCII: American Standard Code For Information Interchange
ADU: Application Data Unit
PDU: Protocol Data Unit
Dec.: Dezimal
Hex.: Hexadezimal
Bin.: Binär
RS-xxx: Recommended Standard xxx
TIA/EIA: Telecommunications Industry Association/Electronic Industrie Alliance
ISO: International Organization for Standardization/
IEC: International Electrotechnical Commission
PC: Personal Computer
V: Volt
Abb.: Abbildung
bps: Bit pro Sekunde
CRC: Cyclic Redundency Check
RJ45: Registered Jack 45 (Steckverbindung)
D-Sub (9): D-Subminiature mit 9 Kontakten (Steckverbindung)
OSI: Open Systems Interconnection Model
8-N-1: Startbit - 8x Datenbit – Stoppbit (Bitübertragungsformat)
7-E-1: Startbit - 7x Datenbit – Even Paritätsbit - Stoppbit (Bitübertragungsformat)
PS/2: Benannt nach dem 1987 IBM Personal System/2 Computer, Schnittstelle für Eingabegeräte, wie Maus, Tastatur oder Trackball
UART: Universal Asynchronous Receiver-Transmitter
FIFO: First In, First Out (Speicherprinzip)
SCADA: Supervisory Control And Data Acquisition (Computersystem in industriellen Steuerungsanlagen)
101er: IEC 60780-5-101 (Übertragungsprotokoll)
104er: IEC 60780-5-104 (Übertragungsprotokoll)
ACK: Acknowledgement
Ver.: Verbindung
P.: Primary (101er)
S.: Secondary (101er)

ASDU: Application Service Data Unit (101er)

APDU: Application Protocol Data Unit (104er)

SPS: Speicherprogrammierbare Steuerung

POU: Program Organization Units (IEC 61131)

WLAN: Wireless Local Area Network

UDP: User Datagram Protocol

SNMP: Simple Network Management Protocol

SSH: Secure Shell

Telnet: Teletype Network

FTP: File Transfer Protocol

DHCP: Dynamic Host Configuration Protocol

DNS: Domain Name System

DC: Direct Current (Gleichstrom)

FC: Function Code (Modbus)

FWG: Fernwirkgerät

USV: Unterbrechungsfreie Versorgung

OSI 7

Die von der ISO entworfene Open System Interconnection Architektur, OSI genannt, war eines der ersten Modelle um die Kommunikation zwischen zwei Computern dazustellen und wird auch heute sehr häufig benutzt, um Programme und Protokolle einzuordnen.

Abb. 1 OSI-Modell [1, S. 32]

Wie in Abb. 1 zu sehen ist, besteht das OSI-Modell aus 7 verschiedenen Schichten, auch Layer genannt. Von jeder dieser Schichten können sich Daten nur jeweils ein Schicht nach oben oder unten bewegen. Für die Kommunikation, also Daten zwischen A und B austauschen, innerhalb und zwischen Netzwerken werden nur die Schichten ein bis vier benötigt. Die unterste Schicht ist der Physical Layer. Dort sitzen Protokolle und Standards, die für die Übertragung von einzelnen Bits nötig sind. Also die elektrischen und physikalischen Spezifikationen des Senders, Empfänger und Übertragungsmedium. Sowie der entsprechende Maschinencode um diese zu betreiben. Eine Ebene höher ist die der Data Link Layer. In dieser Schicht werden die einzelnen Bits zu Paketen zusammengefasst und einfaches Routing innerhalb eines kleinen Netzwerks betrieben. Darüber liegt die Netzwerk Ebene, wo das Management eines großen Netzwerks im Fokus liegt und Daten nicht nur an in einem Gebäude, sondern weltweit über eine Vielzahl von verschiedenen Netzwerkgeräten übertragen werden können. Im Transport Layer wird die Kommunikation zwischen Computern organisiert, unabhängig davon, was für Geräte zwischen ihnen liegen [1, S. 31].

Serielle Kommunikation

Wenn zwei Parteien miteinander kommunizieren wollen, müssen sie sich vorher auf eine Kommunikationsart einigen, damit jede Nachricht sein Ziel erreicht und dort auch verstanden wird. Serielle Kommunikation ist eine sehr simple Kommunikationsart, da nur ein Leiter benötigt wird, welcher an und aus geschaltet wird. Damit Datenbits ausgelesen werden können, wird eine Clock oder eine andere Art an Zeitreferenz benötigt. Bei einer synchronen Verbindung gibt es eine zentrale Clock, die allen Geräten die Zeit vorgibt. Problem bei synchronen Verbindungen ist, dass alle kommunizierende Geräte mindestens einmal nach dem Einschalten das Signal der Clock empfangen haben müssen um sich zu Synchronisieren. Dies ist in vielen Anwendungsfälle nur schwer oder gar nicht möglich [2, S. 11–13].

Bei asynchronen Verbindungen wird keine Clock benötigt, da bei jeder Nachricht zwischen den Geräten diese selbst erkennen, wie die Nachricht auszulesen ist. Es gibt viele verschiedene Formate für das asynchrone Nachrichtenübertragen, ein Beliebtes aber ist 8-N-1. Dort folgen auf ein Startbit acht Datenbits und ein Stoppbit. Das N besagt, dass es kein Paritätsbit gibt. Parität sagt aus, ob es eine gerade oder ungerade Anzahl an 1 in den Datenbits gibt. Wenn bei einer empfangenen Nachricht das Paritätsbit nicht mit der Parität der Datenbits übereinstimmt, so wird die Nachricht verworfen, da es mindestens einen Fehler gibt. Ein Even Paritätsbit wird 1, wenn die Datenbits eine gerade Anzahl an 0 enthält und ein Odd Paritätsbit wird 1, wenn die Nachricht eine ungerade Anzahl an 0 enthält. Die meisten seriellen Schnittstellen unterstützen zwischen einem Start- und einem Stoppbit 5-8 Datenbits sowie ein Paritätsbit [2, S. 13–14].

Die Bitübertragungsrate einer Verbindung wird in Bits pro Sekunde angegeben. Die Datenänderungsübertragungsrate, auch Baud-Rate genannt, ist in vielen Fällen gleich der Bitübertragungsrate, da bei seriellen Verbindungen jedes Bit eine Datenänderung ist. Bei Telefon- oder Ethernetverbindungen jedoch ist die Baud-Rate um einiges niedriger als die Bitübertragungsrate, da in jede Datenänderung mehrere Bits kodiert sind. Das Hinzufügen eines Start- und Stoppbits bei 7-E-1 verringert die Übertragungsgeschwindigkeit um 30%, da auf sieben Datenbits zwei Formatbits und ein Paritätsbit kommen. Bei manchen Übertragungsprotokollen wird das Stoppbit doppelt so lang wie ein Datenbit gesetzt oder mehrere Paritätsbits verwendet um die Fehlerquote zu verringern. Ob die Verringerung der Datenübertragungsrate die erhöhte Fehlerreduktion wert ist, muss von Anwendungsfall zu Anwendungsfall entschieden werden [2, S. 13–15].

Serielle Anschlüsse waren für Jahrzehnte die bevorzugte Art um externe Geräte mit einem Computer zu verbinden. Aber bis auf ein oder zwei synchrone, serielle PS/2-Schnittstellen um Tastatur und Maus anzuschließen [3] werden keine serielle Anschlüsse mehr in modernen Computer verwendet. Serielle Schnittstellen sind und werden aber weiterhin in Kontroll- und Überwachungssystem eingesetzt, da sie günstig, wenig fehleranfällig und einfach zu bedienen sind [2, S. 26].

UART

Ein Universal Asynchronous Receiver-Transmitter, UART genannt, ist die Schnittstelle zwischen dem parallelen internen Bussystem eines (Mikro)Computers und einem seriellen Übertragungssystem. Dieser wandelt die zu sendende Daten des Computers

in ein serielles Format um und wandelt die empfangenen seriellen Daten in ein Format um, welches für den Computer lesbar ist. Außerdem sind viele UART für das Management der seriellen Verbindung zuständig, wie das Übertragungstiming verschiedener Geräte am selben Bus oder das automatische Anpassen an dieselbe Baud-Rate zwischen verschiedenen Geräten [2, S. 26].

In einem UART sind Buffer enthalten, die mehrere Byte zwischenspeichern können. So muss der Computer sich nicht um jeden einzelnen empfangen Bit kümmern, sondern kann die empfangene Nachricht in einem Zug auslesen. Dies ist auch von Vorteil, da das computer-interne Bussystem um ein Vielfaches schneller ist, als eine serielle Leitung. Jenes gilt ebenso beim Senden von seriellen Nachrichten. Der Computer kann die komplette zu sendende Nachricht an den UART senden, wo diese im Buffer gespeichert und daraufhin Stück für Stück gesendet wird. Diese Buffer sind first-In, first-out, kurz FIFO. Es wird also der zuerst empfangene Bit als erstes gesendet [2, S. 27]. Des Weiteren hat ein UART verschiedene Speicherregister. Unteranderem jeweils eines für die als letztes empfangenen und gesendeten Bits. Außerdem sind dort Konfigurationen, Statusinformationen, Bufferbenutzung und Fehlermeldungen gespeichert [2, S. 40–42].

RS-485

RS-485 ist ein Datenübertragungsstandard, wurde von der Telecommunications Industry Association und der Electronic Industrie Alliance entwickelt und ist auch unter dem Namen TIA/EIA-485 und ISO/IEC 8482.1993 bekannt [2, S. 185]. RS-485 wird in Industrie-, Medizin- und Konsum-Elektronik eingesetzt. Dieser Standard liegt auf OSI-Layer 1 und definiert nur die elektrische Hardware der Sender, Empfänger und Übertragungsmedien. Es ist möglich eine RS-485 Implementierung in halb oder voll Duplex zu betreiben, bei voll Duplex werden aber statt zwei vier Leiter benötigt. Außerdem sollte RS-485 in einer Bus-Topologie betrieben werden mit einem Haupt-Bus, von dem kurze Abzweigungen zu den einzelnen Sendern und Empfängern gehen [4, S. 1–2].

RS-485 benutzt ausgeglichene Übertragungsleitungen, wo jede Verbindung aus zwei Leitern (A und B) besteht und in jeder Leitung jeweils die diametrale Spannung zur anderen Leitung herrscht. Der Empfänger reagiert auf die Spannungsdifferenzen der beiden Leitungen und diese Übertragungsart wird deshalb auch Differentialsignalisierung genannt. Ein großer Vorteil von einem solchen Leiteraufbau ist die starke Reduzierung von Rauschen, da die Felder der einzelnen Leiter sich

gegenseitig auslöschen. Außerdem sind die Leiter so relativ immun gegen Unterschiede im Erdpotential bei längeren Übertragungsstrecken [2, S. 186–187].

Ein RS-485-Sender benötigt mindestens eine 1,5V Differenz zwischen seinen beiden Ausgängen. Die Differenz zur Erde ist nicht definiert, darf aber nicht mehr als 7 Volt von der Erde abweichen. Bei dem Empfänger muss die Differenz zwischen den zwei Leitern immer noch mindestens 0,2V betragen um korrekt erkannt zu werden. Wenn Leiter A mindestens 0,2V größer als Leiter B ist, wird eine 1 registriert und wenn Leiter A mindestens 0,2V kleiner als Leiter B ist, wird eine 0 registriert. Das heißt, dass auf dem Weg zwischen Sender und Empfänger mindestens 1,3V an Rauschen zu dem Signal hinzukommen können, es aber trotzdem erkannt wird [2, S. 190–191].

Wie in Abb. 2 zu sehen ist, müssen die Enden der beiden Haupt-Bus-Kabeln mit der, den Kabeln entsprechenden, charakteristischen Impedanz abgeschlossen werden, um Reflektionen vorzubeugen. Da der RS-485 Standard Kabel mit einer charakteristischen Impedanz von 120 Ohm vorschlägt, sind die meisten Abschlusswiderstände auch 120 Ohm stark. Wenn eine Verbindung in einem Umfeld mit starken elektrischen Interferenzen gelegt werden soll, werden die 120 Ohm Abschlusswiderstände meist mit zwei 60 Ohm Widerständen und einem Tiefpassfilter dazwischen ersetzt [4, S. 3].

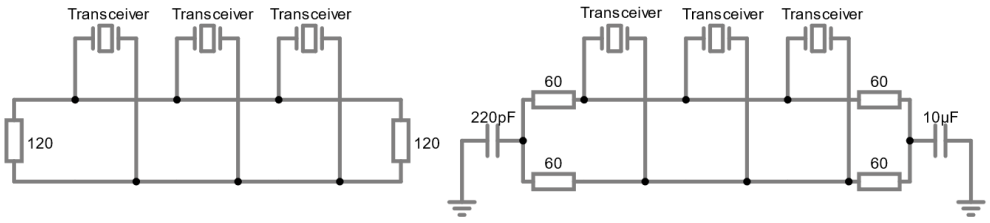


Abb. 2 RS-485 Abschlusswiderstände [4, S. 3]

Eine RS-485-Verbindung kann entweder bis zu 10 Mbps schnell oder bis zu 1200 Meter lang sein. Wie man in Abb. 3 sieht, steigt die Übertragungsrate ab 100 Kbps aufgrund der geringeren Kapazität des Kabels bei kürzerer Länge [2, S. 193].

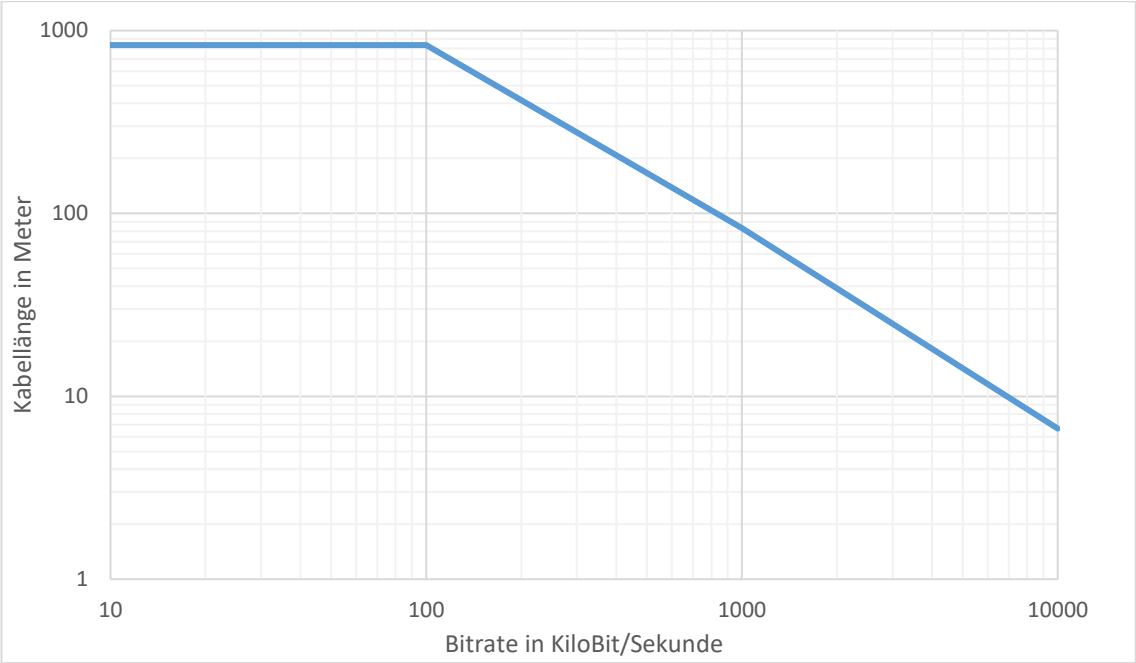


Abb. 3 RS-485 Geschwindigkeit in Abhängigkeit der Kabellänge [2, S. 193]

Modbus

Modbus ist ein Datenübertragungsprotokoll, welches von Modicon, jetzt Schneider Electric, im Jahr 1979 entwickelt wurde. Es wird von hunderten verschiedenen Herstellern in tausenden verschiedenen Geräten eingesetzt und ist damit der de facto Standard um PLCs, Computer, Sensoren und Aktoren miteinander zu verbinden [5, S. 508], [6]. Es gibt verschiedene Versionen von Modbus, unter anderem Modbus RTU, Modbus ASCII, Modbus über TCP/IP, Modbus Plus und Modbus Enron [5, S. 509], [6]–[8]. Serieller Modbus (Modbus RTU, Modbus ASCII,) beruht auf dem Master/Slave-System, wo eine einziges Master-System bis zu 247 Slave-Systeme steuern kann [5, S. 508]. Die folgenden Kapitel behandeln nur Modbus RTU.

Das Master-System kann zwei Arten von Nachrichten (Requests) an Slave-Systeme schicken: Unicast, wo das Master-System genau ein Slave-System anspricht, welches einen Prozess ausführt und ihm danach mit einer Reply antwortet, und Broadcast, wo das Master-System allen Slave-Systemen eine Request schickt, welche aber nur ein Schreibbefehl sein kann. Bei einem Broadcast wird keine Reply von den Slaves übermittelt.

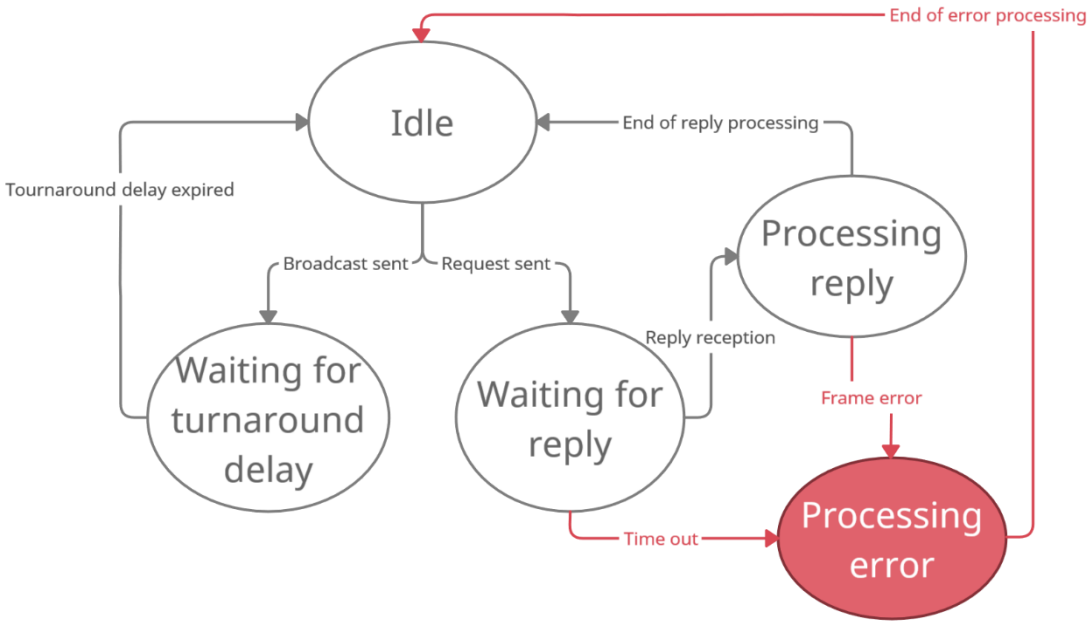


Abb. 4 Modbus Master Zustandsdiagramm [9, S. 9]

In Abb. 4 ist der vereinfachte Prozessablauf eines Modbus-Master-Systems dargestellt. Dieser empfängt von einem im OSI-Modell höher liegenden Protokoll Befehle. Das Master-System ist ohne Auftrag von außen im Ruhemodus (Idle). Wenn nun ein Protokoll einen Prozess in einem oder mehreren Slaves ausführen will, sendet es einen Befehl an das Master-System. Wenn es ein Broadcast-Befehl ist, sendet der Master diesen an alle Slaves und wartet dann eine vorher definierte Zeit (Turnaround delay) bevor es wieder in den Ruhemodus zurück geht. Bei einem Unicast-Befehl sendet der Master diesen an das ausgewählte Slave-System und wartet auf eine Reply. Wenn diese nicht nach einer im Voraus festgelegten Zeit zurückkommt oder diese empfangen wurde, aber Fehler beinhaltet, geht der Master zurück in den Ruhemodus. Wenn eine fehlerfreie Reply innerhalb des festgelegten Zeitfenster empfangen wird, wird diese an das auftraggebende Protokoll gegeben [9, S. 9–10].

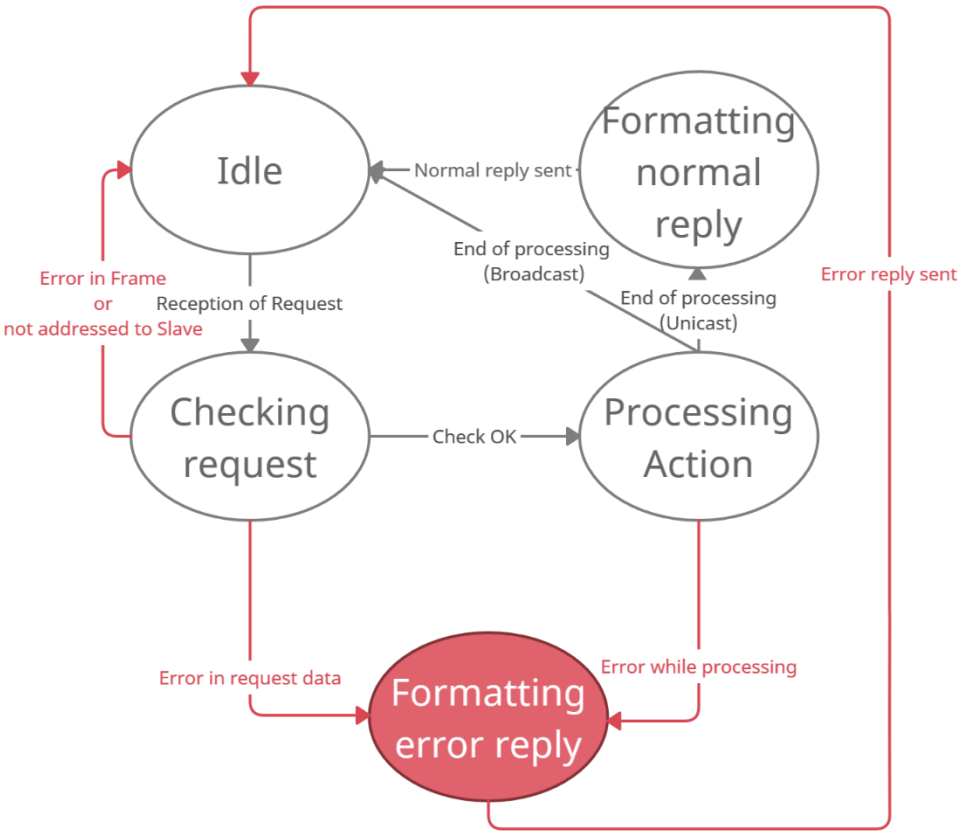


Abb. 5 Modbus Slave Zustandsdiagramm [9, S. 10]

In Abb. 5 ist der komplette Prozessablauf eines Modbus Slaves dargestellt. Beim Starten des Slaves-Systems befindet sich dieses im Ruhemodus (Idle). Wenn der Slave eine Nachricht empfängt, wird diese zuerst auf das korrekte Format überprüft. Außerdem stellt der Slave fest, ob die Nachricht an seine Adresse gerichtet ist. Wenn die Nachricht förmliche Fehler oder die falsche Adresse enthält, wird diese Nachricht ignoriert und der Slave geht zurück in den Ruhemodus. Ist die Nachricht an den Slave adressiert und förmlich korrekt, wird zuerst überprüft ob er den Befehl in dieser Nachricht ausführen kann. Danach wird der Befehl ausgeführt. Wenn bei diesen beiden Prozessen Fehler vorkommen, wird eine Fehlernachricht (error reply) an den Master geschickt, die einem dem Fehler entsprechenden Code enthält. Gibt es keine Fehler, geht der Slave bei einer Broadcast-Nachricht direkt nach der Ausführung, und bei eine Unicast-Nachricht nach dem Senden der Antwort (Reply), in den Ruhemodus [9, S. 10].

Jedes übertragene Daten-Byte in einer Modbus-Nachricht enthält acht Datenbits, beziehungsweise zwei hexadezimal Zeichen, und 3 Formatbits, das Start-, Stopp-, und Paritätsbit. Im Default-Paritätsmodus wird bei einer ungeraden Anzahl an Einsen das Paritätsbit auf eins gesetzt. In Abb. 6 ist zu sehen, dass zuerst das Startbit, dann die

Datenbits und danach das Paritäts- und das Stoppbit. Außerdem ist es möglich das Paritätsbit durch ein zweites Stoppbit zu ersetzen [9, S. 12].

Start	Data	Data	Data	Data	Data	Data	Data	Data	Par	Stop

Abb. 6 Bitsequenz Modbus mit Paritätsbit [9, S. 12]

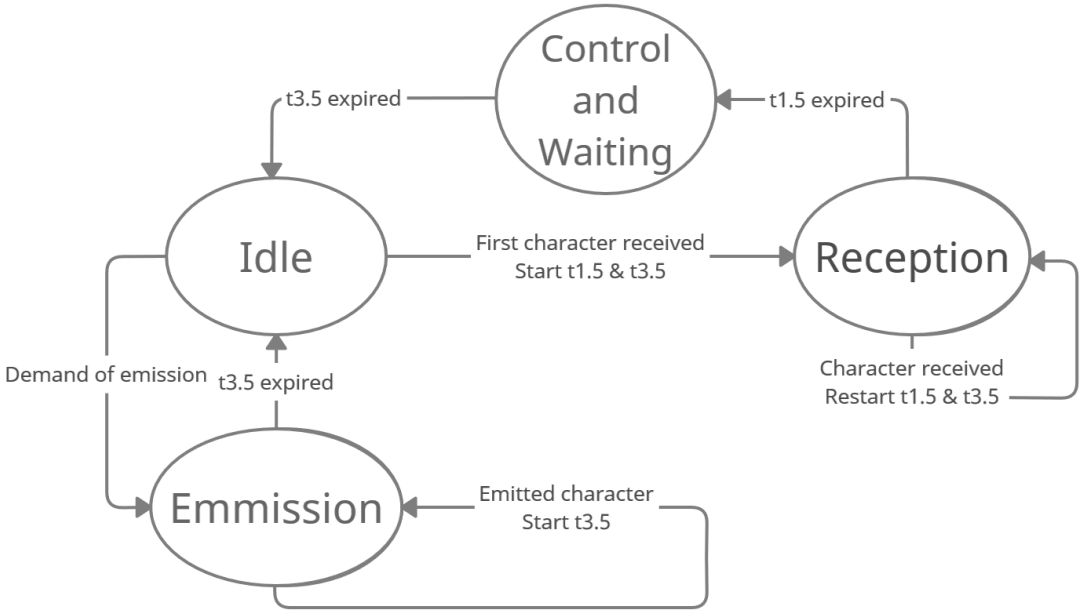


Abb. 7 Modbus Übertragung Zustandsdiagramm [9, S. 14]

In Abb. 7 sind die verschiedenen Zustände eines Modbus-Senders/Empfängers dargestellt. Diese sind bei Master und Slave gleich. Zwischen zwei Modbus-Nachrichten müssen mindestens 3,5 zeichenlang Stille sein und wenn zwischen 2 Zeichen mehr als 1,5 zeichenlang Stille ist, wird die Nachricht vom Empfänger als beendet betracht. Nach dem Empfangen eines Zeichens starten beide Timer (t1.5 & t3.5) und werden beim Empfangen des nächsten Zeichen wieder neugestartet. Nachdem der 1,5 Zeichen Timer überschritten wird, wird bei der empfangenen Nachricht die Controlframes (Parität, CRC und richtige Adresse) überprüft. [9, S. 13].

Eine vollständige Nachricht im Modbus-Protokoll ist in Abb. 8 zu sehen. Diese wird Application Data Unit (ADU) genannt und besteht aus einer Adresse, der Protocol Data Unit (PDU), bestehend aus dem Funktionscode und den zu übertragenden Daten, und schlussendlich dem Fehlercheck. Eine Nachricht kann maximal 256 Byte groß sein [9, S. 13].

ADU			
	PDU		
Slave Adresse	Funktionscode	Daten	CRC
1 Byte	1 Byte	0 bis 252 Byte	2 Byte

Abb. 8 Modbus Frame [9, S. 13]

Es gibt 247 valide Slave-Adressen und eine Broadcastadresse. Im Funktionscodefeld steht eine acht Bit lange Nachricht, die dem Slave sagt, welchen Befehl er auszuführen hat. Es gibt elf verschiedene Funktionscodes, welche alle Modbusübertragungsmedien unterstützen. Jene sind in TABELLE I dargestellt sind. Außerdem gibt es mehrere dutzende Funktionscode, welche Übertragungsmedium spezifisch sind. Desweiteren ist es möglich, dass Gerätehersteller eigene Modbusbefehle implementieren. Das Datenfeld ist n*8 Bit lang und enthält die zu übertragenden Daten. Als letztes kommt das 16-bit-lange Fehlercheckfeld, wo mit Hilfe von Cyclical-Redundancy-Check-Rechnungen die empfangene Nachricht überprüft werden kann [10, S. 18–25].

TABELLE I ALLGEMEINE MODBUS FUNKTIONEN [10, S. 31–66]

Code in Dec.	Name	Funktion
01	Read Coil Status	Liest den AN/AUS Status eines Coils
02	Read Input Status	Liest den AN/AUS Status eines Inputs
03	Read Holding Register	Liest die Werte von Holding Registern
04	Read Input Register	Liest die Werte von Input Registern
05	Force Single Coil	Setzt den AN/AUS Status eines Coils
06	Preset Single Register	Setzt die Werte von einem Holding Register
07	Read Exception Status	Liest den Status von prädefinierten Status Coils. Bspw.: Maschine in Betrieb
08	Diagnostic	Mehrere Unterfunktionen für verschieden Status von Slaves
15	Force Multiple Coils	Setzt den AN/AUS Status mehrere Coils
16	Preset Multiple Register	Setzt die Werte von mehreren Holding Register
17	Report Slave ID	Zeigt die Slave ID an

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

Abb. 9 Beispiel Modbus Funktion 03 [10, S. 15]

In Abb. 9 kann man eine Beispielsregisterabfrage sehen. Es wurde nur die PDU dargestellt, bei der gesamten Nachricht wurde am Anfang noch die Adresse und am

Ende die CRC-Summe übermittelt. Alle Registeradressen und Registerwerte werden in einen High- und Lowteil aufgeteilt, welche hintereinander gesetzt werden. Der Master fragt dort die Daten von Register 0x6B, 0x6C und 0x6D ab, indem das abzufragende Startregister und die Anzahl der darauffolgenden Register definiert werden. Der Slave antwortet daraufhin erst mit der Funktion und der Byteanzahl der Antwortwerte. Daraufhin folgen die Werte der abgefragten Register. Der Master kann nun die Antwort (Register 0x6B = 0x022B, Register 0x6C = 0x000 und Register 0x6D = 0x0064) an das höherrangige Protokoll senden.

Der Standard Modbus-Datenübertragungsstandard ist RS485 und alle Systeme sind parallel angeschlossen. Es werden mindestens zwei Datenübertragungskabel benötigt, idealerweise aber auch ein Erdkabel. Diese Kabel werden entweder mit Schraub-, RJ45- oder D-Sub(9)-Steckverbinder ausgestattet. Die Standardübertragungsgeschwindigkeit beträgt 19200 bps. Es können aber auch Andere benutzt werden, wie 1200 bps, 9600 bps oder 115000 bps [9, S. 20]. Bei einem RS485-Modbus-Netzwerk können maximal 32 Systeme ohne Repeater angeschlossen werden und werden alle parallel an einen zentralen Bus angeschlossen. Bei einem 9600 bps und 0,2 mm dicken Kabel liegt die maximale Länge des Buses bei 1000 Metern. Abzweigungen von diesem dürfen nicht länger als 20 Meter sein. Damit es zu keinen Reflexionen am Ende des Buses kommt, müssen dort die beiden Signalkabel mit ein 150 Ohm Abschlusswiderstand oder jeweils ein 1 nF Kondensator und ein 120 Ohm Abschlusswiderstand verbunden sein [9, S. 27–28].

IEC 60870-5

Der Standard IEC 60870-5 wurde ab dem Jahr 1990 Stück für Stück veröffentlicht. In den Sektionen IEC 60870-5-1 bis IEC 60870-5-5 wurden einzelne Teile des schlussendlichen Übertragungsprotokoll definiert. Dieses wurde 1995 unter dem Namen „IEC 60870-5-101 Companion Standard for Basic Telecontrol Tasks“ vorgestellt, kurz 101er. Es ist das erste komplette, einsatzbereite SCADA-Protokoll um über ein räumlich ausgedehntes Gebiet energietechnische Anlagen zu überwachen und zu steuern. In den folgenden IEC 60780-5-102 und IEC 60780-5-103 wurde Unterstützung für Netzschutztechnik, wie Distanz- und Differentialschutz, hinzugefügt. In der, im Jahr 2000 veröffentlichten, IEC 60780-5-104, kurz 104er, wurde dem 101er ein Netzwerk- und Transportlayer hinzugefügt, sodass nun auch 101er-Nachrichten über TCP/IP übertragen werden können [11, S. 177–178].

IEC 60870-5-101

101er arbeitet auf den OSI-Ebenen 1, 2 und 7, also dem Physical, dem Data Link und dem Application Layer. Um zwei Geräten, die mit 101er kommunizieren, physikalisch

zu verbinden, werden die bestehenden Standards RS-232 und RS-485 benutzt, sowie Hersteller spezifische Hardware. Das Protokoll operiert auf dem Master-Slave-Prinzip, wo ein Master-Gerät, auch Master Station genannt, einen oder mehrere Slave-Geräte, auch Outstation genannt, kontrolliert.

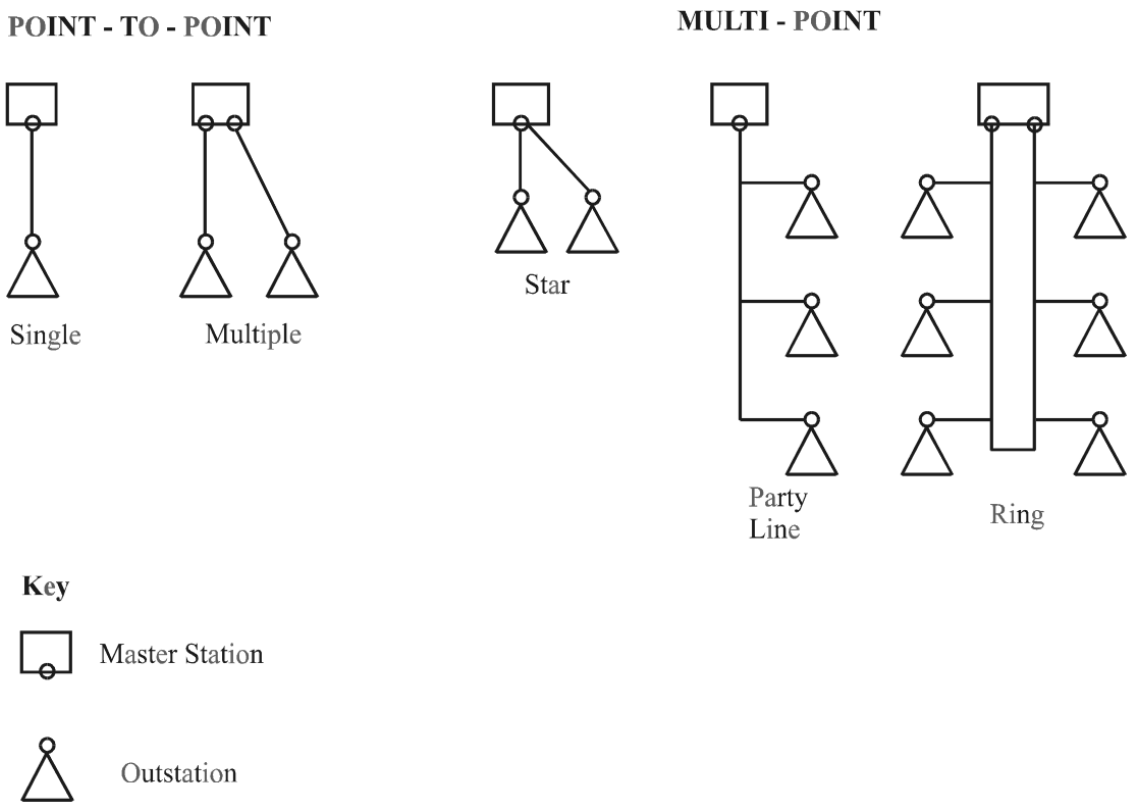


Abb. 10 101er Netzwerkaufbauarten [11, S. 186]

Wie in Abb. 10 zu erkennen ist, gibt es zwei grundlegende Arten Master Stations mit Outstations zu verbinden: Punkt-zu-Punkt-, und Mehrpunkt-Verbindungen. Bei einer Punkt-zu-Punkt-Verbindung ist jeweils eine Master Station mit einer Outstation verbunden und beide können gleichzeitig und unabhängig voneinander kommunizieren. Bei einem Multipunk-Setup sendet die Master Station immer an alle Outstations, diese müssen sich aber alle einen Kanal teilen. Deswegen kann immer nur eine Outstation zu selben Zeit senden [11, S. 185–186].

Auf der Data Link Ebene muss vor allem bei der Energietechnik immer sichergestellt sein, dass auf dem Weg zwischen Netzwarte und Leistungsschalter keine Informationen verfälscht oder verloren werden. Deswegen ist ein 101er Datenpaket, beziehungsweise Frame, so klein wie nötig. Da bei weniger übertragenen Bits die Wahrscheinlichkeit kleiner ist, dass bei einem Frame Fehler auftreten. In Abb. 11 sind die drei verschiedenen Arten an Frames dargestellt: Variable Length Frames, welche Daten übertragen; Fixed Length Frames, welche Control Commands und Acknowledgments übertragen; Single Control Character, welche nur Acknowledgments übertragen.

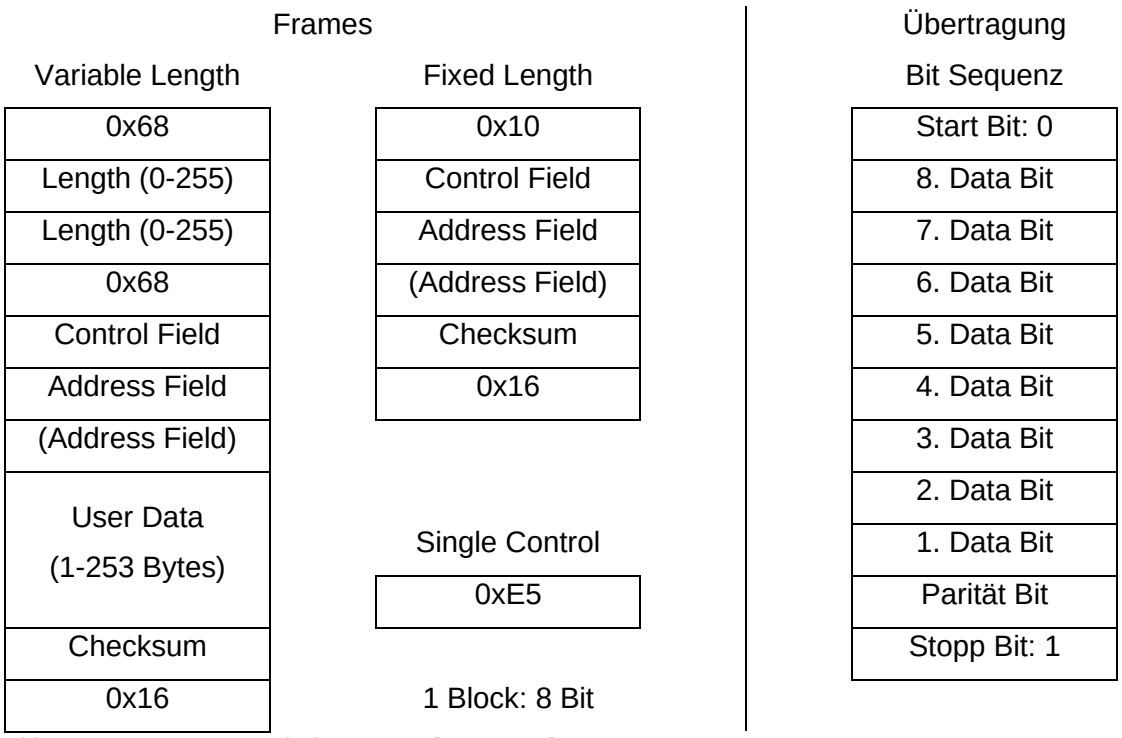


Abb. 11 101er Frames und Bit Sequenz [11, S. 188]

Je nach Konfiguration werden null, ein oder zwei Byte lange Adressen verwendet und die Broadcast Adresse ist dann je nachdem 0xFF oder 0xFFFF. Damit ein Frame als valide akzeptiert wird, müssen beide Length Byte denselben Wert haben und die empfangene Checksum muss mit der selbst berechneten Checksum übereinstimmen. Um diese zu bestimmen wird das komplette Frame Modulo 256 gerechnet. Die Hamming Distance beträgt 4 Bits. Es müssen also mindestens 4 Bits in der Nachricht verfälscht sein, sodass ein inkorrekte Nachricht als fehlerfrei deklariert wird [11, S. 187–189, 202].

Außerdem kann man in Abb. 11 sehen, wie die Bits der Frames übertragen werden. Als erstes wird eine 0 als Startbit übertragen, dann folgen die 8 Data Bit eines Frame Blocks in umgekehrter Reihenfolge, darauf folgt ein Paritätsbit und als letztes kommt eine 1 als Stoppbit. Die empfangenen Bits werden im UART wieder in die „richtige“ Reihenfolge gesetzt [11, S. 189].

Stationports werden in 2 Klassen aufgeteilt: Primary, welche in der Lage sind Kommunikation zu initiieren und Secondary, welche nur auf Nachrichten von Primary Stationports antworten können. Eine Unbalanced Verbindung liegt vor, wenn von zwei miteinander verbundenen Stations eine Primary und die andere Secundary ist. Die Verbindung ist also Unbalanced. Im Gegensatz haben bei einer Balanced Verbindung beide Stations einen Primary und Secondary Port und sind so beide in der Lage Kommunikation zu initiieren. Dies ist aber nur bei Punkt-zu-Punkt-Verbindung möglich.

Deshalb ist es auch möglich die Adresse wegzulassen, da es immer nur einen möglichen Adressaten gibt [11, S. 190–191, 195].

Es gibt drei Arten der Kommunikation zwischen Primary und Secondary Stations: Send/NoReply wird bei Broadcasts und Nachrichten verwendet, wo die Ankunftsbestätigung nicht wichtig ist. Send/Confirm wird bei Nachrichten benutzt, welche wichtig genug sind, dass bestätigt werden muss, dass sie angekommen sind. Request/Respond wird bei Nachrichten verwendet, wo Daten übertragen werden müssen. Ein Confirm wird direkt nach dem Empfangen des Send geschickt, während bei einer Request der Empfänger selbst erst auf Daten zugreifen muss [11, S. 193–194].

P. -> S.	DIR	PRM	FCB	FCV	Function Code			
S. -> P.			ACD	DFC				
	7	6	5	4	3	2	1	0

Code	Bedeutung	Beschreibung
DIR	Direction	Nur bei Balanced Ver.: 1 => A nach B; 2 => B nach A
PRM	Primary	1 => Von P. nach S.; 0 => Von S. nach P.
FCB	Frame Count Bit	Wechselt zwischen 1 und 0 bei jeder P. Nachricht
FCV	Frame Count Valid	1 => FCB ist valid; 0 => Ignoriere FCB
ACD	Access Demand Bit	1 => Class 1 Data verfügbar; 0 => Keine Class 1 Data
DFC	Data Flow Control	1 => Buffer von S. ist voll; 0 => Noch Platz im Buffer

Abb. 12 101er Control Field [11, S. 195–200]

Wie in Abb. 12 zu erkennen ist, besteht das Control Field zu einer Hälfte aus Flow Management Bits und zur anderen Hälfte aus Function Code Bits. Das erste Bit, DIR, zeigt bei einer Balanced Verbindung die Richtung an. Also ob es von A nach B oder von B nach A geschickt worden ist. Bei Unbalanced Verbindungen ist es immer 0. Danach folgt das PRM Bit, welches angibt ob die Nachricht von einer Primary Station an eine Secondary Station oder andersherum gesendet wurde. Das Frame Count Bit ist für das Erkennen für verlorene oder doppelt Nachrichten an der Secondary Station da. Solange das Frame Count Valid Bit 1 ist, muss das FCB bei jeder gesendeten Nachricht der Primary Station geändert werden, damit die Secondary Station die Nachricht akzeptiert. Wenn dies nicht der Fall ist, akzeptiert die Secondary Station keine Nachrichten mehr bis die Primary Station die Verbindung zurückgesetzt hat. Das DFC Bit wird von einer Secondary Station so lange 1 gesetzt, bis in ihren Buffer wieder Nachrichten passen. Währenddessen unterbricht die Primary Station das Senden von Daten, sondern sie sendet Link Status Requests bis das DFC Bit wieder 0 ist. Es gibt

Class 1 und Class 2 Data. Wenn Class 1 Data verfügbar ist, wird das ACD Bit auf 1 gesetzt [11, S. 197].

TABELLE II FUNCTION CODES 101ER [11, S. 195–199]

Primary zu Secondary		Secondary zu Primary	
Code	Beschreibung	Code	Beschreibung
0	Send - Reset Link	0	Confirm - ACK
1	Send - Reset User Process	1	Confirm - NACK
3	Send - User Data	8	Respond - User Data
4	Send - User Data (No Confirm)	9	Respond - NACK No Data
9	Request - Link Status	11	Respond - Link Status
10	Request - User Data Class 1	14	Verbindung funktioniert nicht
11	Request - User Data Class 2	15	Verbindung nicht genutzt

Die Funktion der Funktion Codes ist, wie in TABELLE II zu erkennen ist, abhängig davon, ob die Primary oder die Secondary Station sendet. Auf die Codes 0, 1 und 3 der Primary Station werden ACK oder NACK der Secondary Station erwartet. Auf die Codes 9, 10 und 11 erwidert die Secondary Station mit den entsprechenden Daten. Um eine Verbindung zu initialisieren senden Primary Stationports solange Link Status Requests aus, bis sie eine Link Status Respond erhalten. Daraufhin sendet der Primary Port einen Link Reset und wenn er ein ACK zurückerhält ist die Verbindung aktiv [11, S. 195].

Die übertragenen Daten werden in Form einer Application Service Data Unit übertragen, welche auf **Fehler! Verweisquelle konnte nicht gefunden werden.** zu erkennen ist. Nur eine ASDU kann per Frame übertragen werden.

ASDU	
Data Unit Identifier	Type ID
	Variable Structure Qualifier
	Cause Of Transmission
	Commen Address Of ASDU
Information Object 1	Information Object Address
	Information Elements
Information Object 2	Information Object Address

	Information Elements
…	…
Information Object n	Information Object Address
	Information Elements

Abb. 13 ASDU 101er [11, S. 204]

Die ASDU besteht aus zwei Teilen. Dem Data Unit Identifier und den Daten selbst. Das erste Segment des Data Unit Identifier ist die Type ID und kann Werte zwischen 1 und 255 annehmen, von denen die Hälfte von Herstellern anpassbar sind. Es sind 58 verschiedene Typen von Daten im Standard vordefiniert und diese Typen können in vier grobe Klassen eingeordnet werden: Monitored Information, Control Information, Parameter und File Transfer. Diese werden wiederum in eine Vielzahl an Unterklassen eingeteilt. Außerdem können Datentypen mit oder ohne Zeitstempel, sowie in verschiedenen Zahlenarten übertragen werden. Beispielsweiße hat M_ME_TA_1 die Type ID 10 und steht für Monitored Information, Measured Value, mit Zeitstempel und normalisiert mit Qualitätsbeschreibern [11, S. 204–209].

Darauf folgt der Structure Qualifier. Wenn dessen erster Bit 0 beträgt, werden eine, in den folgenden sieben Bit spezifizierte, Anzahl an Information Objects übertragen, jedes mit eigener Adresse und Zeitstempel. Wenn der erste Bit jedoch 0 beträgt wird nur ein Information Object übertragen, aber mit mehreren Information Elements im selben Datenformat [11, S. 210–211].

Als nächstes kommt das Cause Of Transmission Feld. Es besteht aus drei Teilen: Erst ein Testbit, danach ein Confirm Bit zur Bestätigung von Commands und die letzten 6 Bit ist der eigentliche Cause Of Transmission. Dieser kann zwischen 0 und 63 liegen und gibt, wie der Name sagt, den Grund der Übertragung an. Zum Beispiel ist er 1, wenn eine periodische Übertragung vorliegt oder 13, wenn Files übertragen werden. Wenn in einem Netzwerk mehrere Master Stations sind, wird nach dem Cause Of Transmission noch die Adresse des Senders übertragen [11, S. 211–212].

Das letzte Feld des Data Unit Identifiers ist die Common Address Of ADSU und ist entweder ein oder zwei Oktett Bit lang. Unter dieser Adresse befinden sich alle Daten des ADSU. Meistens ist sie die Stationsadresse, sie kann aber auch in nur einen Teil der Daten einer Station ansprechen, vergleichbar mit IP-Adressen (124.32.0.0/16) [11, S. 213].

Die Information Object Address ist das erste Feld des Information Object. Sie ist 8 bis 24 Bit lang und identifiziert exakt ein Datenobjekt [11, S. 213–214]. Darauf folgen die übermittelten Daten. Es können verschiedene Arten von Daten übertragen werden: Process Data, welche eine Vielzahl von Zahlentypen enthält. Protection Data, welche Daten und Befehle an und von dem Schutzequipment beinhaltet. File Transfer Data, welche sowohl aus den zu übertragenden Dateien an sich, als auch aus den dazugehörigen Dateieneigenschaften und dem stattfindenden Übertragungsmanagment umschließt; Qualifiers für die Stationsverwaltung als auch Time Data und Commands [11, S. 214–218].

IEC 60870-5-104

Um die sich immer weiter ausbreitende Internetinfrastruktur auszunutzen, wurde in der IEC 60870-5-104, kurz 104er, OSI-Layer 1-4 durch die TCP/IP-Suite ersetzt. Wie in Abb. 14 zu sehen ist, wird die komplette Übertragung von Nachrichten zwischen zwei Orten nun per TCP, IP, Ethernet und verschiedenen physischen Medien übernommen.

Layer	Source	Function
Application	IEC 60870-5-101	ASDUs & Information Elements
Transport	Transmission Control Protocol	
Network	Internet Protocol	
Data Link	PPP & HDLC	Ethernet
Physical	X.21	

Abb. 14 IEC 60780-5-104 Layer [11, S. 300–302]

Jedes 104er fähige Gerät hat eine IP-Adresse und einen TCP-Port. Der Standard Port für 104er ist 2404. Über diesen werden TCP-Verbindungen eingerichtet. Jeder ASDU wird eine Application Protocol Data Unit vorangestellt, welche aus der Länge von ASDU und APDU besteht sowie vier Kontrolloktetten. Innerhalb diesen werden empfangene und gesendete Nachrichten gezählt [11, S. 300–304].

Speicherprogrammierbare Steuerungen

Die Steuerungsebene, in der in Echtzeit Signale empfangen, verarbeitet und Neue ausgegeben werden, ist in der Automatisierungstechnik die Schnittstelle zwischen der Feldebene, in jener Sensorik und Aktorik liegen, und der Prozessleitebene, wo der komplette technische Betriebsprozess kontrolliert wird. Die beliebteste Steuerungstechnik ist Speicherprogrammierbare Steuerung, oder auch Programmable Logic Controller genannt. Diese empfängt auf verschiedenen Wegen Daten, verarbeitet diese digital mit Hilfe von vorher geschriebenen Programmen und gibt, auf Basis der gespeicherten Programmierung, unterschiedliche Arten von Daten und Signalen aus

[12, S. 26–35]. Für eine SPS wird zum einen Geräte benötigt, welche Signale lesen und ausgeben können, sowie einen (Micro)Computer, der die programmierte Steuerung speichern und ausführen kann. Desweitern wird eine der Hardware angepasste Software benötigt. Diese wird sowohl für das Zusammenspiel der physischen Einzelteile als auch für die flexible Programmierung benötigt. Es gibt unzählige verschiedene Ein- und Ausgabegeräte, von sehr simplen, welche digitale oder analoge Signale lesen oder ausgeben. bis hin zu sehr komplexen, wie Touch-Displays oder Kommunikationsschnittstellen. Diese Geräte wandeln zwischen externe Signale und Daten und interne verarbeitbare Daten. Der Rechner der SPS kann interne Daten nun seinen temporären Speicher laden und anhand seines Programmes verarbeiten [13, S. 311–314].

In der IEC 61131 wird die Programmierung von SPS normiert, sodass dasselbe Programm auf den Systemen unterschiedlicher Hersteller laufen kann. Es gibt verschiedene grundlegende Elemente in dieser Programmiersprache.
Data Type definiert den Namen, die Größe, die Initialisierung und vieles mehr von Datenarten, wie Boolean, Unsigned Long Integer, Time Of Day oder Double Byte Charactar. Es gibt vordefinierte und benutzer-definierte Data Types und alle Daten, die eine SPS empfängt, speichert oder ausgibt, müssen einem Data Type entsprechen.
Definierte Daten, also jene welche einen potentiell änderbaren Wert oder Inhalt haben, werden Variables genannt und enthalten einen Data Type, einen Namen und einen Wert. Es gibt verschieden Unterarten von Variables, aber alle müssen vor dem Start eines Programmes deklariert sein.
In Program Organization Units, kurz POU, sind die eigentlich Programme entahlten. Sie haben Inputs und Outputs und können mehrfach aufgerufen und ausgeführt werden. POUs können Funktions, Functionblocks, Classes oder Programs sein.
Innerhalb Function wird nach dem Ausführen nichts gespeichert. Deswegen werden normalerweise Function für temporäre Berechnungen benutzt, beispielsweise X:= ADD(A, B). A und B sind die Inputs, X ist der Output und nachdem X ausgegeben wurde, wird kein Teil der Function wieder aufgerufen.
Ein Functionblock wird zur Modularisierung und zur Strukturierung von Programmen benutzt. Jener besteht aus Input, Output und internen Variables, sowie den Operationen mit diesen Daten. Jeder Functionblock hat einen eindeutigen Namen und eigene interne Daten. Zum Beispiel gibt es einen vordefinierten SR-Latch Functionblock, der mit einem Aufruf gesetzt werden und mit einem anderen Aufruf zurückgesetzt werden kann. Der Zustand wird zwischen den beiden Aufrufen gespeichert. Function und Functionblock können jeweils andere Function und Functionblocks aufrufen.

Ein Program ist einem Functionblock ähnlich, soll aber als eigenständiges Programm eingesetzt werden, also aus allen Teilen zusammengesetzt werden, die für die Ausführung eines Prozesses nötig sind. Es kann nur Functionblocks aufrufen, keine anderen Programs. Außerdem können in Programs globale Variablen definiert, sowie auf spezielle Kommunikationsvariablen zugegriffen werden.
Classes, Interfaces und Methods werden in objektorientierter Programmierung verwendet [14, S. 18–155].

Es gibt fünf verschiedene Programmiersprachen, zwei basierend auf Text und drei Graphische. Sie sind alle miteinander austauschbar und stellen dasselbe Programm nur anders da. Es können auch Teile eines Programmes graphisch und andere Teile textuell geschaffen werden. Bei den textuellen Sprachen wird Instruction List, welche Maschinencode sehr ähnlich ist, in der nächsten Version der Norm gestrichen. So wird es nur noch Structured Text geben, welcher an die Programmiersprache Pascal angelehnt ist [14, S. 195–208]. Die graphischen Programmiersprachen sind in der Ablaufreihenfolge und Netzwerkstruktur gleich, nur die Darstellung ändert sich. Die Ladder Diagram Darstellung basiert auf der Darstellung elektrischer Schaltungen. Im Function Block Diagram und im Sequential Function Chart werden einzelne Functions und Functionblocks miteinander verbunden. Beim Function Block Diagram ist die Anordnung der einzelnen Blöcke egal, während ein Sequential Function Chart von oben nach unten abgearbeitet wird. Sollen dort zwei Prozesse parallel stattfinden, müssen diese auch auf derselben Ebene liegen.

CODESYS

Die CODESYS Group mit ihrer Softwareplattform CODESYS stellt IEC 61131 kompatible Software, sowie entsprechende Komplementärsoftware, her. Kern ihres Produktportfolio ist ein modulares Laufzeitsystem für Automatisierungstechnik. Es ist Geräteherstellern möglich alles, von nur dem grundlegenden Laufzeitsystem bis zu einem kompletten Betriebssystem inklusive Programmierumgebung, Cloud und Wartung, einzusetzen. Über 1.000.000 Geräte im Jahre werden mit CODSYSE-Code verkauft [15].

Wago SPS

Die WAGO GmbH & Co. KG, bekannt für ihre Federklemmen, stellt seit 1995 modulare I/O-Systeme her [16]. Sie bestehen aus einem zentralen Steuerungsmodule und können mit einer Vielzahl unterschiedlichen Modulen erweitert werden. Das Laufzeitsystem ist CODESYS V2 und WAGOs eigenes, auf CODESYS V3 bestehendes, e!Runtime. Die Modularität basiert auf einer schraubenlosen und auf einer Tragschiene aneinander steckbaren Design, wie auf Abb. 15 zu erkennen ist.

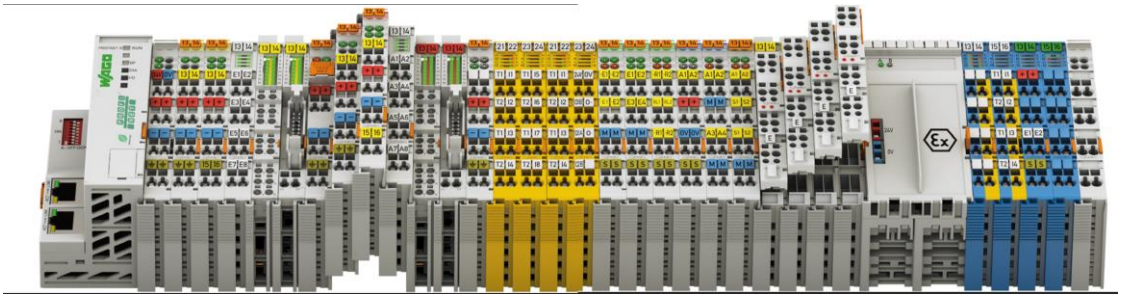


Abb. 15 WAGO I/O SYSTEM 750 [17]

Die WAGO Steuerungen lassen sich in gängige Feldbussysteme eingliedern und mit Hilfe von Erweiterungsmodule lässt sich auch über Mobilfunk oder WLAN mit der Steuerung kommunizieren. Es gibt mehr als 500 Erweiterungsmodule, darunter digital und analoge Ein- und Ausgänge, Zähler, Steppermotorcontroller, Filter und diverse Schnittstellen. Diese werden über einen Lokalbus miteinander verbunden. Dessen Kontakte sind an der Module und werden beim Zusammenstecken automatisch miteinander verbunden [17].

e!Cockpit

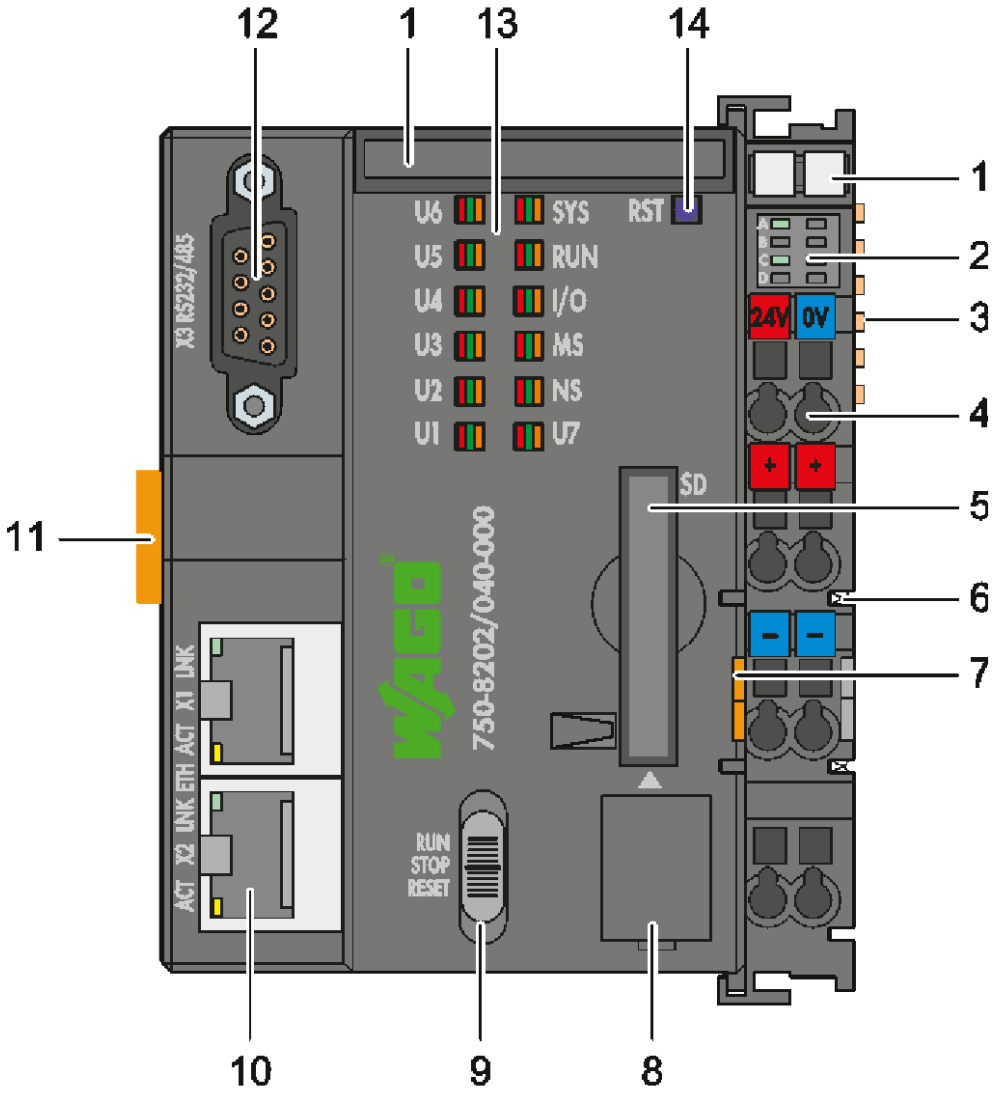
e!Cockpit ist die integrierte Entwicklungsumgebung für WAGO SPS. In dieser können alle Steuerungen und Module konfiguriert und eingerichtet werden. Es kann die Abhängigkeiten sowohl interner als auch externe Geräte eines Netzwerkes verschiedener Protokolle dargestellt als auch bearbeitet werden. e!Cockpit ist IEC 61131 kompatibel und kann erstellte Programme simulieren. Außerdem ist die Visualisierung dort entweder für HMI oder Websites erstellbar. Es ist des Weiteren in der Lage ein laufendes Netzwerk darzustellen, inklusive aller aktueller Werte, Fehlermeldungen und Statusinformationen [18].

Der zentrale Bestandteil des Prüfgerätes ist eine, in Abb. 16 dargestellte, WAGO SPS, die WAGO I/O SYSTEM 750 XTR 750-8202/040-000.

Abb. 16 WAGO I/O SYSTEM 750 XTR 750-8202/040-000 [19]

Diese SPS ist, wie alle Geräte der WAGO 750 Serie, auf einer Hutschiene montierbar und wird im Maschinen- und Anlagenbau, sowie in der Prozessindustrie und Gebäudetechnik eingesetzt. Außerdem können Erweiterungsmodule an das Gerät über den Lokalbus angeschlossen werden. Es kann mit allen IEC-61131-3-kompatiblen Sprachen programmiert werden, entweder mit dem Laufzeitsystem CODESYS 2 und WAGO-I/O-Pro oder dem Laufzeitsystem e!Runtime/CODESYS 3 und e!Cockpit. Für diese Programmierung werden 60 MByte dynamisch verteilter Programm- und Datenspeicher sowie 128 kByte Remenant-Speicher benutzt. Über die beiden RJ-45 Schnittstellen können andere Geräte mit ETHERNET oder Modbus via TCP/UDP angeschlossen werden. Die grundlegende Firmware basiert auf Linux und es ist auf den Controller über Internet zuzugreifen. Dieses findet über das Web-Based-Management statt, wo über HTML-Seiten die Konfiguration und Status dessen

abgerufen und bearbeitet werden kann. Des Weiteren existieren neben der Laufzeitumgebung mehrere Kommunikations- und Verbindungmanagementprogramme auf der SPS. Darunter ein SNMP- und ein SSH-Server/Client, sowie ein Telnet-, ein FTP,- ein DHCP- und ein DNS-Server [20, S. 25–27].

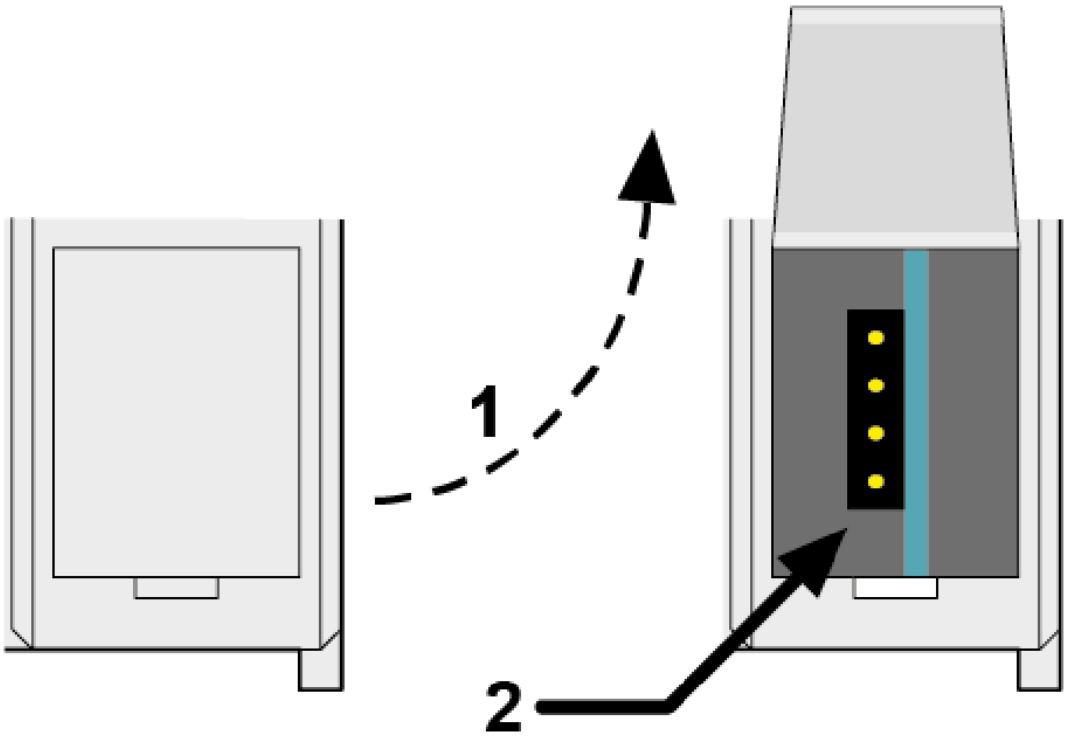


1	Beschriftungsmöglichkeit
2	LED-Anzeigen - Versorgung
3	Datenkontakte - Lokalbus
4	CAGE CLAMP-Anschlüsse für Spannungsversorgung
5	Steckplatz für Speicherkarte
6	Leistungskontakte für Versorgung nachfolgender I/O-Module
7	Entriegelungslasche
8	Service-Schnittstelle (hinter Klappe)
9	Betriebsartenschalter
10	ETHERNET-Anschlüsse - X1, X2
11	Verriegelungsscheibe

12	Serielle Schnittstelle - X3
13	LED-Anzeigen - System
14	Reset-Taster (hinter Bohrung)

Abb. 17 WAGO SPS Ansicht [20, S. 28–29]

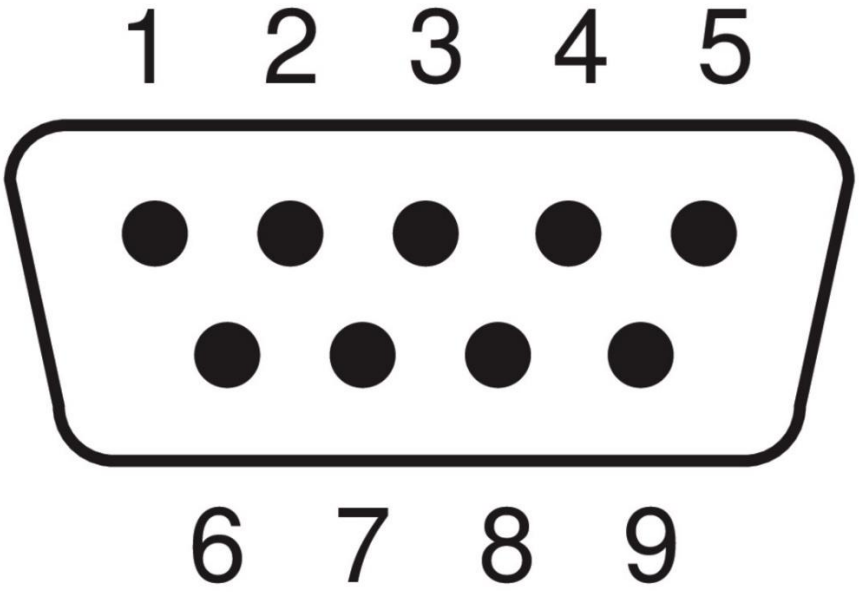
Wie in Abb. 17 dargestellt ist, befindet sich rechts von der Computereinheit die Spannungsversorgung. Diese beträgt 24V DC und wird an dem obersten Kontaktpaar via Klemmkontakt angeschlossen. Darunter befinden sich zwei Kontaktpaare um nicht-WAGO-kompatible Geräte zu versorgen. Um kompatible WAGO-Module zu versorgen befinden sich rechts kleine Kontakte, die beim Zusammenstecken mit anderen Modulen die Spannungsversorgung sicherstellen. Daten werden an diese mit 6 Datenkontakten übertragen. Auch diese werden beim Zusammenstecken automatisch verbunden. In Abb. 18 ist die, mit einer Abdeckklappe geschützte, Serviceschnittstelle zu sehen. Auf diese Art wird die SPS entweder mit Kabel oder über Funk mit einem Computer verbunden. Über diese werden die ETHERNET-Einstellungen der SPS konfiguriert [20, S. 31–34].



1	Abdeckklappe öffnen
2	Service-Schnittstelle

Abb. 18 WAGO SPS Service-Schnittstelle [20, S. 34]

Die serielle Kommunikationsschnittstelle ist ein D-Sub (9) Port und kann entweder über RS-232 oder RS-485 betrieben werden. Wie in Abb. 19 zu sehen ist, müssen die Kontaktbelegungen dem Protokoll angepasst werden, da es sonst zu Schäden an den Kommunikationspartnern kommen kann [20, S. 36–38].



Kontakt	RS-232		RS-485	
	Signal	Beschreibung	Signal	Beschreibung
1	NC	Nicht belegt	NC	Nicht belegt
2	RcD(Out)	Receive Data	NC	Nicht belegt
3	TxD(In)	Transmit Data	A (Tx/Rx+)	Transmit/Receive Data +
4	NC	Nicht belegt	NC	Nicht belegt
5	FB_GND	Masse	FB_GND	Masse
6	NC	Nicht belegt	FB_5V	Versorgung
7	RTS(in)	Request To Send	NC	Nicht belegt
8	CTS(out)	Clear To Send	B (Tx/Rx-)	Transmit/Receive Data -
9	NC	Nicht belegt	NC	Nicht belegt
Gehäuse	Schirm	Schirmung	Schirm	Schirmung

Abb. 19 WAGO SPS Kommunikationsschnittstelle [20, S. 36], [21]

Bei beiden Anzeigeelementen heißt Grün okay, Orange Problem und Rot Fehler. Bei den Anzeigeelementen des Systems steht SYS für den Systemstatus, RUN für den Programmstatus, I/O für den Lokalbusstatus, MS für den Modulstatus und NS ist unbelegt. U1 bis U7 sind programmierbare Anwender-LEDs [20, S. 39–40]. Mit dem Betriebsartenschalter wird das Laufzeitsystem und damit auch die zu laufenden Programme gestartet und gestoppt. Das Laufzeitsystem kann mit verschieden langem Ziehen des Betriebsartenschalters entweder teilweise oder komplett zurückgesetzt werden. Mit der Reset-Taste, welche mit beispielweise einer aufgebogenen Büroklammer bedient wird, kann je nach Position des Betriebsschalter die IP-Einstellungen zurückgesetzt, die Software neugestartet oder das Gerät auf

Werkseinstellungen zurückgesetzt werden. Es können Programme entweder auf dem Gerät selbst oder auf einer SD-Karte gespeichert werden [20, S. 43–45].

Nachdem Montieren des Controllers und der zugehörigen Module, sowie des Anschließens der Daten- und Stromleitungen, als auch der Erdung und Schirmung der Module, kann das System über das Einschalten des Netzteiles angeschaltet werden. Wenn das System erfolgreich gestartet wurde, leuchten die SYS- und I/O-LED grün. Wenn sich auch ein ausführbares IEC-61131-3 Programm im Speicher befindet, leuchtet auch die RUN-LED grün. Um die SPS programmieren zu können, muss ihr eine IP zugewiesen werden. Dazu muss die SPS ausgeschaltet, über die Service-Schnittstelle mit einem PC verbunden und danach wieder angeschaltet werden. Daraufhin kann ihr über das, in Abb. 20 gezeigte, „WAGO Ethernet Settings“ Programm entweder eine statische oder dynamische IP zugewiesen werden [20, S. 110–111].

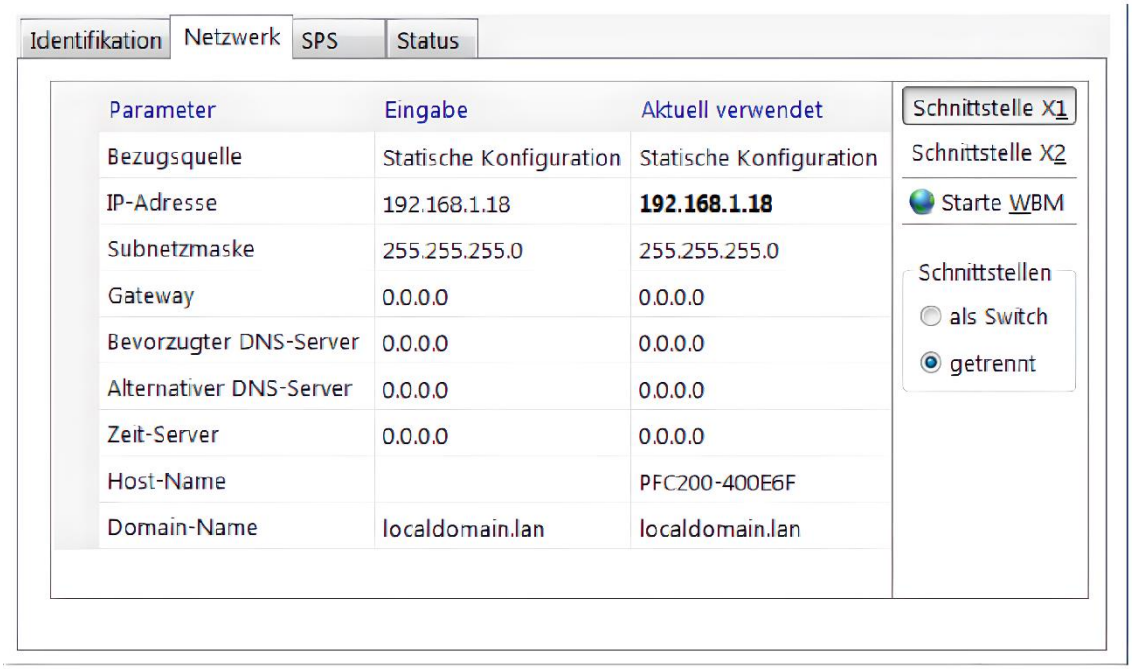


Abb. 20 WAGO Ethernet Settings

Die WAGO SPS kann Modbus TCP, Modbus UDP und Modbus RTU gleichzeitig betreiben. Es werden zehn Modbus Function Codes unterstützt: FC1 bis FC6, FC15, FC16, FC22 und FC23. Es können bis zu 1000 Register gespeichert und gelesen werden. Bei Modbus RTU kann die Geräte ID, die Antwortzeit, die Baudrate (zwischen 1200 und 115200), die Anzahl der Stoppbits, die Art der Parität und die Schnittstellenart (RS-232 oder RS485) konfiguriert werden. In der SPS sind auch WAGO spezifische Register implementiert zum Konfigurieren und zum Status Auslesen des Controllers. Diese liegen zwischen 0x1000 und 0x2FFF und sind so außerhalb des IEC-61131 Adressbereich [20, S. 195–320].

Eine Ortnetzstation der SNB muss folgende Fernwirkbefehle und Meldungen übertragen: Fernschaltung der Lasttrennschalter, Lasttrennschalterstellungsmeldung, Kurzschlussanzeigemeldung, Stromversorgungstörmeldung, Fernwirkkomponentensystemmeldung, Schaltanlagenstörmeldung und Überstromzeitschutzgerätmeldung [22, S. 6].

Die 10kV-Schaltanlage ist mit 24V DC Motorantrieben ausgerüstet, welche in unter zwei Sekunden in die jeweilige Endstellung fahren. Der gesamte Schaltvorgang dauert weniger als 30 Sekunden [22, S. 6–7].

Die Unterbrechungsfreie Stromversorgung und das Fernwirkgerät sind Teileinschubkassetten und werden zusammen in einem Montagerahmen montiert, siehe Abb. 21. Dieser kann entweder senkrecht oder waagrecht in der Station montiert sein. Wenn eines der beiden Module einen Fehler hat, kann es so einfach wieder ausgetauscht werden [22, S. 9–10].

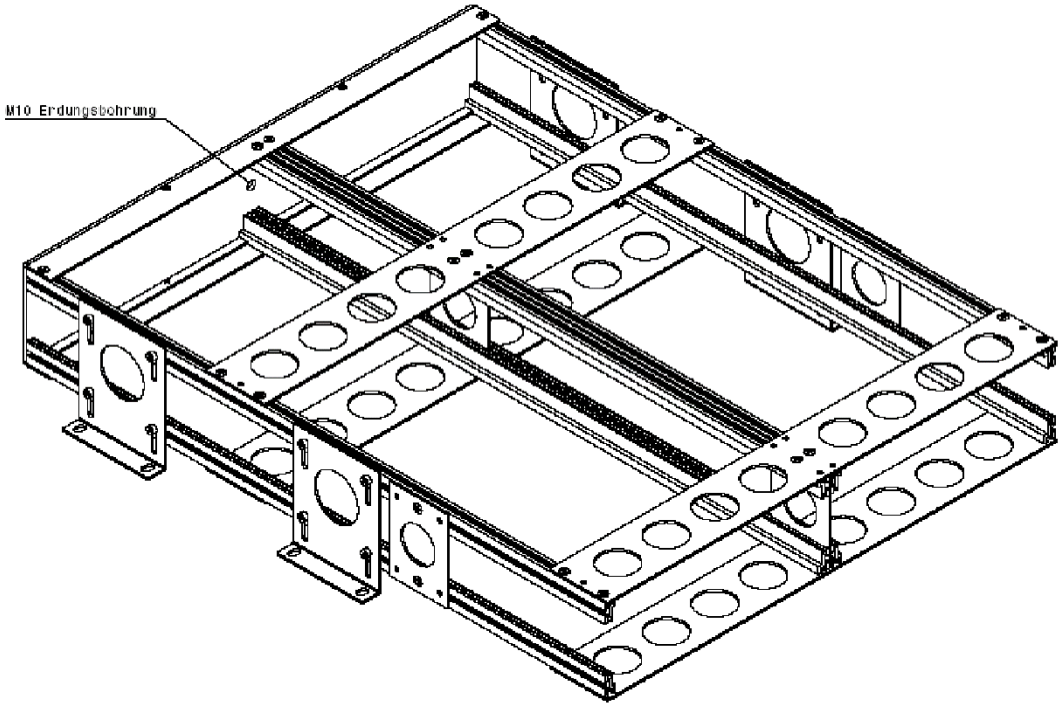


Abb. 21 Montagerahmen [22, S. 25]

Die Übertragungs- und Fernsteuerungstechnik sowie die Stromversorgung sind für einen Temperaturbereich von -20°C bis + 55°C ausgelegt [22, S. 16].

Die Unterbrechungsfreie Stromversorgung speichert Energie mit Hilfe von Kondensatoren. Sollte die externe Netzversorgung unterbrochen werden, übernehmen die aufgeladenen Kondensatoren die Energieversorgung. Die USV versorgt das FWG,

eine Modemeinheit und die Schalter der Mittelspannungsschaltanlage und auch im Normalbetrieb werden diese Systeme durch die USV versorgt. Die Kondensatoren können mindestens 46,2kJ speichern. Wie auf zu erkennen ist, kann die Fernwirkgerät, das Modem und die Mittelspannungsschaltanlage jeweils separat an- und abgeschaltet werden. Über LEDs wird das Vorhandensein der Netzversorgung und der Motorspannung, sowie das Sinken des Ladezustands unter 30%, als auch eine Erdschlussmeldung angezeigt [23, S. 3–7].

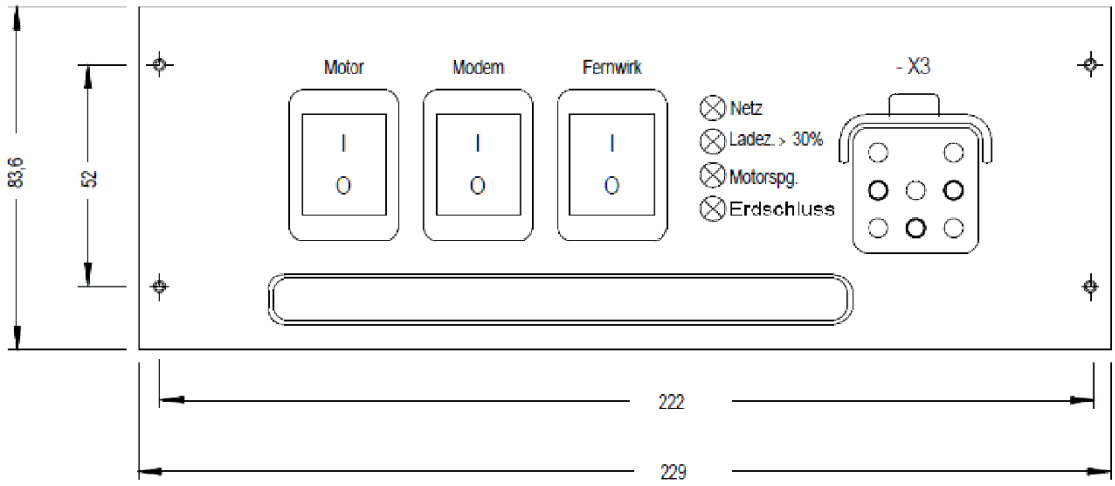


Abb. 22 Unterbrechungsfreie Stromversorgung [23, S. 7]

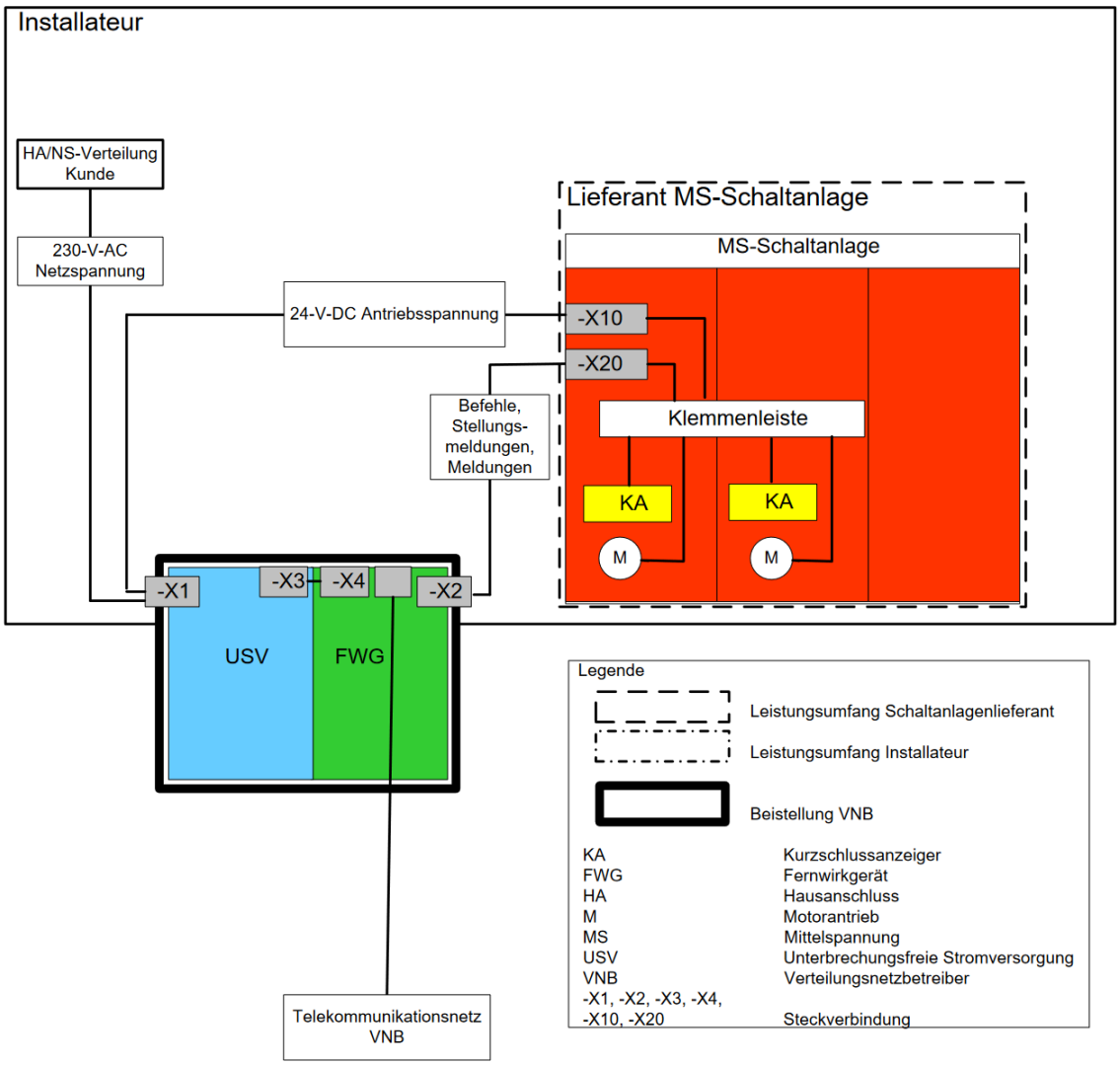


Abb. 23 Funktionsschema in einer Mittelspannungsstation [23, S. 4]

Verzeichnisse

Quellenverzeichnis

[1] L. L. Peterson und B. S. Davie, *Computer Networks; a systems approach*. Cambridge, United States: Elsevier Inc., 2022.

[2] J. Axelson, *Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks.* Madison: Lakeview Research, 2002.

[3] Adam Chapweske, „StuBS: The PS/2 Mouse/Keyboard Protocol“. https://www.sra.uni-hannover.de/Lehre/WS21/L_BST/doc/ps2.html (zugegriffen 21. Juni 2022).

[4] Thomas Kugelstadt, „Texas Instruments: The RS-485 Design Guide“. Mai 2021. Zugegriffen: 20. Juni 2022. [Online]. Verfügbar unter: https://www.ti.com/lit/an/slla272d/slla272d.pdf

[5] B. Drury, *The control techniques drives and controls handbook*, 2nd ed. Stevenage: Institution of Engineering and Technology, 2009.

[6] „Modbus FAQ“. https://modbus.org/faq.php (zugegriffen 5. Mai 2022).

[7] „About Modbus | Simply Modbus Software“. http://www.simplymodbus.ca/FAQ.htm (zugegriffen 5. Mai 2022).

[8] „Modbus Plus | Schneider Electric USA“. https://www.se.com/us/en/product-range/576-modbus-plus/ (zugegriffen 5. Mai 2022).

[9] Modbus Organization, Inc, „MODBUS over Serial Line Specification and Implementation Guide V1.02“. 20. Dezember 2006. Zugegriffen: 20. Mai 2022. [Online]. Verfügbar unter: https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf

[10] MODICON, Inc., Industrial Automation Systems, „Modicon Modbus Protocol Reference Guide“. Juni 1996.

[11] Gordon Clarke, Deon Reynders, und Edwin Wright, *Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems*. Burlington, MA 01803: Elsevier, 2004.

[12] O. Leps, *Hybride Testumgebungen für kritische Infrastrukturen: effiziente Implementierung für IT-Sicherheitsanalysen von KRITIS-Betreibern*. Wiesbaden [Heidelberg]: Springer Vieweg, 2018.

[13] B. Heinrich, P. Linke, und M. Glöckler, *Grundlagen Automatisierung: Sensorik, Regelung, Steuerung*, 2., Überarbeitete und Erweiterte Auflage. Wiesbaden [Heidelberg]: Springer Vieweg, 2017. doi: 10.1007/978-3-658-17582-5.

[14] Commission Electrotechnique Internationale, International Electrotechnical Commission, und International Electrotechnical Commission, *IEC 61131-3*. Genève: IEC, 2013.

[15] „Warum CODESYS?“ https://de.codesys.com/das-system/warum-codesys.html (zugegriffen 30. Juni 2022).

[16] „WAGO – Ihr Partner für Automatisierungs- und Verbindungstechnik“, *WAGO Deutschland*. https://www.wago.com/de (zugegriffen 30. Juni 2022).

[17] WAGO Kontakttechnik GmbH & Co. KG, „WAGO I/O SYSTEM 750“. Zugegriffen: 30. Juni 2022. [Online]. Verfügbar unter: https://www.wago.com/de/d/Info_60472821

[18] „e!COCKPIT“. https://techdocs.wago.com/Software/eCOCKPIT/de-DE/index.html#96787723 (zugegriffen 30. Juni 2022).

[19] „Controller PFC200 (750-8202/040-000) | WAGO“. https://www.wago.com/global/plcs-%E2%80%93-controllers/controller-pfc200/p/750-8202_040-000#downloads (zugegriffen 4. Juli 2022).

[20] WAGO Kontakttechnik GmbH & Co. KG, „Handbuch WAGO I/O SYSTEM 750 XTR 750-8202/040-000 PFC200 2ETH RS XTR Controller PFC200; 2 × ETHERNET;

RS-232/-485; extrem“. 3. Januar 2007. Zugegriffen: 4. Juli 2022. [Online]. Verfügbar unter: https://www.wago.com/medias/m07508202-00400000-0de.pdf?context=bWFzdGVyfGRvd25sb2Fkc3w2OTg5MzA4fGFwcGxpY2F0aW9uL3BkZnxoZjcvaDBjLzEyMjAwNDUwOTgxOTE4L20wNzUwODIwMl8wMDQwMDAwMF8wZGUucGRmfDk5M2U4YjM1YjhkZjAxNmEyZTJmMGFhNDMyNDRkZTUzMDI2NWZmYjk1YmFhZGFiY2E0MmEyN2RiYWM2YTg5MmI&attachment=true

[21] „Connection Line SUB D, 9-pin (ZCLL001)“, *wenglor sensoric group*. https://www.wenglor.com/en/System-Components/Connection-Equipment-and-Connection-Boxes/Connection-Line-SUB-D-9-pin/p/ZCLL001 (zugegriffen 4. Juli 2022).

[22] Stromnetz Berlin GmbH, „Technische Beschreibung 3340 FERNSTEUERUNG VON NETZ- UND KUNDENSTATIONEN IM MS-NETZ“. 2. Mai 2019.

[23] Stromnetz Berlin GmbH, „Technische Beschreibung 3341 Unterbrechungsfreie Stromversorgung (230-V-AC/24-V-DC) zur Hilfsenergieversorgung von motorischen Antrieben und Fernwirkgeräten“. 16. Februar 2011.

Abbildungsverzeichnis

Tabellenverzeichnis