DIPLOMARBEIT

Herr
Maximilian Lichan

Performance- und Systemvergleich von homogenen und inhomogenen industriellen Steuerungssystemen unter besonderer Berücksichtigung des Feldbusses

2015

Fakultät **Elektro- und Informationstechnik**

DIPLOMARBEIT

Performance- und Systemvergleich von homogenen und inhomogenen industriellen Steuerungssystemen unter besonderer Berücksichtigung des Feldbusses

Autor:
Maximilian Lichan

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
ET14sA-M

Erstprüfer:
Prof. Dr.-Ing. Sven Schmeißer

Zweitprüfer:
Prof. Dr.-Ing. Dietmar Römer

Mittweida, Juni 2015

Bibliografische Angaben

Lichan, Maximilian: Performance- und Systemvergleich von homogenen und inhomogenen industriellen Steuerungssystemen unter besonderer Berücksichtigung des Feldbusses, 108 Seiten, 79 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Elektro- und Informationstechnik

DIPLOMARBEIT, 2015

Satz: LaTeX

Referat

Der Inhalt der Diplomarbeit umfasst eine Untersuchung der Performance und der Flexibilität von verschieden modernen Steuerungsmodellen. Hierbei wird zunächst der Feldbus hinsichtlich der Leistungsfähigkeit betrachtet. Die zweite Betrachtungsweise bezieht sich auf die CPU Performance des SPS- Gesamtsystems. Hierbei werden unterschiedliche Steuerungsgenerationen miteinander verglichen. Als Steuerungen werden zum einem Mirco- SPS und zum anderen klassische Kompaktsteuerungen untersucht.

This Diploma thesis examines the issue “performance and system comparison of homogeneous and inhomogeneous industrial control systems with special emphasis on the field bus“. It is investigating whether a loaded fieldbus can deliver cycle times and performance and weather this implementation is independent of the duration of the telegram stacks, CPU performance, telegram processing and software.

I. Inhaltsverzeichnis

Inhaltsverzeichnis I

Abbildungsverzeichnis II

Tabellenverzeichnis III

Abkürzungsverzeichnis IV

Danksagung V

1 Einleitung 1

2 Grundfunktion und Randbedingungen der Testumgebung 3

3 Technologie und Applikationen der Systeme 9

3.1 PROFINET- Kommunikation . . . 11
3.1.1 SIEMENS Anwendungssysteme . . . 13
3.1.2 TURCK Anwendungssystem . . . 19
3.1.3 PhoenixContact Anwendungssysteme . . . 21
3.2 MODBUS- Kommunikation . . . 23
3.2.1 SchneiderElectric Anwendungssystem . . . 24
3.2.2 PhoenixContact Anwendungssystem . . . 25
3.2.3 TURCK Anwendungssystem . . . 26
3.3 EtherCAT- Kommunikation . . . 26
3.3.1 BECKHOFF Anwendungssystem . . . 28
3.4 SERCOS III- Kommunikation . . . 29
3.4.1 BOSCH Rexroth Anwendungssystem . . . 32
3.5 POWERLINK- Kommunikation . . . 33
3.5.1 B&R Anwendungssystem . . . 35
3.6 PROFIBUS- Kommunikation . . . 36
3.6.1 SIEMENS Anwendungssystem . . . 38
3.7 Gesamtübersicht . . . 41

4 Systembetrachtungen 44

5 Implementierung 48

5.1 SIEMENS Implementierung . . . 48
5.2 PCworx Implementierung . . . 52
5.3 Unity Implementierung . . . 55
5.4 CodeSys Implementierung . . . 57
5.5 IndraWorks Implementierung . . . 61
5.6 TwinCat Implementierung . . . 62
5.7 AutomationStudio Implementierung . . . 65

6 Performance **68**
6.1 Monitoringsystem 68
6.2 Auswertung 76

7 Fazit **84**

Literaturverzeichnis **86**

Anhang **90**

Anhang A FUP Programm **90**

Anhang B Wireshark- Mitschnitt für Anlauf von S7-317 **95**

Anhang C OVIS Visualisierung für VT250 **96**

Anhang D B&R Visualisierung PowerPanel 65 **97**

Anhang E Visualisierungen des Monitoringsystems **98**

Anhang F Fotografie **100**

II. Abbildungsverzeichnis

2.1 Dekadischer Aufbau 4

2.2 Programmcode 1. Zähldekade 5

2.3 Programmcode 2. Zähldekade 6

3.1 Profinet Telegrammaufbau mit den Leistungsklassen [10, S.74] 12

3.2 Profinet Ethernet Frame [7, S.7] 13

3.3 S7315F-2 PN/DP Rackaufbau 13

3.4 ET200s Rackaufbau 14

3.5 S7315F-2 PN/DP– ET200s Systemaufbau 14

3.6 S7-1511 Rackaufbau 15

3.7 S7-1511- 1 PN– ET200s Systemaufbau 15

3.8 S7-1516- 3 PN/DP Rackaufbau 16

3.9 S7-1516- 3 PN/DP– ET200s Systemaufbau 16

3.10 S7315F-2 PN/DP– WAGO 750-370 Systemaufbau 17

3.11 WAGO 750-370 Rackaufbau 17

3.12 S7- 317T– inhomogener Aufbau 18

3.13 Inline ILC 350PN– inhomogener Aufbau 21

3.14 Axioline Profinet– inhomogener Aufbau 22

3.15 Axioline Profinet– homogener Aufbau 23

3.16 Modbus Telegramm [11] 24

3.17 Axioline Modbus– homogener Aufbau 25

3.18 Aufbau Ethernet Frame mit EtherCAT Protokolldaten [51] 27

3.19 Gesamtaufbau 28

3.20 Sercos Topologievarianten [14] 30

3.21 Sercos Querverkehr [15] 31

3.22 Powerlink SCNM im klassischen Ethernet 33

3.23 Zyklusperioden der POWERLINK Kommunikation 34

3.24 Kommunikation unter Profibus [17, modifiziert] 37

3.25 S7315F-2 PN/DP– ET200s Systemaufbau ProfiBus 39
3.26 ET200s Rackaufbau 39
3.27 S7 413-2DP– ET200s Systemaufbau ProfiBus 40
3.28 SIMATIC Microbox PC 427B– ET200s Systemaufbau ProfiBus 40
4.1 Einflussfaktoren auf das Performanceverhalten des Systemaufbaus 44
5.1 Mindestzykluszeit 49
5.2 Kommunikationslasteinstellungen bei Controllern im TiaPortal 49
5.3 Optimierter Bausteinzugriff 50
5.4 PhoenixContact Inline Problem in Step7 51
5.5 Implementierung von inhomogenen Komponenten 52
5.6 Unterstützte MODBUS- Funktionen 53
5.7 Modbusregistereditor 53
5.8 Prozessdatenzuordnung 55
5.9 Modicon TSX Busaufbau 56
5.10 Client/Server Adresstabelle 56
5.11 Prozessdatenzuordnung über E/A Scanner 57
5.12 Ethernetadapter für ProfiNet oder Modbus 58
5.13 VT250– ProfiNet Netzwerk 59
5.14 VT250– Modbus Netzwerk 59
5.15 Buskoppler– Modbus Slave-Kanal 60
5.16 ModbusTCP Slave I/O Abbild 60
5.17 IndraWorks Projektstruktur 62
5.18 Compatible TwinCAT RT-Ethernet Adapter 63
5.19 Busstruktur 64
5.20 X20CP– CPU Zeitverhalten konfigurieren 66
5.21 POWERLINK Zykluszeit konfigurieren 67
6.1 System Management Console mit Integration Server 71
6.2 HOME– InTouchoberfläche 73
6.3 Tabellen Design 74
6.4 Auswertung– InTouchoberfläche 75
6.5 Zählerstandentwicklung nach einer Laufzeit von 3 Tagen 76

6.6 Zählerstandentwicklung nach einer Laufzeit von 3 Tagen 77
6.7 ProfiNet Kommunikation in Wireshark 79
6.8 Modbus Kommunikation in Wireshark 80
6.9 Zykluszeit der Turck VT250 81
6.10 Zykluszeit der BoschRexroth IndraControl L45 81
F.1 System 1, 2 (v. O.) 100
F.2 System 3 100
F.3 System 4 101
F.4 System 5 101
F.5 System 6, 12 102
F.6 System 7 102
F.7 System 8 103
F.8 System 9 103
F.9 System 10 104
F.10 System 11 104
F.11 System 13 105
F.12 System 14 105
F.13 System 15 106
F.14 System 16 106
F.15 System 18 107
F.16 System 17, 19 (v. O.) 107

III. Tabellenverzeichnis

2.1 Visuelle Darstellung des aktuellen Zählerstandes 4

3.1 Gesamtübersicht der Systeme 43

4.1 Systeme und deren Performancevorraussetzungen 45

4.2 Prognostizierte Zählerentwicklungen anhand der Zykluszeiten 47

6.1 Bearbeitungszeiten für Dekade 2 81

IV. Abkürzungsverzeichnis

EPSG ..... Ethernet POWERLINK Standardization Group, Seite 33

EtherCAT ..... Ethernet for Control Automation Technology, Seite 26

FB ..... Funktionsblock, Seite 7

FUP ..... Funktionsplan, Seite 5

GSDML ..... General Station Discription, Seite 12

M/S ..... Master/ Slave Telegramme, Seite 31

MBP ..... Manchester Bus Powered, Seite 37

MDT ..... Acknowlege Telegramme, Seite 31

OB ..... Operationsbaustein, Seite 7

ODBC ..... Open Database Connectivity, Seite 75

OPC ..... Open Platform Communications, Seite 68

OSI ..... Open Systems Interconnection Model, Seite 24

Profibus ..... PROcess FIeld BUS, Seite 36

PROFINET ..... Process Field Network, Seite 11

SCNM ..... Slot Communication Network Management, Seite 33

SDDML ..... SERCOS Device Description Makeup Language, Seite 29

SERCOS ..... SErial Realtime Communication System, Seite 29

SPDML ..... SERCOS Profile Description Markup Language, Seite 29

SPS ..... Speicherprogrammierbare Steuerung, Seite 1

TwinCAT ..... The Windows Control and Automation Technology, Seite 62

UDP ..... User Datagram Protocol, Seite 12

V. Danksagung

An dieser Stelle möchte ich mich bei all denen bedanken, die durch ihre fachliche und persönliche Unterstützung zum Erfolg meines Studiums beigetragen haben.

Mein Dank gilt zunächst Herrn Prof. Dr.-Ing. Schmeisser für die Bereitstellung dieses Themas. Seine Anregungen und Hinweise haben stets mein Interesse am vorliegenden Aufgabengebiet erweitert und eine zielgerichtete Arbeitsweise ermöglicht.

Weiterhin möchte ich mich bei den Firmen TURCK, PhoenixContact und B&R für die Bereitstellung der Gerätschaften und der freundlichen Beratung bedanken. Durch die Leihgabe dieser Geräte wurde ein noch umfassenderer Vergleich möglich. Zusätzlich, zu den während des Studiums bereits bekannten Geräten, konnte ich umfassende Erfahrungen und Erkenntnisse mit diesen Steuerungssystemen sammeln.

Ein besonderer Dank gilt meiner Familie. Vor allem haben mich meine Eltern und Großeltern bei der Erstellung dieser Arbeit und auch während meines Studiums in vielen fachlichen aber auch alltäglichen Dingen unterstützt und motiviert. Vor allem meinem Opa 'Fred', meiner Oma 'Regine' und meinem Vater für die Zeit die in der Korrektur dieser Arbeit investiert wurde.

Einen nicht unwesentlichen Anteil an das Gelingen meines Studiums hatten natürlich auch meine Mitkommilitonen der Seminargruppe 'EI11'. Die vielen schönen und amüsanten Abende und Unternehmungen, aber auch die tatkräftige Unterstützung in den ersten Semestern, haben über manche Probleme hinweggeholfen. Besonderer Dank gilt meinen Freunden Christian, Marco und Thomas. Ihr habt die Studienzeit hier in Mittweida für mich zu einem sehr schönen Lebensabschnitt gemacht.

1 Einleitung

Aus der Industrie- und Prozessautomatisierung ist die speicherprogrammierbare Steuerung nicht mehr wegzudenken. Sie hat sich aufgrund der rasanten Weiterentwicklung der Halbleitertechnik und der daraus hervorgehenden stetigen Steigerung der Rechenleistung permanent weiterentwickelt. Durch diese Entwicklung wurde es möglich, objektorientierte Steuerungen von der Micro-SPS bis zum Hochleistungsmultiprozessorsystem in allen Industriebereichen zu etablieren. Damit werden heutzutage erweiternde Technologiefunktionalitäten wie Motion- und RobotControl sowie Industrial Ethernet basierende Feldbussysteme gleichzeitig abgebildet. Die heutige SPS übernimmt eine Vielzahl unterschiedlicher Aufgaben in komplexen Automatisierungsanlagen. Voraussetzung dafür ist die enorme Rechenkapazität der heutigen Prozessorengeneration.

Der Inhalt der Diplomarbeit umfasst eine Untersuchung der Performance und der Flexibilität von verschieden modernen Steuerungsmodellen. Hierbei wird zunächst der Feldbus hinsichtlich der Leistungsfähigkeit betrachtet. Es wird untersucht, ob ein belasteter Feldbus die Zykluszeiten und Performance einhält und ob dieses unabhängig von der Programmlaufzeit der CPU, Telegrammverarbeitung oder Software Implementierung erfolgt. Für diese Untersuchung werden schlussendlich unterschiedliche Feldbusse miteinander verglichen. Die Basis hierfür sind systemspezifische Master Module verschiedener speicherprogrammierbarer Steuerungen und herstellerspezifischer oder herstellerneutraler Slave Module.

Die zweite Betrachtungsweise bezieht sich auf die CPU Performance des SPS- Gesamtsystems. Hierbei werden unterschiedliche Steuerungsgenerationen miteinander verglichen. Als Steuerungen werden zum einen Micro- SPS und zum anderen klassische Kompaktsteuerungen untersucht.

Zur Erreichung dieser Zielsetzung werden unterschiedliche Systeme in Betrieb genommen. Jedes System besteht aus einer SPS und 5 Slave-Teilnehmer. Die Masteranschaltung für den Feldbus ist jeweils in der SPS implementiert. Die Slave-Teilnehmer werden durch Buskoppler oder E/A Devices realisiert. Die Untersuchung erfolgt auf Basis von verschiedenen herstellerspezifischen Masteranschaltungen deren Slave-Module basierend auf eine homogene und inhomogene Netzstruktur miteinander kommunizieren. Hierbei sollen Erkenntnisse gezogen werden, ob dies Performenceunterschiede im Vergleich zu den Systemen hervorruft.

Vergleichbarkeit setzt gleiche Bedingungen voraus!

Für die Bewertung der Performance der Systeme und die Belastung der Feldbusse wird ein 5-fach kaskadierter Zähler erstellt. Dieses Programm wird auf jedem System zu den gleichen Bedingungen umgesetzt. Die aktuellen Zählwerte des kaskadischen Zählers werden über die 5 Buskoppler auf den digitalen Ausgängen der 5 Ein-/ Ausgabe Module binär angezeigt. Diese Buskoppler haben jeweils 8 Ein- und 8 Ausgabekanäle. Dies ermöglicht eine binäre Darstellung der maximalen Zählbreite der Ausgabemodule von 256.

Die Umsetzung des Projekts erfolgt mit der jeweiligen Entwicklungsumgebung des Steuerungsherstellers. In dieser wird der geschachtelte Zähler programmiert und die Hardware konfiguriert. Damit ist die Basis für die Vergleichbarkeit der unterschiedlichen Systeme gegeben. Zusätzlich wird ein Monitoringsystem zur visuellen Darstellung, Aufzeichnung, Archivierung und Auswertung der Ergebnisse implementiert. Die Daten hierfür werden über einen SQL Server aufgezeichnet. Der Datenaustausch für dieses Monitoringsystem wird durch ein, für die Systeme geeigneten OPC Server realisiert.

2 Grundfunktion und Randbedingungen der Testumgebung

In diesem Kapitel wird die Grundfunktion des entwickelten Programms zur Untersuchung der Performance der Systeme näher erläutert. Im zweiten Teil dieses Kapitels sind die Festlegungen und Bedingungen für die Controller- und Taskeinstellungen definiert. Diese Definitionen sind die Basis für die Vergleichbarkeit der Systeme untereinander.

In dieser Arbeit wurde ein 5-fach gestufter Zähler programmiert. Jeder einzelne Zählerstand wird auf einem Buskoppler ausgegeben. Die Zählerstände werden über das Bussystem von der Masteranschaltung der SPS an den entsprechenden Buskoppler übermittelt. Jeder einzelne Zähler von 1 bis 5 stellt eine Kaskade dar. Sobald ein Startsignal vom Monitoringsystem anliegt, zählt der Zähler 1 von 1 bis n hoch. Hat dieser Zähler die vorgegebene Zählbreite n erreicht, gibt es ein Überlaufsignal. Anschließend wird der Zähler zurückgesetzt und der Zähler 2 um eins inkrementiert. Hat der Zähler 2 die Zählbreite n erreicht, ist das Funktionsprinzip gleich wie im Zähler 1. Es gibt ein Überlaufsignal, der Zähler wird durch ein Reset auf den Zählwert Null zurückgesetzt und die dritte Zählkaskade 'Zähler 3' wird um eins erhöht. Alle Zähler erhalten ein Taktsignal zum Inkrementieren beim Überlauf der Zählbreite des Vorgängerzählers. Diese Methode wird wiederholt bis der Zähler 5 die Zählbreite erreicht hat und der Prozess damit beendet wird.

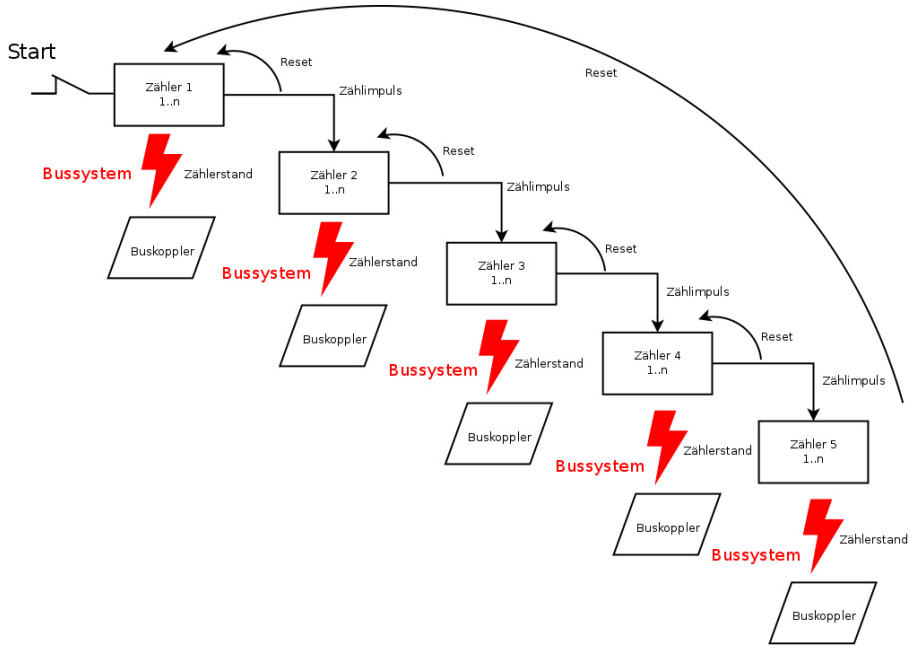


Abbildung 2.1: Dekadischer Aufbau

Wie in der Abbildung 2.1 dargestellt ist, arbeitet der Zähler nur wenn das externe Startsignal des Monotoringsystems anliegt. Somit beginnen die einzelnen Zähler ihren Wert fortlaufend pro CPU Zyklus zu inkrementieren. Die aktuellen Zählwerte werden über die unterschiedlichen Feldbustechnologien auf die 5 Slavegeräten gesendet. Die Zählerstände werden als BCD- Zahl auf den Buskopplern dargestellt. Die Buskoppler können auf ihren digitalen Ausgangskanälen genau 1 BYTE darstellen. Somit kann auf den Buskopplern eine maximale Zählbreite von 256 erreicht werden.

Bit -Nr.	**4**	**3**	**2**	**1**
Wertigkeit	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Tabelle 2.1: Visuelle Darstellung des aktuellen Zählerstandes

Die Tabelle 2.1 stellt beispielhaft die möglichen Zählerstände für eine Zählbreite von

10 in Binärschreibweise dar. Hierfür wurde auf die jeweiligen Ausgabemodule der Buskoppler die jeweiligen Kanäle gesetzt oder nicht gesetzt. Wenn diese Ausgabekanäle gesetzt sind, wird dies durch eine LED am jeweiligen Kanal visualisiert. Die Orange dargestellten Tabellenzellen kennzeichnen das Schalten der Kanal LEDs.

Das Programm wurde in allen Programmierumgebungen durch die Programmiersprache Funktionsbaustein FBS/ FUP umgesetzt. Diese Programmiersprache entspricht dem Softwarestandard IEC 61131-3. Die EN 61131-3 ist die weltweit einzige gültige Norm für Programmiersprachen von Steuerungen [1, Vgl.]. Um eine Vergleichbarkeit auf allen herstellerspezifischen Steuerungssystemen zu gewährleisten, musste eine Programmiersprache gewählt werden die von allen Herstellerentwicklungsumgebungen unterstützt wird. Nach einer Recherche wurde festgestellt, dass dies durch die Programmiersprache “Funktionsbaustein“ auf allen Steuerungssystemen realisierbar ist. Die unterschiedlichen Funktionsblöcke dieser Programmiersprache sind nach der IEC Norm entwickelt und können somit auf den unterschiedlichen Steuerungen gleich programmiert werden.

Das entwickelte Programm für den dekadischen Zähler wurde mit simplen Vergleichs-, Addier- und Schiebebausteinen umgesetzt. Das hat den Vorteil, dass diese Bausteine in jeder Entwicklungsumgebung standardmäßig unterstützt werden und von jeder Steuerung einfach implementierbar sind. Ein weiterer Vorteil besteht in der Taktbaustein unabhängigen Programmstruktur, die teilweise systembedingt notwendig ist. Wäre der Programmcode mit einem Zählbaustein umgesetzt worden, so könnten Abhängigkeiten des CPU Zyklus der Steuerungen nicht ausgeschlossen werden. Mit dem entwickelten Programmcode wird das Programm in Abhängigkeit des CPU Zyklus abgearbeitet.

Im folgenden wird näher auf den Programmcode und die dabei verwendeten Funktionsblöcke eingegangen. Hierbei wird auch auf die Funktionsweise betrachtet.

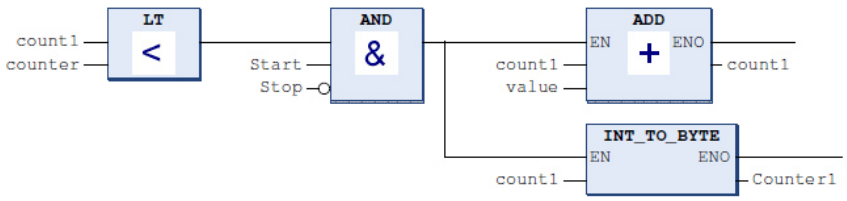


Abbildung 2.2: Programmcode 1. Zähldekade

Diese Dekade beginnt zunächst mit einem ’LT’ Block. Dies ist ein Boolescher Operator mit dem Ergebnis TRUE, wenn der erste Operand kleiner als der zweite ist. Hierbei wird der aktuelle Zählerstand mit der Zählbreite verglichen. Der zweite Block ist ein ’AND’ Block mit dem Ergebnis TRUE wenn das Startsignal anliegt und nicht das Ende des de-

kadischen Zählersystems erreicht ist. Wenn diese Ereignisse eintreten, sind der 'ADD' Block und ein Konvertierungsblock aktiviert. Dies wird durch die 'EN' – Enable Eingänge realisiert. Der 'ADD' Block ist ein Addierbaustein, der den aktuellen Zählerstand dieser Dekade hoch zählt. Es werden für den Addiervorgang und die Vergleichsoperationen jeweils Variablen des Datentyps Integer – 'INT' verwendet. Dies ist zwar nicht bei allen Systemen nötig, aber einige Systeme unterstützen nur Operationen mit INT Variablen. Aus diesem Grund müssen die Zählerstände zur Ausgabe auf die entsprechenden Buskoppler mithilfe eines Konvertierungsoperator in das Format Byte gewandelt werden. Die digitalen Ausgangskanäle der Buskoppler können genau ein 'BYTE' ausgeben. Aus diesem Grund muss für die Ausgabe des aktuellen Zählwertes auf den Buskopplern der Zählwert von 'INT' nach 'BYTE' konvertiert werden. Durch diese Konvertierung kommt es zu einem Informationsverlust von einem Byte, was in diesem Fall aber irrelevant ist, weil die Buskoppler maximal ein Byte in BCD darstellen können. Die Konvertierung wird mit dem Block 'INT_TO_BOOL' realisiert. An den Eingang des Bausteins wird die Variable vom Datentyp 'INT' deklariert und am Ausgang die Variable vom Datentyp 'BYTE'. In diesem Baustein wird durch einen Rechts- Schiebebefehl 'SHR' um 8 Stellen die INT Variable in eine BYTE Variable konvertiert. Diese BYTE Variable kann dann mit den Ausgangskanälen der Buskoppler verbunden werden. Hat der Zählerstand dieser ersten Dekade die Zählbreite erreicht, gibt es ein Überlaufsignal und die nächste Zähldekade wird um eins inkrementiert. In der Abbildung 2.3 wird der Programmcode für die folgenden Zähldekaden bis zu Dekade 5 dargestellt.

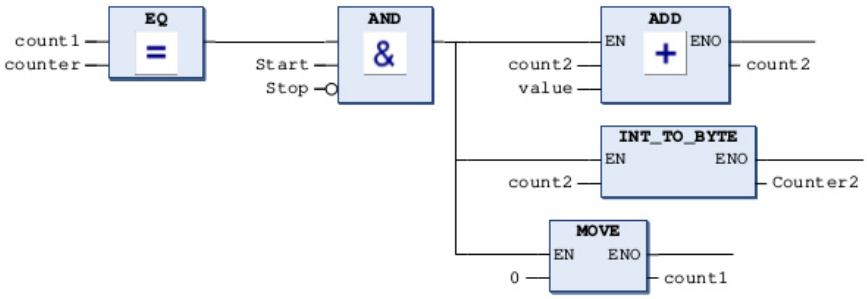


Abbildung 2.3: Programmcode 2. Zähldekade

Der Programmcode für die folgenden Dekaden unterscheidet sich zu der ersten Dekade durch die erste Eingangsbedingung. Es wird durch ein 'EQ' Block abgefragt, ob die Zählbreite der hervorgehenden Dekade entspricht. Der 'EQ' Block ist ein Boolescher Operator mit Ergebnis TRUE, wenn beide Operanden gleich sind. Wenn diese Bedingung erreicht ist, ist die Abarbeitung des Netzwerks synchron zur ersten Dekade. Eine weitere Ergänzung zum ersten Netzwerk ist der 'MOVE' Block. Er setzt die hervorgehende Dekade zurück. Der 'MOVE' Befehl ist ein Schiebebefehl, der eine Zuweisung einer Eingangsvariablen auf eine Ausgangsvariablen des selben Typs realisiert.
Die noch folgenden 3 Zähldekaden entsprechen dem Netzwerk in der Abbildung 2.3. Hat die fünfte Zähldekade die Zählbreite erreicht, wird das Stoppsignal gesetzt. Das

vollständige Programm ist in der Anlage A auf S.90 dargestellt.

Im folgenden Abschnitt werden Festlegungen und Bedingungen für die Controller- und Taskeinstellungen definiert. Diese Definitionen sind die Basis für die Vergleichbarkeit der Systeme untereinander.

Ein sehr wichtiger und entscheidender Faktor für den Vergleich der Performance der unterschiedlichen Systeme ist die Taskzeit bzw. Zykluszeit der Programmabarbeitung. Die Taskzeit in der SPS, ist die Zeit für die Abarbeitung eines Programm-Zyklus, einschließlich aller Kommunikationsaufgaben. Sie ist abhängig von der Rechenleistung des Controllers und der benötigten Rechenzeit für die Programmabarbeitung. Sie steht somit im kausalen Zusammenhang der eigentlichen Programmlänge bzw. der Anzahl der Anweisungen im Programm. [2, Vgl.] Auf den konfigurierten Systemen wurde das Programm einer freilaufenden Task zugeordnet. Im freilaufenden Modus beginnt jeder Taskzyklus, wenn der vorhergehende Zyklus abgeschlossen wurde und ein kurzer Zeitraum mit Systemverarbeitung stattgefunden hat. [3, Vgl. S.39] Dieser Fakt wird bei SIEMENS Systemen durch den Operationsbaustein 1 realisiert. In diesem OB wird dann der Funktionsblock mit dem FUP Programm aufgerufen. Der 'optimierte Bausteinaufruf' wird bei den Systemen zusätzlich aktiviert. Bei Systemen ohne freilaufende Task wird eine zyklische Task parametriert. Diese zyklische Task hat die minimale Zykluszeit des Controllers als Taskzykluszeit definiert.

Eine weitere Bedingung die auf allen Systemen umgesetzt wird, ist das die symbolische Programmierung angewendet wird. Bei der symbolischen Programmierung wird nicht der Operand direkt über die absolute Adressierung angesprochen, sondern stellvertretend ein Symbolname anstelle des Operanden eingegeben. In dem Programmcode für die Rechenoperationen wurden jeweils Integer-Variablen verwendet. Die Rechenoperationen des Controllers werden mit Wortoperationen durchgeführt. Für die Variablen, die auf den Ausgabemodule schreiben, wird der Type BYTE verwendet.

Die Aktualisierungszeiten für die dezentrale Peripherie des Systems, wird entsprechend der minimalen Aktualisierungs- und Updatezeiten der Slavemodule gesetzt. Die Kommunikationslast wird bei Systemen mit dieser Funktion auf 20% gesetzt. Diese Bedingung betrifft vor allem SIEMENS Systeme, da bei den modernen Systemen eine höhere Kommunikationslast DEFAULT-seitig eingestellt ist. Die Mindestzykluszeit wird bei allen Systemen, wo dies möglich ist, deaktiviert. Bei Systemen mit Profibus DP wird eine Übertragungsgeschwindigkeit von 1,5 Mbit/s verwendet.

Bei dem Hardwareseitigen Aufbau wurde für Systeme mit Patchkabel eine Länge von 0.2 bis 0.5 Meter verwendet. Bei Systemen mit Profibus wurde generell eine Länge von 0.5 Meter realisiert.

Die Festlegungen und Bedingungen für die Testumgebung sind auf jedem System um-

gesetzt und somit die Basis für die Vergleichbarkeit der unterschiedlichen Steuerungssysteme.

3 Technologie und Applikationen der Systeme

Die moderne Automatisierungstechnik hat sich aufgrund der neuen Möglichkeiten in der Halbleitertechnik gewandelt. So wird heute oftmals nicht nur die klassische Verwendung als Steuerung vorgesehen, sondern immer mehr treten auch kompakte SPS mit spezieller Aufgabenstellung in Erscheinung. Das Wachstum PC-basierter Steuerungen ist ungebrochen. Dank moderner Prozessoren versprechen diese eine hohe Performance. [4, Vgl.]

So auch in diesem Performancevergleich. Es werden Steuerungssysteme aus der modernen Automatisierungstechnik betrachtet sowie klassische Steuerungssysteme. Das Kapitel wird zunächst die unterschiedlichen Steuerungsarten und dann folgend die eingesetzten Bussysteme näher erläutern. Hierbei wird zunächst einmal kurz die Funktionsweise des Bussystems erläutert. Danach werden die Steuerungssysteme der unterschiedlichen Hersteller vorgestellt, die auf Basis dieses Feldbusses untersucht wurden.

Es wurden klassische Hard/ Kompakt-SPS, Panel-SPS und Mikro-SPS betrachtet. Die klassische SPS besteht aus einem Controller (CPU) für Programme, Daten und Kommunikation mit der Peripherie. Zum anderen hat dieser Controller Schnittstellen zum Programmiergerät und ein internes Bussystem. Im allgemeinen hat diese Zentralbaugruppe einen μP, RAM und EEPROM. Die Controller können eine MultiCore CPU oder eine SingleCore CPU besitzen. Die MultiCore CPUs haben mehrere Kerne und können somit die Daten- , Programm- und Kommunikationsverarbeitung auf verschiedene Kerne aufteilen. Die SingleCore CPUs verarbeiten die Daten- und Kommunikationsmechanismen in einem Prozessorkern. Diese unterschiedlichen Architekturen haben entscheidende Auswirkungen auf die Gesamtperformance des Steuerungssystems. Über einen Rückwandbus sind die Einsteckkarten mit der SPS verbunden. Der interne Bus ist für die Stromversorgung, Kommunikation und Verbindung zu den E/A Peripheriebaugruppen verantwortlich. Es gibt Einsteckkarten zur Feldbusankopplung oder klassische Eingangs- und Ausgangskanäle zur Kommunikation und Aktion mit Sensoren und Aktoren. Dieser Systemaufbau der Steuerung wird oftmals als klassische SPS beziehungsweise Hard-SPS bezeichnet und ist in vielen komplexen Steuerungsanlagen verbaut.

Neben leistungsfähigen Prozessoren und schnellen Steuerungsprogrammen geht es zunehmend auch um komfortable Bedienung, Visualisierung, Erfassung sowie Verwaltung größerer Datenmengen. Für die Visualisierungen ist die Erfassung und Verwaltung von großen Datenmengen ein wesentlicher Bestandteil. Diese müssen heutzu-

tage mit anderen Softwarewerkzeugen wie zum Beispiel MS SQL weiter verarbeitbar sein. Vorrausetzung hierfür ist eine weitere Zusammenführung der PC-/Office Welt mit der Automatisierungs- und Steuerungstechnik. Die Anforderung hierfür ist, nicht nur die E/A-Baugruppen aus dem Schaltschrank zu eliminieren sondern auch die CPU als kommunikationstechnisches Bindeglied zu befähigen. Dies setzen sogenannte Panel- und Micro SPS Systeme um. Sie basieren auf Microcontroller- Plattformen, sind echtzeitoptimiert und verfügen über Multitasking-Fähigkeiten. Diese Eigenschaften ermöglichen schnelle synchrone Feldbuszykluszeiten und minimale zeitliche Toleranzen (Jitter). Solche Steuerungssysteme bestehen aus folgenden Komponenten.

- Coprozessor mit eigenem Betriebssystem (zumeist multitaskingfähiges Echtzeitbetriebssystem) und dem Programmspeicher
- Busmaster-Prozessor für Verwaltung des Busbetriebs und seine Diagnose -> unverzichtbar, da ein IPC überwiegend keine E /A-Baugruppen hat
- Multiport-Memory RAM (MPM) als Zwischenlager und zentrale Verteilerstelle für alle Daten

Der Name Soft-SPS besagt, dass die SPS-Funktionalität allein durch Software gewährleistet wird. Ein gewisser Schwachpunkt hinsichtlich Industrietauglichkeit kann dabei das Betriebssystem des Rechners spielen, sodass klassische SPS mit speziellem Prozessor und Interpreterprogrammen ein weit größeres Einsatzgebiet bedienen können. [34, Vgl.] Es gibt von dieser Art von Steuerung eine Kombination aus Panel-PC und SPS-System, sogenannte Panel- SPS oder Embedded PC's mit integrierter Soft-SPS, sogenannte Micro- SPS. Diese Steuerungen basieren auf Multicore-Plattformen, die eine parallele Abarbeitung von rechenintensiven Steuerungs-, Visualisierungs-, Datenverarbeitungs- und Kommunikationsaufgaben ermöglichen. Auf diesen Steuerungssystemen läuft ein Embedded Betriebssystem wie beispielsweise Windows Embedded oder ähnliche Systeme der Linux Architektur. Die CPU-Priorität liegt dabei immer auf dem Steuerungsteil. Diese Steuerungen besitzt entweder eigene E/A oder bietet entsprechende Kopplungsmöglichkeiten. Obwohl der Umfang der zu verarbeitenden digitalen und analogen Signale nicht ganz an den leistungsstarken Hardware-SPS herangeht, lassen sich mit HMI-Steuerungen auch umfangreiche Applikationen realisieren. [5]

Um Prozesse zu überwachen, steuern und Informationen auszutauschen ist es wichtig, ein Netzwerk zwischen den technischen Geräten zu realisieren. Die Performance dieses Netzwerks ist abhängig von der Art der Übertragungsleitung, über die die Informationen gesendet werden, sowie der Arbeitsweise und Aufbau des Kommunikationstelegramms. Die Verwendung von Netzwerken und Bussystemen ist heutzutage kaum noch aus der Automatisierungsindustrie wegzudenken. Die Bussysteme vernetzen die Gerätschaften aus der Steuerungs- und Sensorebene. Des weiteren werden Sie zur Kommunikation zwischen den Automatisierungsrechnern, Bedienstationen und anderen Ebenen der Automatisierungstechnik eingesetzt. [6, Vgl. S.107]
Die Bussysteme sind das Verbindungsglied für die Informationsübertragung der Master-

systeme mit den Slavegeräten verantwortlich. An dieses Kommunikationsmedium werden besondere Anforderungen gestellt, wie

- hohe Übertragungsgeschwindigkeit
- Zuverlässigkeit
- Echtzeitfähigkeit

Die Echtzeitfähigkeit ist ein entscheidender Faktor und lässt die unterschiedlichen Bussysteme in verschiedene Echtzeitklassen einteilen. So werden die Feldbussysteme unterschiedlichen Anwendungsgebieten zugeordnet. Zu diesen Anwendungsgebieten gehören zum einem, digitale Regelsysteme oder Motion-Control-Anwendungen mit Zykluszeiten von unter einer Millisekunde, sowie die Förder- und Gebäudetechnik mit Zykluszeiten im zwei- bis drei-stelligen Millisekunden Bereich. Darauffolgend gibt es auch für diese speziellen Anwendungssysteme eine Unterteilung in Echtzeitklassen. Es gibt die harte Echtzeit, die die Signallaufzeiten in einem exakt vorgegebenen Zeitrahmen realisieren muss und es gibt die weiche Echtzeit. Bei dieser Art von Echtzeitklasse werden Abweichungen innerhalb einer gewissen Zeitspanne toleriert. [7, Vgl. S.4]

In dieser vorliegenden Arbeit wurden unterschiedliche Bussysteme, der Feldbus- und Ethernettechnologie betrachtet. Es wurde untersucht, inwiefern ein anderes Bussystem die Arbeitsweise und Performance des Gesamtsystems beeinflusst. Als Grundlage für das Testszenario dient der dekadischen Zähler. In dem Vergleich wurden folgende Bussysteme berücksichtigt:

- ProfiNET
- Modbus
- EtherCat
- SERCOS III
- POWERLINK
- Profibus

In den folgenden Unterkapiteln wird zunächst die Arbeitsweise und das Funktionsprinzip dieser Bussysteme kurz erläutert. Darauffolgend werden die im Vergleich verwendeten Systeme beschrieben, die über diesen Feldbus kommunizieren.

3.1 PROFINET- Kommunikation

PROFINET (“Process Field Network“) ist ein Ethernet basiertes Bussystem, dass als Nachfolger von PROFIBUS DP gesehen wird. Das Ethernet Bussystem ist durch Vollduplex Übertragung und 100Mbit/s Datenkommunikation charakterisiert. Dieses Bussystem ist für den Datenaustausch von I/O Controllern, Parametrierung und den Aufbau von Netzwerken in automatisierten Produktionsanlagen gedacht. [8, Vgl.] PROFINET

wurde von SIEMENS und den Mitgliedern der PROFIBUS Organisation entwickelt. In diesem Konzept gibt es aufgrund der hohen Anforderungen an die Übertragungsgeschwindigkeit verschiedene Ausprägungen. Zu diesen gehören PROFINET CBA und PROFINET IO. Diese Arten unterschieden sich in dem Prinzip des Datenaustausches. Für die unterschiedlichen zeitlichen Anforderungen gibt es PROFINET RT und PROFINET IRT. Profinet RT ist für weiche und keine Echtzeit und Profinet IRT für harte Echtzeit. In dieser Arbeit wurde PROFINET IO mit der Leistungsklasse RT verwendet. Dies ist eine zugeschnittene Lösung für die Kommunikation zwischen der Steuerung und der dezentralen Peripherie. Für die Implementierung der Gerätschaften in ProfiNet ist es nötig, die GSDML- Dateien einzupflegen. Diese auf XML- basierte Datei beschreibt die Eigenschaften und Funktionalität des entsprechenden Profinet-Geräts.
Die Adressierung in Profinet erfolgt mit der eindeutigen MAC- Adresse der Ethernet Geräte. In dem Profinet System wird ein Gerät durch einen symbolischen Namen eindeutig identifiziert.

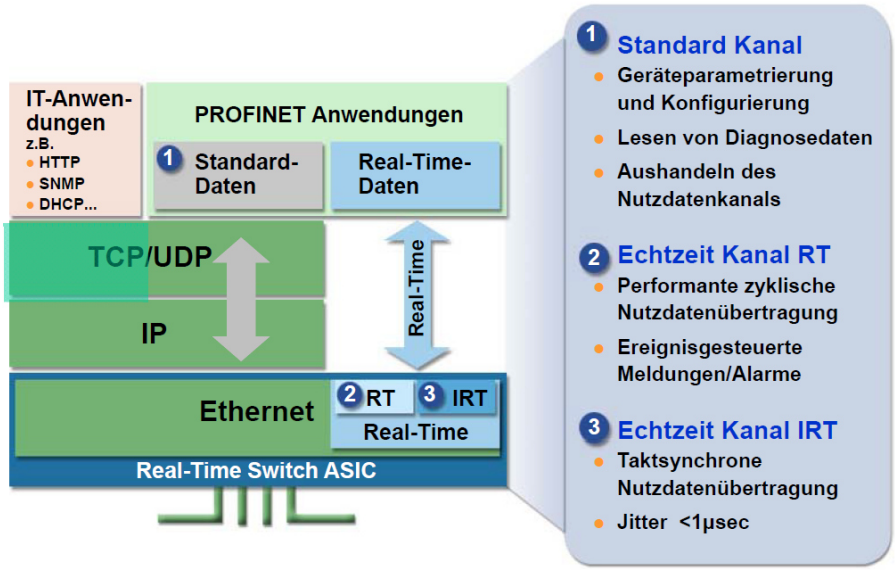


Abbildung 3.1: Profinet Telegrammaufbau mit den Leistungsklassen [10, S.74]

Wie in der Abbildung 3.1 dargestellt, verwendet Profinet als überlagertes Protokoll UDP/ IP. Diese Art von Datenaustausch beinhaltet die ungesicherte verbindungslose Broadcast-Kommunikation in Verbindung mit IP. [9, Vgl. S. 279] Die Art der Kommunikation wird je nach Leistungsklasse unterteilt. So werden Diagnose und Konfigurationsdaten über UDP/ IP und hochpriorisierte Nutzdaten direkt über das Ethernet- Protokoll versendet. Hierfür werden priorisierte Ethernet- Frames über VLAN verwendet. Durch diese Arbeitsweise kann für diese Leistungsklasse, Zykluszeiten von 1 ms realisiert werden. Für taktsynchrone Anwendungen mit, zum Beispiel, Motion Control Systemen dient Profinet IRT. Hierbei können auf Grundlage des Zeitmultiplexverfahrens Zykluszeiten von unter einer Millisekunde erreicht werden. [7, Vgl. S.7] Diese Art von Profinet wurde in dieser Arbeit nicht verwendet.

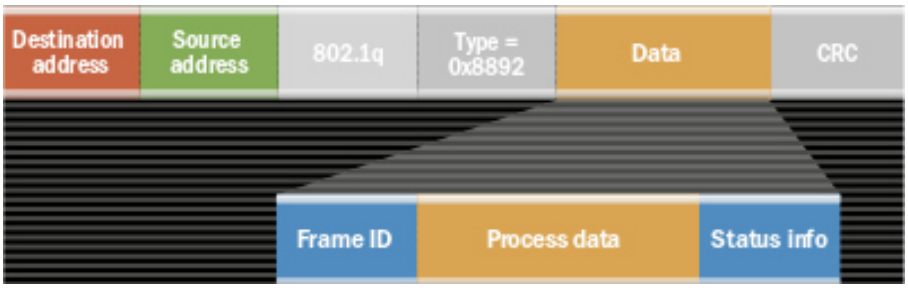


Abbildung 3.2: Profinet Ethernet Frame [7, S.7]

In der Abbildung 3.2 ist das Profinet RT Ethernet Frame dargestellt. Dieses Frame besteht aus dem Ethernet- Standard Frame, sowie einem Profinet- spezifischen Frame. Die kleinste Framegröße ist 64 Bytes. Dies bedeutet, das zwischen der Frame ID und der Cylic (Zyklusnummer) der Prozessdaten mindesten 40 Bytes übertragen werden müssen. Optional kann, wie in der Abbildung 3.1 zu sehen, auch die Frames über UDP/IP übertragen werden.
Im folgenden werden die im Testszenario verwendeten Systeme vorgestellt, die über Profinet kommunizieren.

3.1.1 SIEMENS Anwendungssysteme

System 1: S7-315F und dezentrale Peripherie ET200s:

Dieses System besteht aus einer CPU S7-315F PN/DP als Mastergerät und 5 ET200s als Slavegräte für die Ausgabe der Zählerstände. Das Mastergeräte ist somit eine Hard-SPS mit einem Rückwandbus. Das Gesamtsystem stellt ein homogenes System dar, da nur Bauelemente des Herstellers SIEMENS verwendet werden.

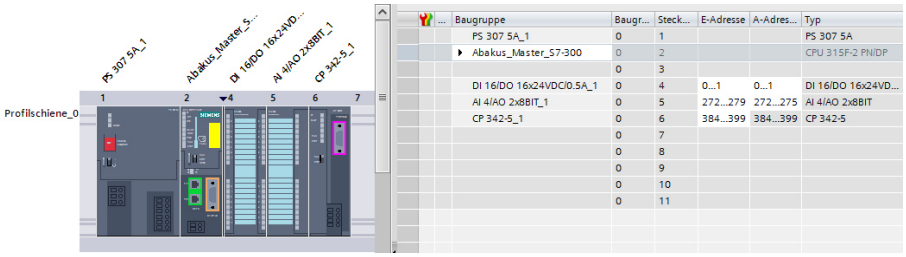


Abbildung 3.3: S7315F-2 PN/DP Rackaufbau

An Rückwandbus sind in diesem Systemaufbau ein digitales E/A Modul und ein analoges E/A Modul angeschlossen. Die dezentrale Peripherie ET200s besteht aus 1 Byte Eingangskanäle und 1 Byte Ausgangskanäle. Die Ausgangskarten im Rack der Steuerung besitzen je 2 Byte Nutzdaten für die Eingangs- und Ausgangskarten. Neben den

Eingangs- und Ausgangskarten sind noch 2 Module für die Spannungsversorgung verbaut.

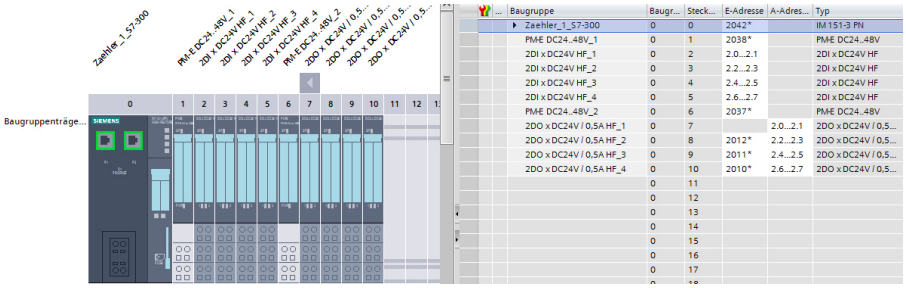


Abbildung 3.4: ET200s Rackaufbau

Die Zusammenschaltung dieser Komponenten ergibt folgenden Systemaufbau:

Abbildung 3.5: S7315F-2 PN/DP– ET200s Systemaufbau

Bei diesen Aufbau wurde Profinet RT in der Linientopologie als Bussystem verwendet. Die verwendete CPU hat einen mittleren Programmspeicher und eine Konfiguration zum Aufbau von fehlersicheren Automatisierungssystemen. Diese Konfiguration kann über die integrierte PROFINET Schnittstelle oder PROFIBUS Schnittstelle auf die Feldbuskomponenten zugreifen. Die sicherheitsgerichtete Kommunikation über das integrierte PROFIsafe Profil wurde nicht in diesem Performancevergleich verwendet. Die CPU verfügt über einen Mikroprozessor der Bearbeitungszeiten von 50ns bei Bitoperationen und 450ns bei Gleitpunktoperationen erreicht. In der Baugruppe ist ein 512kByte großer Arbeitsspeicher für sicherheitsgerichtete und Standard Programmteile verbaut. Die beiden integrierten Profinet Schnittstellen besitzen eine, 2 Port Switch, basierend auf TCP/IP. Dieser kann für die Inbetriebnahme und Diagnose über Step7 und dem TiaPortal verwendet werden oder für die S7 Kommunikation zwischen SIMATIC Steuerungen. Des weiteren kann für den Datenaustausch eine offene TCP/IP- Kommunikation über Profinet oder OPC für die Kommunikation mit anderen Controllern und Peripheriegeräten mit eigener CPU verwendet werden. [18, Vgl.]

Als dezentrale Peripherie wurde in diesem System fünf ET200s für die Ausgabe der binärcodierten Zählerstände verwendet. Diese bestehen aus einem Interfacemodul und 1 Byte Ausgangs- und Eingangskanäle. Das Interfacemodul IM 151-3PN ermöglicht die Anbindung der ET200s an Profinet und wickelt die Kommunikation zwischen den Modulen und dem übergeordneten S7-315F Controller selbstständig ab. An diesem Modul ist ein 2 Port Switch für den linienförmigen Topologieaufbau verbaut. Die Einsteckkarten

und das Interfacemodul sind von der Variante High Feature. Dies ermöglicht eine erweiterte Diagnosemöglichkeit an den Einsteckkarten, sowie am Interfacemodul und eine erhöhte Anlagenverfügbarkeit. Bei High Feature werden die Känale sauberer getrennt. Hierbei werden saubere Filter verbaut, die einen Flankenwechsel innerhalb von 0,02 ms erkennen. Die Module haben minimale Aktualisierungszeit von 1ms. Die Aktualisierungszeit ist abhängig von dem verwendeten Controller[1]. [19]

System 2: S7-1511 und dezentrale Peripherie ET200s:

Dieses System besteht aus einer CPU S7-1511- 1 PN als Mastergerät und 5 ET200s als Slavegeräte für die Ausgabe der Zählerstände. Das Mastergerät ist eine Hard- SPS mit einem Rückwandbus. Das Gesamtsystem stellt ein homogenes System dar, da nur Bauelemente des Herstellers SIEMENS verwendet werden.

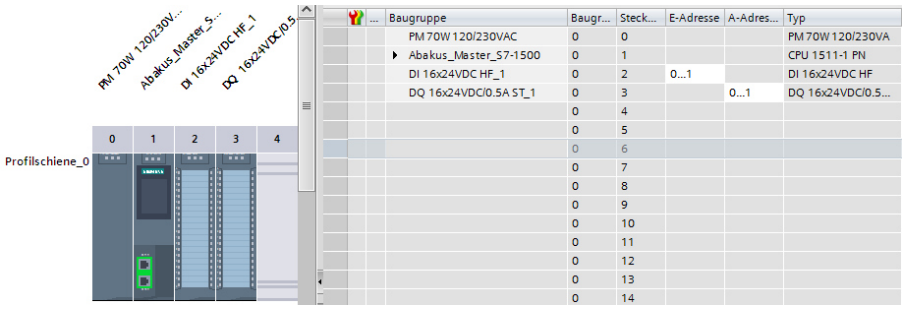


Abbildung 3.6: S7-1511 Rackaufbau

An dem Rückwandbus des Mastergeräts sind in diesem Aufbau ein digitales 16 Bit Eingangs- und 16 Bit Ausgangsmodul angeschlossen. Das Eingangsmodul ist von der Variante HighFeature und das Ausgangsmodul von der Variante Standard. Als dezentrale Peripherie wurde wie im System 1 die ET200s verwendet. Der Systemaufbau der ET200s Module entspricht der Konfiguration aus dem Systemaufbau 1. Eine Beschreibung ist im Absatz 1 vorzufinden. Der 2 Port Switch des Controller unterstützt Profinet RT und IRT. Durch die 2 Port Profinet Schnittstelle können Anlagen in der Linienstruktur verwendet werden. Das Gesamtsystem wird unter Profinet RT als Linien Topologie aufgebaut und ergibt folgendes Gesamtsystem:

Abbildung 3.7: S7-1511- 1 PN– ET200s Systemaufbau

Die CPU S7-1511 PN ist die preiswerte Einstiegs- CPU der neuen S7-1500 Generation des Herstellers SIEMENS. Diese CPU ist für einen Anwendungsbereich mit mittle-

[1] Systemaufbau ET200s

ren Anforderungen an Verarbeitungsleistung und Reaktionsgeschwindigkeit in der Fertigungstechnik geeignet. Die Reaktionszeiten werden bestimmt durch den 100 Mbit/s optimierten Rückwandbus. Das Mastergerät verfügt über einen leistungsstarken Prozessor mit Befehlszeiten von 60ns für Bitoperationen und 72ns für Wortoperationen. Des weiteren gibt es einen umfangreichen Arbeitsspeicher von 150 kByte für Programm und 1 Mbyte für Daten. Auf dem Controller gibt es ein Display zur Anzeige von allgemeinen Statusinformationen . Über dieses Display können Basiseinstellungen vorgenommen werden, sowie Adressen der CPU und angeschlossener Ethernet- Kommunikationsprozessoren. [20] Die Systemdiagnose kann über das Display oder über das TiaPortal vorgenommen werden. Für die Diagnose, Programmierung und Inbetriebnahme wird das TiaPortal verwendet.

System 3: S7-1516 und dezentrale Peripherie ET200s:

Dieses System besteht aus einer CPU S7-1516- 3 PN/DP als Mastergerät und 5 ET200s als Slavegeräte für die Ausgabe der Zählerstände. Das Gesamtsystem stellt ein homogenes System dar, da nur Bauelemente des Herstellers SIEMENS verwendet werden.

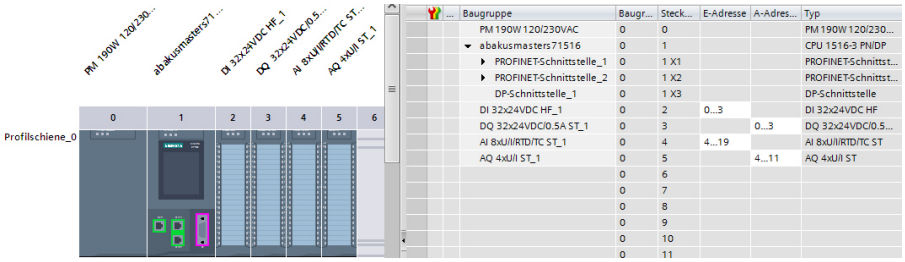


Abbildung 3.8: S7-1516- 3 PN/DP Rackaufbau

An dem Rückwandbus des Mastergeräts sind in diesem Aufbau ein digitales 16 Bit Eingangs- und 16 Bit Ausgangsmodul, sowie ein analoges Eingangs- und Ausgangsmodul angeschlossen. Als dezentrale Peripherie wurde wie im System 1 die ET200s verwendet. Der Systemaufbau der ET200s Module entspricht dem der aus dem Systemaufbau 1. Eine Beschreibung ist im Absatz 1 vorzufinden. Es ergibt sich folgender Gesamtaufbau:

Abbildung 3.9: S7-1516- 3 PN/DP– ET200s Systemaufbau

Die CPU S7-1516 -3 PN/DP ist eine Zentralbaugruppe mit großen Programm- und Datenspeicher. Die Baugruppe ist geeignet für Applikationen mit hohen Anforderungen an Programmumfang und Vernetzung. Die Steuerung dieses Testaufbaus entspricht der

selben SIEMENS Generation wie Testaufbau 2. Dieses System ist leistungsstärker als das System 2.
Der leistungsfähigere Prozessor erreicht Befehlslaufzeiten von 10ns für Bitoperationen und 12ns für Wortoperationen. Daraus folgt eine wesentlich höhere Verarbeitungsgeschwindigkeiten in Binär- und Gleitpunktarithmetik gegenüber dem System 2. Neben der leistungsstarken CPU besitzt diese Steuerung einen umfangreichen Arbeitsspeicher von 1 Mbyte, reserviert für Programm, und 5 Mbyte, reserviert für Daten. [21] Neben den 2 Port Switch verfügt dieses System noch über eine zusätzliche Profinet Schnittstelle mit separater IP- Adresse und einer Profibus- DP Master Schnittstelle. Dies kann zum Beispiel für eine Netzwerktrennung genutzt werden. Auf dem Controller gibt es ein größeres Display, als bei dem Controller S7-1511, zur Anzeige von Übersichtsinformationen. Das System kann nur mit dem TiaPortal für die Implementiert werden.

System 4: S7-315F und dezentrale Peripherie WAGO 750-370:
In diesem Aufbau wird ein inhomogenes System getestet. Dieses System besteht aus dem Controller aus dem System 1 und fünf Buskoppler 750-370 der Firma WAGO. Der Feldbuskoppler verfügt über einen integrierten zwei Port Switch und ermöglicht somit den einfachen Aufbau einer Linienstruktur ohne zusätzliche Infrastrukturelemente. Nach dem zusammenschalten der Komponenten ergibt dies folgenden Gesamtaufbau.

Abbildung 3.10: S7315F-2 PN/DP– WAGO 750-370 Systemaufbau

Durch dieses System soll im Vergleich zum System 1 betrachtet werden, inwiefern das Zählverhalten durch die anderen Buskoppler beeinflusst wird. Die Buskoppler WAGO 750-370 bestehen jeweils aus dem Interfacemodul, zwei 4 Bit Ausgangskanäle und zwei 1 Byte Eingangkanäle. Des weiteren muss für den internen Bus des Kopplers eine Abschlussklemme verbaut werden.

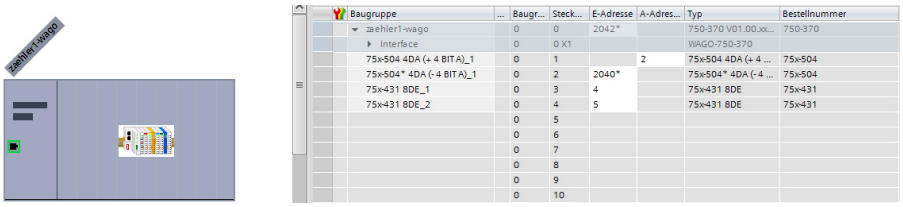


Abbildung 3.11: WAGO 750-370 Rackaufbau

Der Buskoppler ermittelt bei der Initialisierung den physischen Aufbau des Knotens. Dadurch erkennt der Buskoppler die gesteckten E/A Karten und erstellt daraus ein lokales Prozessabbild der Ein- und Ausgänge. [22] Die Aktualisierungszeiten für den Feldbuskoppler werden vom Hersteller auf 4ms gesetzt. Untersuchungen aus anderen Projek-

ten der Fakultät haben ergeben, das diese je nach Netzlast 8ms bis 16ms betragen. In diesem Performancevergleich wurde eine Aktualisierungszeit von 16ms verwendet. Die Konfiguration wird über GSDML Dateien für den Betrieb und die Diagnose wird in dieser Arbeit über das TiaPortal realisiert. Die Konfigurations- und Parameterdaten werden vom Mastersystem nichtflüchtig in den Koppler geschrieben.

System 5: S7-317 und inhomogene dezentrale Peripherie:
Dieser Systemaufbau stellt wiederum ein inhomogenes Feld dar. Als Steuerung wurde eine CPU 317-2 PN/DP verwendet. Die dezentrale Peripherie wurde mit Buskoppler der Firma PhoenixContact und der Firma Wago installiert.

- 2x PhoenixContact ILB PN 24
- 1x PhoenixContact FL IL 24
- 2x WAGO 750-340

Daraus folgend das die Buskoppler WAGO 750-340 einen 1x Rj45 Port besitzen, wurde bei diesem Aufbau eine sternförmige Topologie realisiert. Der Gesamtaufbau hat somit folgende Struktur:

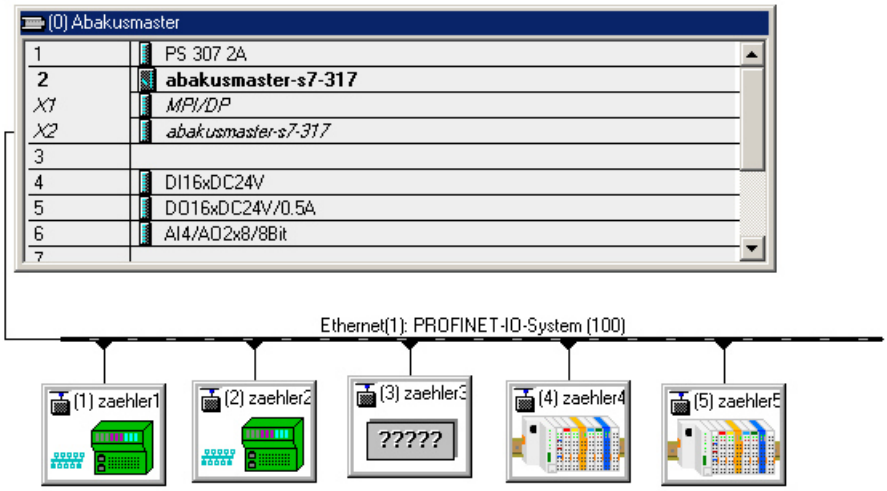


Abbildung 3.12: S7- 317T– inhomogener Aufbau

Der in diesem System verwendete Controller ist eine Siemens CPU der Baureihe S7 mit einem 512 kByte großen Programmspeicher. Sie ist für anspruchsvolle und komplexe Steuerungsanwendungen geeignet. Diese CPU kommt in Anlagen zur Anwendung, wo neben zentraler Peripherie auch dezentrale Automatisierungsstrukturen enthalten sind. Der Prozessor dieses Controllers erreicht eine Bearbeitungszeit von 0.05ms für 1000 Anweisungen von Wortoperationen. Dieser Controller ist somit insbesondere bei Wort- und Doppelwortbefehlen und bei 32 Bit- Festpunktbefehlen bezüglich der Bearbeitungszeit immens leistungsstark. Diese Steuerungen bilden eine ideale Plattform für einfache softwarerealisierte technologische Automatisierungsaufgaben.

Der Inline Block der Firma PhoenixContact besitzt 16 digitale Eingänge und 16 kombinierte Ein- oder Ausgänge. Dieser Buskoppler wurden für die erste und zweite Zähldekade verwendet. Die minimale Aktualisierungszeit dieser Komponenten beträgt 1ms. Die dritte Zähldekade wird bei diesem System auch durch ein Inline Buskoppler der Firma Phoenix Contact realisiert. Dieser Koppler kann mit bis zu 63 weiteren Inline Modulen erweitert werden. Der eingesetzte Koppler wurde mit einem 16Bit Ausgabemodul (IB IL 24 DO 16) bestückt. Bei diesem Modul kann wiederum eine minimale Aktualisierungszeit von 1ms verwendet werden. Das Inline Modul hat nur einen Port für die Profinet Kommunikation und muss somit über einen Switch in das System integriert werden. Als Switch wurde ein SIEMENS SCALANCE X208 verwendet.
Die vierte und fünfte Zähldekade wird über den Buskoppler WAGO 750- 340 realisiert. Diese Module haben nur einen Busanschluss für Profinet und werden auch über den Switch in das Netzwerk integriert. Die minimale Aktualisierungszeit der Wago-Module beträgt laut Herstellerangaben 4ms. In dem Gesamtsystem wurde aber auf Grund der selben Untersuchungen wie bei dem Wagomodul 750-370, die Aktualisierungszeit auf 16ms gesetzt. Dieses System wurde mithilfe von Step7 V5.5 programmiert.

3.1.2 TURCK Anwendungssystem

System 6: VT250 und homogene dezentrale Peripherie:
Dieser Systemaufbau besteht aus einer Steuerung VT250 und fünf verschiedenen Buskopplern der Firma Turck. Die in dem Systemaufbau verwendeten Buskoppler sind folgende:

- 2x BL20 Multiprotokoll Gateway mit 1 Byte Eingangs- und Ausgangskarte
- 1x FEN20 Multiprotokoll I/O Modul mit 16 universellen Kanälen
- 2x TBEN-L1 Multiprotokoll I/O Modul mit 16 universellen Kanälen und ein Modul mit 1 Byte Eingangs- und Ausgangskanal

Das VT250 SPS ist eine sogenannte Panel- SPS. Diese Steuerung besitzt ein HMI Panel mit 5,7“ TFT- Display und Touch- Screen. Sie kann sowohl als reines Bedienterminal und auch als Steuerung eingesetzt werden. Der Variablenaustausch zwischen den Panel- und der Soft- SPS erfolgt über ein im System integriertes Variableninterface. [24] Die zwei RealTime Ports der VT250 Steuerung ermöglichen, eine Linientopologie aufzubauen. Als Chipsatz wurde der netX Chip von Hilscher verwendet. Die auf netX-Controller basierte Automatisierungslösung kombiniert die Softsteuerung CODESYS mit Feldbus oder RealTime- Ethernet- Kommunikation. Die Controller der netX Reihe der Firma Hilscher bieten hierfür ausreichende Leistungsreserven. So können neben dem Protokollstack für die Ethernet- Kommunikation auch die SPS Funktionalitäten parallel abgearbeitet werden. [23] Die Netzwerkcontrollerfamilie netX ist eine intelligente Lösung, die alle bewährten Feldbus- und Echtzeit- Ethernet- Systeme auf kleinstem Raum verwirklichen kann. Auf diesem Chip ist das Laufsystem Codesys SP

bereits implementiert. Der Controller kann somit die Kommunikation von allen Feldbussen und RealTime- Ethernet- Systemen auf einen einzigen Chip realisieren und dabei als Master oder Slave fungieren. Aus diesem Grund kann das Turck System beispielsweise gleichzeitig als Profibus Master und Profinet Slave, oder mit einem anderen Feldbus- und Ethernet Protokoll kommunizieren. Diese Steuerung unterstützt die Protokolle Profibus-DP, CANopen, DeviceNET, Profinet, ErherNET/IP ModbusTCP und Modbus RTU. [25,26] Der in der VT250-Steuerung verwendete netX-Prozessor verwendet als Echtzeit-Betriebssystem “rcX“ der Firma Hilscher. Die 32 Bit RISC Single Chip CPU mit 200 MHz sichert nicht nur die Kommunikation, sondern bietet auch die nötigen Hardware Ressourcen für die Visualisierungs- und Steuerungssoftware. Für die Visualisierung verwendet Turck die QVIS Software von HILSCHER. Diese Umgebung ist auf die Hardware zugeschnitten und kann die Ressourcen des Steuerungssystems optimal ausschöpfen. Das integrierte Variablen-Interface arbeitet systemübergreifend mit der Bedienoberfläche und dem CODESYS SP V3 Laufsystem zusammen. Auf den VT250 setzt TURCK für die Programmierung das Hardware- unabhängige System CODESYS v3 ein. [27]

Die eingesetzten Buskoppler der Firma TURCK sind alle multiprotokollfähig. Hierunter ist zu verstehen, dass die verwendeten Feldbusgateways und I/O Module die drei Ethernet Protokolle ModbusTCP, PROFINET und EtherNET/IP unterstützen. Somit unterstützen die Protokollstacks dieser Module alle auf dem Markt relevanten Protokolle. Sie lassen sich automatisch in jedem der drei Protokolle betreiben. Möglich wird diese Multiprotokoll Funktionalität durch eine intelligente Identifikationsroutine der Gateways. Hierbei ermitteln die Module während der sogenannten Snooping Phase durch mithören des Traffics welches Ethernet Protokoll verwendet wird. Die Geräte wechseln dann automatisch in dem erkannten Protokoll und ignorieren die anderen Telegramme. Alle Geräte können mittels interner Switches in Linientopologie installiert werden und unterstützen Funktionen wie Priorisierter Hochlauf, sowie Features wie Topologieerkennung und Adresszuweisung. Die Module verfüngen für die Parametrierung und Inbetriebnahme über einen Webserver. [24]
In dem Systemaufbau wurden für die Zähldekade 1 und 2 das Modell BL20-E-GW-EN verwendet. Dieses BL20 Gateway stellt den Kopf der BL20 Station dar, der mit Steckkarten für 1 Byte Eingangskanäle und 1 Byte Ausgangskanäle erweitert wurde. Sämtliche Module der BL20 Station kommunizieren über einen internen Modbus. Die Daten der Module werden über das Gateway zum Feldbus weitergeleitet. [24]
In der dritten Zähldekade wurde das Modell FEN20 verwendet. Dies ist wiederum ein Multiprotokoll Slave für Profinet, EtherNET/IP und Modbus TCP. Dieses Modul verfügt über 16 universell als digitale Kanäle nutzbare Ein- oder Ausgänge. Die I/O Signale der Kanäle verteilen sich auf drei unabhängige, voneinander getrennte Potenzialgruppen. Die Adressen der Module können manuell über die am Modul befindlichen Drehschalter eingestellt oder automatisch über DHCP vom Master aus vergeben werden. [24]
Die Module der vierten und fünften Zähldekade werden durch die Module TBEN-L1 angesteuert. Diese beiden E/A-Module unterscheiden sich in der Bandbreite der Aus- und

Eingänge. So besitzt das eine Modul 8 Eingangs-/ 8 Ausgangskanäle und das andere 16 universal nutzbare Kanäle. Sie sind in galvanisch getrennten Potenzialgruppen, in abschaltbare und nicht abschaltbare I/O Kanäle, eingeteilt. Diese multiprotokollfähigen Slave-Module verfügen über pnp Eingänge bzw. pnp Ausgänge bis max. 2A. Das Gehäuse der Geräte ist glasfaserverstärkt, schock- und schwingungsgeprüft sowie voll vergossen. Aufgrund der Schutzart von IP65/ IP67 sind diese Geräte für ein robustes Industrieumfeld geeignet. [24, Vgl.]

3.1.3 PhoenixContact Anwendungssysteme

System 7: ILC350 und inhomogene dezentrale Peripherie ET200s:
Dieser Systemaufbau realisiert ein inhomogenen Feld, bestehend aus einer Phoenix-Contact ILC350 Steuerung als Profinetmaster und fünf ET200s als dezentrale Peripherie für die Zählerstandausgabe. Dieses System wurde als Linientopologie aufgebaut. Als Entwicklungsumgebung wurde PcWorx verwendet.

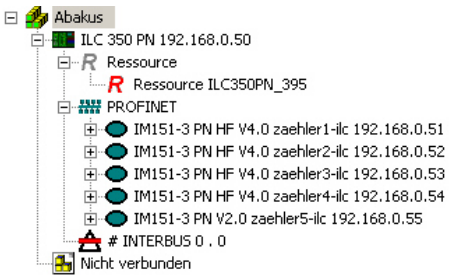


Abbildung 3.13: Inline ILC 350PN– inhomogener Aufbau

Das Mastersystem ist ein modularer, leistungsfähiger Profinet Controller vom Typ ILC 350 PN. Mit dieser Steuerung können mittlere bis anspruchsvolle Automatisierungsaufgaben realisiert werden. Alle Informationen sind auf einer Speicherkarte hinterlegt. Durch den modularen Aufbau der Steuerung kann diese einfach durch Module auf die jeweiligen Bedürfnisse angepasst werden. Der Controller hat 12 direkte Eingänge und 4 direkte Ausgänge. Über die integrierte Profinet Schnittstelle lässt sich der Controller über die Entwicklungsumgebung PcWorx nach IEC Norm parametrieren und programmieren. Der Controller unterstützt unter Profinet eine Aktualisierungszeit der Profinet Teilnehmer von 1ms. Neben der Profinet Schnittstelle verfügt die SPS über eine Interbusmaster G4 Schnittstelle. Der Programmspeicher dieser SPS beträgt 1 MByte und 2 MByte für Datenspeicher. Die Bearbeitungsgeschwindigkeit für Bitanweisungen beträgt 9 μs und für Mix- Anweisungen 0,5 ms pro 1000 Anweisungen. Die minimale Taskzykluszeit des Controllers beträgt 1ms. [28] Als dezentrale Peripherie wurde die ET200s E/A Baugruppe verwendet. Der komplette Systemaufbau mit dem ET200s Modulen entspricht der Konfiguration aus dem Systemaufbau 1. Eine Beschreibung ist im Absatz 1 vorzufinden.

System 8: AXC1050 und inhomogene dezentrale Peripherie ET200s:
Dieses System besteht aus dem Axioline Controller AXC 1050 und als dezentrale Peripherie für die Zählerstandausgabe die ET200s- Buskoppler. Der Systemaufbau ist somit ein inhomogenes System. Die Geräte können aufgrund der verbauten Switche als Linientopologie aufgebaut werden. In der Entwicklungsumgebung ergibt dies folgenden Systemaufbau:

Abbildung 3.14: Axioline Profinet– inhomogener Aufbau

Die Steuerung AXC 1050 der Produktfamile AxioControl eignet sich für kleine bis mittlere Steuerungsapplikationen. Eine erweiterte Temperaturspezifikation sämtlicher in der AxioControl SPS verbauten Bauteile, sowie eine schützende Lackierung der Leiterplatte sorgen dafür, dass die AXC 1050 auch unter rauen Umweltbedingungen störungsfrei arbeitet. Die AXC 1050 unterstützt Standard Protokolle wie ModbusTCP und ProfiNet, sodass dezentrale Stationen und Geräte anderen Hersteller angekoppelt werden können. Zum Aufbau lokaler Stationen lassen sich die I/O-Module des Axioline-I/O-Systems direkt über den integrierten Axiobus an die Steuerung anreihen. [29] Die Aktualisierungszeit des Controllers beträgt für Profinet 4ms. Die Bearbeitungsgeschwindigkeit für Bitanweisungen beträgt 90μs und für Mixanweisungen 1,3ms pro 1000 Anweisungen. Die Steuerung hat einen Programmspeicher von 1 MByte und einen Datenspeicher von 2 MByte. Die minimale Taskzykluszeit des Controllers beträgt 1ms. [28] Als dezentrale Peripherie wurden, wie im System 1, die ET200s Baugruppen verwendet. Der Systemaufbau der ET200s Module entspricht der Konfiguration aus dem Systemaufbau 1. Eine Beschreibung ist im Absatz 1 vorzufinden.

System 9: AXC1050 und homogene dezentrale Peripherie:
In diesem Systemaufbau wurde die AXC 1050 Steuerung als Mastergerät verwendet und als dezentrale Peripherie die Axioline Buskoppler “AXL F BK PN“. Es wird somit hier ein homogenes System aus der PhoenixContact Produktreihe Axioline abgebildet. Das System wurde als Linientopologie aufgebaut und hat folgenden Gesamtaufbau:

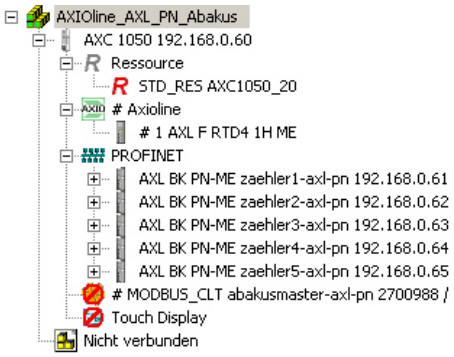


Abbildung 3.15: Axioline Profinet– homogener Aufbau

Die PhoenixContact Axioline F-Buskoppler stellen das Bindeglied zwischen dem Profinet Netzwerk und dem Axioline F-System dar. Der Buskoppler besteht aus einem Controller und einem Ein-/ Ausgabemodul. Diese beiden Elemente werden durch einen Lokalbus miteinander verbunden. Die Buskoppler sind über einen Lokalbus mit den Ein-/ Ausgabemodul verbunden. Die typische Zykluszeit des Axioline F-Lokalbusses sind ca. 10 μs. Die Laufzeit im Buskoppler sind vernachlässigbar (gegen 0 μs). [55, 56, Vgl.] Der Buskoppler hat 2 Profinet Ports und es können bis zu 63 Axioline F-Teilnehmer angeschlossen werden. Das eingesetzte I/O-Modul AXL “F DI8/1 DO8/1“ besteht aus 8 digitalen Eingangs und 8 digitalen Ausgangskanälen.

3.2 MODBUS- Kommunikation

Modbus/TCP wurde bereits 1979 von der Firma Modicon für die Automatisierungstechnik entwickelt. Modbus gibt es in unterschiedlichen Varianten und wurde ursprünglich für eine Kommunikation über serielle Verbindungen (RS-232, RS-485) entwickelt. Bei der Modbus Kommunikation wird unterschieden zwischen Modbus- Plus (High-Speed-Kommunikation über ein Token-Passing-Netzwerk) und Modbus- RTU (asynchrone Übertragung über RS-232 oder RS-485). Die Übertragung der Daten kann in Modbus- ASCII, Modbus-RTU oder ModbusTCP erfolgen. Bei der TCP/IP Variante wurden Modbus-Dienste und das Objektmodell der klassischen Varianten unverändert beibehalten und auf TCP/IP als Übertragungsmedium abgebildet. So haben alle Varianten ein einheitliches Anwendungsprotokoll, das ein universelles Telegramm für automatsierungstechnische Daten und Kommunikationsdienste darstellt. In diesem Vergleich wurde ausschließlich Modbus TCP/UDP betrachtet.
Das ModbusTCP Bussystem zählt mit zu den ältesten Industrial Ethernet Bussen und arbeitet nach den Client/ Server Prinzip. Es ist durch eine schnelle, flexible und effektive Kommunikation in industriellen Netzen gekennzeichnet.
ModbusTCP ist das einzige industrielle Kommunikationsprotokoll, das über einen eigenen sogenannten „Well known port (Port 502)“ verfügt, der es im Internet routingfähig macht. Für die Übertragung des Modbus Protokolls wird für die Ethernet basierte

Datenübertagung das TCP Protokoll verwendet. Die Parameter und Daten des Modbustelegramms werden dabei nach dem Einbettungsprinzip in die oberen Ebenen (Anwendungsschicht des OSI- Referenzmodells) der TCP/ IP Telegramms integriert. Beim Vorgang der Einbettung erzeugt der Client einen “Modbus Application Header“ (MBAP), der dem Server die eindeutige Interpretation der empfangenen Modbus-Parameter und Befehle ermöglicht. Grundsätzlich darf in einem TCP/IP-Telegramm nur ein Modbus-Anwendungstelegramm eingebettet werden. [11, Vgl.]

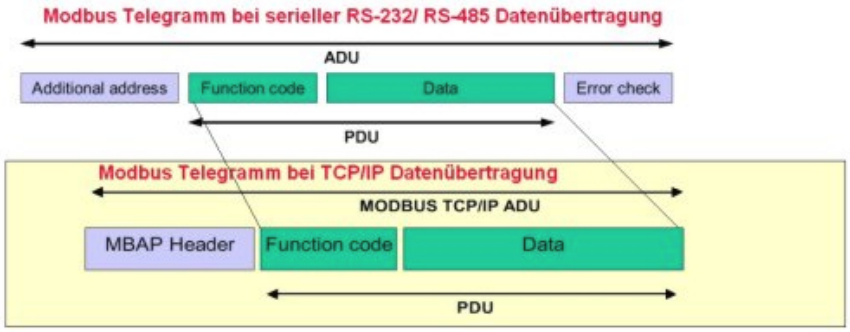


Abbildung 3.16: Modbus Telegramm [11]

Die Abbildung 3.16 zeigt, wie das Modbus-Protokoll der seriellen Datenübertragung (Standardprotokoll) in einem TCP/IP-Frame eingebettet ist. Bei dem Standardprotokoll werden neben den Befehlscodes auch die Slave Adressen, Prüfsummen und Daten übertragen. Die Modbus-Kommunikation über TCP/IP nutzt hierfür die unteren Schichten des TCP/IP Protokolls. Die Datenübertragung zwischen dem Modbus Client und dem Modbus Master wird vom Client aus gesteuert. So sendet der Client eine Request Anfrage in Form von Modbus- Funktionscodes an den Server. Nach diesem Dienstaufruf antwortet der Server, nach der erfolgreichen Requestanfrage, mit dem Response Telegramm. Request- und Response-Telegramm enthalten Parameter, Informationen, Fehlermeldungen und Daten, die übertragen werden sollen. Desweiteren stellt das Modbusprotokoll verschiedene Register und Funktionscodes für den lesenden und schreibenden Zugriff auf die Kanäle der Peripheriebaugruppen zur Verfügung. [11, Vgl.]
Im folgenden werden die im Testszenario verwendeten Systeme vorgestellt, die über ModbusTCP miteinander kommunizieren.

3.2.1 SchneiderElectric Anwendungssystem

System 10: Modicon TSX Premium und inhomogene dezentrale Peripherie:
Dieser Systemaufbau besteht aus der Modicon TSX Premium P56 und einer inhomogenen Peripherie. Die Peripherie besteht aus:

- 2x BL20 Multiprotokoll Gateway mit 1 Byte Eingangs- und Ausgangskarte
- 1x FEN20 Multiprotokoll I/O Modul mit 16 universellen Kanälen

- 2x TSX MOMENTUM

Die modulare Steuerung ist für schnelle Prozesse optimiert. Sie eignet sich aufgrund der enormen Leistungsfähigkeit bei boolischen und arithmetischen Operationen sehr gut für den Einsatz in komplexen Infrastrukturen. Der Modicon Prozessor ist eine High Performence CPU mit 37 ns Bearbeitungszeit pro Anweisung und 1024 kB internen RAM für Programm und Daten. Die Ausführungszeit pro Befehl beträgt bei Wortoperationen 45ns bis 60ns. Des weiteren hat die Steuerung eine Multitasking Struktur für Anwendungen mit Echtzeitverarbeitungen. Das Programm wurde auf der Modicon mithilfe einer Fast Task umgesetzt. Die Steuerung besitzt nur ein Uplink-Port. Aus diesem Grund wurden die MOMENTUM-Buskoppler mithilfe eines Switches an eine Sterntopologie angeschlossen. In diesem Systemaufbau werden zum einen Buskoppler der Firma TURCK verwendet und zum anderen Buskoppler der SchneiderElectric MOMENTUM Serie. Eine Systembeschreibung der oben aufgeführten TURCK Controller ist im Absatz der Systembeschreibung zu System 6 vorzufinden. Der Kommunikationsadapter MOMENTUM wird über Modbus TCP/IP konfiguriert. Dieser Controller verfügt über einen RJ-45 10Base-T Anschluss und muss direkt an den Switch des Systemaufbaus angeschlossen werden. Die Adresszuweisung wird über den BOOTP- Server konfiguriert. Der Controller besitzt 3 Register für den Kommunikationsadapterzugang. Dazu zählen das Datenregister, Konfigurationsregister und das Statusregister. Die Aktualisierungsrate für die Buskoppler beträgt bei diesem Systemaufbau 10ms. Für die Parametrierung, Inbetriebnahme, Programmierung und Diagnose wurde die Entwicklungsumgebung Unity PRO verwendet.

3.2.2 PhoenixContact Anwendungssystem

System 11: AXC 1050 und homogene dezentrale Peripherie:
Dieser Systemaufbau besteht aus dem Axioline Controller AXC 1050 und den Axioline F- Buskoppler “AXL F BK ETH“. Der homogene Systemaufbau konnte aufgrund der durchgehend integrierten dual Port Switch als Linientopologie vernetzt werden.

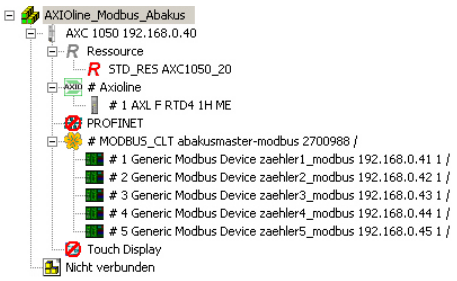


Abbildung 3.17: Axioline Modbus– homogener Aufbau

Das Mastergerät entspricht der Systembeschreibung aus System 8 und 9. In diesem Systemaufbau wurde die Kommunikation über Modbus-TCP realisiert. Die verwende-

ten Axioline Buskoppler binden über den Lokalbus die Ein- und Ausgabemodule an das Kommunikationsnetz. Als Ein- und Ausgabemodul wird das Modul “AXL F DI8/1 DO8/1“ verwendet. Der Buskoppler hat eine minimale Aktualisierungszeit von 4ms. Für die Adressierung der Module im ModbusTCP Netzwerk wurden die Drehcodierschalter am Gerät verwendet. Der Buskoppler unterstützt sowohl einen ModbusTCP- als auch einen ModbusUDP Server. Die Prozessdaten, der an dem Buskoppler angeschlossenen Geräten, werden auf fest bestimmten Registerbereichen abgebildet. Hierbei wird nicht zwischen allgemeinen Modbus Register und Modbus Input/Output Register unterschieden. Die Registerzuordnung erfolgt somit bei allen Funktionscodes gleich. Jedem Modul wird entsprechend der Datenbreite eine Anzahl an Registern zugeordnet. Dabei umfasst jedes Register 16 Bit. Es erfolgt keine Unterscheidung zwischen Digital- und Analog Modulen. [30, Vgl.] Für die Parametrierung, Inbetriebnahme, Programmierung und Diagnose wurde die Entwicklungsumgebung PcWorx verwendet.

3.2.3 TURCK Anwendungssystem

System 12: VT250 und homogene dezentrale Peripherie:
Dieser Systemaufbau entspricht dem Aufbau wie System 6. Der homogene Aufbau besteht aus den Komponenten:

- TURCK VT250
- 2x BL20 Multiprotokoll Gateway mit 1 Byte Eingangs- und Ausgangskarte
- 1x FEN20 Multiprotokoll I/O Modul mit 16 universellen Kanälen
- 2x TBEN-L1 Multiprotokoll I/O Modul mit 16 universellen Kanälen und ein Modul mit 1 Byte Eingangs- und Ausgangskanal

Aufgrund der Multiprotokollfähigkeit der verbauten Hilscher netX Chips konnte dieser Systemaufbau unter ModbusTCP realisiert werden.

3.3 EtherCAT- Kommunikation

EtherCAT ist ein von der Firma BECKHOFF entwickeltes Echtzeit Ethernet. Es ist in der Automatisierungstechnik für harte und weiche Echtzeitanforderungen geeignet. EtherCAT ist durch folgende Eigenschaften gekennzeichnet:

- kurze Zykluszeiten ($\leq$100μs)
- geringen Jitter für exakte Synchronisation ($\leq$1μs)
- niedrige Hardwarekosten [50, Vgl.]

Durch die optimale Nutzung der Ethernet Bandbreite können auch kleine Datenmengen mit extrem kurzen Zykluszeiten und einer hohen Übertragungsleistung übertragen wer-

den. [44, Vgl.]
Bei der EtherCAT Busstruktur durchläuft ein EtherCAT Telegramm alle angeschlossenen Geräte. Dabei nehmen die einzelnen EtherCAT Slave Teilnehmer nur den für sie bestimmten Datenbereich aus dem Telegramm. Bei diesem Durchlauf werden die Eingangs- und Ausgangsdaten des jeweiligen Gerätes in dem Telegramm entnommen oder eingefügt. Die Prozessdatengröße kann von 1 Bit bis zu 60 KByte groß sein und auf mehrere Frames verteilt werden. [10, Vgl.] Da die Telegrammbearbeitung in der Hardware des EtherCAT Slaves erfolgt, verfügt EtherCAT über eine hohe Netzwerk Performance. [52, 53, Vgl.]
Bei EtherCAT erfolgt der Datenaustausch nach dem Master- Slave Prinzip. In dieser Arbeit stellt das BECKHOFF Board das Mastermodul dar. Die Phoenix Buskoppler sind die Slave-Module. Die Masteransteuerung sendet die Ausgangsdaten an die Slavemodule und die Slavemodule stellen die Eingangsdaten zur Verfügung.
Das EhterCAT Telegramm verwendet das Standard Frame nach IEEE 802.3. Das EhterCAT Protokoll ist für Prozessdaten optimiert und kann direkt über das Ethernet Frame transportiert werden.

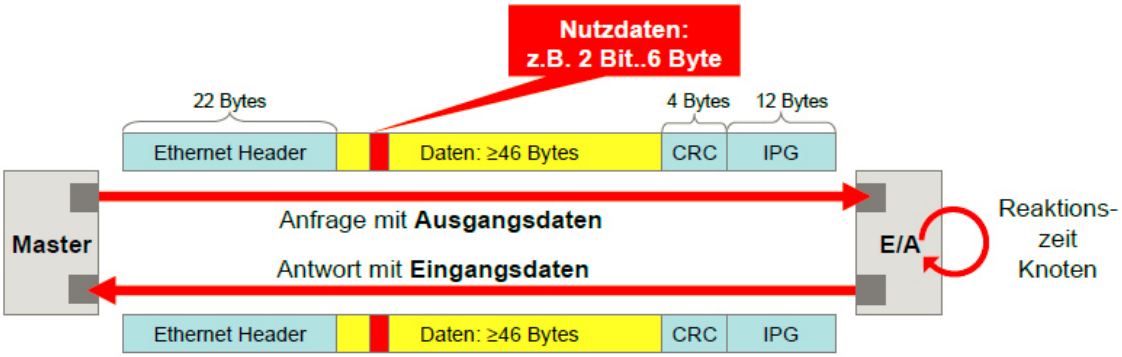


Abbildung 3.18: Aufbau Ethernet Frame mit EtherCAT Protokolldaten [51]

Standardmäßig werden alle I/O-Daten in so wenig wie möglich Ethernet/EtherCAT Frames integriert. Dadurch wird eine effiziente, optimierte und den Feldbus wenig belastende Kommunikation erreicht. Das Ethernet Frame besteht aus einen Header, den Daten und dem Quersummer 'CRC'. Dieser Ethernet Frame trägt die EtherCAT Protokolldaten. Die Daten bestehen wiederum aus einem EtherCAT Header und den Datagrammen. Es gibt im EtherCAT Telegramm bis zu 15 Datagramme. [52, Vgl.] Jedes Datagramm hat einen explizit Lese- oder Schreibauftrag. Die Datagramme bestehen wiederum aus einem Header und Daten. Zusätzlich ist zur Überwachung ein Working Counter 'WC'. Der 'WC' ist eine 16 Bit Zahl, die vom Master mit Null initialisiert losgeschickt wird. Jeder EtherCAT Slave, der von diesem Datagramm schreibend oder lesend angesprochen wird, erhöht diese Zahl um 1,2 oder 3. Dies ist abhängig von der Art des Zugriffs. [52, Vgl.] Wenn das Datagramm die gesamte Struktur durchlaufen hat, kommt der 'WC' mit einem Wert größer Null zum Master zurück. Dem Master ist bekannt, wie viele Slaves dieses Datagramm bearbeiten muss und hat somit einen Erwartungswert für den 'WC'.

Gibt es Differenzen zwischen den erwarteten 'WC' und dem empfangenden 'WC' muss der Master herausfinden welcher Slave betroffen ist und ob das Datagramm nochmals gesendet werden muss.
Ethercat ist ein wahlfrei adressierbares Protokoll. Es ist eine Broadcast-, Multicast- und Querkommunikation zwischen den Slaves möglich. [52, 53, Vgl.]

3.3.1 BECKHOFF Anwendungssystem

System 13: BECKHOFF Evaluation Kit EL9840 und inhomogene dezentrale Peripherie:
Das Gesamtsystem ist wie folgt aufgebaut:

Abbildung 3.19: Gesamtaufbau

Der Master der EtherCAT Struktur ist das BECKHOFF Evaluation Kit EL9840. Dieses Kit besteht aus der Basisplatine EL9800 und dem EtherCAT Controller FB1130. Das Evaluation Kit ist für die Entwicklung von EtherCAT Slaves gedacht. Die Basisplatine EL9800 verfügt über folgende Schnittstellen:

- 8/16 Bit μC Interface
- Digital I/O
- PIC-Interface
- SPI
- FPGA/ PIC- Programmer [54, Vgl.]

Sämtliche Anschlüsse auf der Platine sind offen gelegt und auf Stiftleisten geführt. Welche Schnittstelle auf der Platine genutzt wird, wird über einen PDI Selector eingestellt. Die Kommunikation mit den EtherCAT Slaves erfolgt über den EtherCAT Controller FB1130 mithilfe des EtherCAT Protokolls. Dieser Controller stellt einen EtherCAT Slave mit 2 EtherCAT Ports und einem digitalen Prozessdateninterface dar. [54, Vgl.]
Die PhoenixContact Axioline F-Buskoppler sind die Bindeglieder zwischen dem EtherCAT Netzwerk und dem Axioline F-System. Der Buskoppler besteht aus einem Controller und einem Ein-/ Ausgabemodul. Diese beiden Elemente werden durch einen Lokalbus miteinander verbunden. In dieser Arbeit sind diese Buskoppler die Slaves des EtherCAT Netzwerks. Die Buskoppler sind über einen Lokalbus mit dem Ein-/ Ausgabemodul verbunden. Die typische Zykluszeit des Axioline F-Lokalbusses sind ca. 10 μs.

Die Laufzeit im Buskoppler sind vernachlässigbar (gegen 0 μs). [55, 56, Vgl.] Der Buskoppler hat 2 EtherCAT Ports und diese EtherCAT Anschlüsse sind richtungsgebunden. Der Port X1 ist der Eingangsport für das EtherCAT und der Anschluss für die Anbindung vom Master aus kommend. Der Port X2 ist der Ausgangsport für EtherCAT Port und der Anschluss für weitere Module zur Kopplung im EtherCAT Netzwerk. [55, Vgl.]
Der Buskoppler hat die Drehkodierschalter S1 und S2 für die Adressenvergabe im Netzwerk. Standardmäßig werden den Teilnehmern die Adressen automatisch zugewiesen. Zur Synchronisation der Module sind zwei Modi vorgesehen, die in der TwinCAT Umgebung deklariert werden können.

- SM Synchronous
- DC Synchronous [55, Vgl.]

In SM Synchronous Modus arbeitet EtherCAT System und der Lokalbus asynchron. Der Lokalbus befindet sich im Modus Auto-Run und läuft mit der für die aktuelle Modulkonfiguration minimal möglichen Zykluszeit. Dieser Modus ist auch für den realisierten kaskadierten Zähler aktiv.
Im Modus 'DC Synchronous' ist der Buszyklus des Lokalbusses auf den EtherCAT Zyklus synchronisiert. [55, Vgl.]
Über Ein-/ Ausgabemodul werden die Zählerzustände visuell ausgegeben.Die Slave Controller sind über den jeweiligen Lokalbus mit dem 'AXL F DI8/1 DO8/1' Modul verbunden. Dieses Modul verfügt über 8 digitale Eingänge und 8 digitale Ausgänge. [56, Vgl.]

3.4 SERCOS III- Kommunikation

SERCOS ist ein Echtzeit Kommunikationsstandard für Motion- Control Anwendungen. Es gibt von der SERCOS Technologie 3 Generationen. Die ersten beiden Technologien basieren auf Lichtwellentechnik und werden bei diesem Versuch nicht weiter betrachtet. Das SERCOS III Protokoll verbindet die Offenheit des Standard-Ethernet mit den Anforderungen an Echtzeit-Präzision in der Automatisierungstechnik. Dieser Automatisierungsbus basiert auf den technischen Eigenschaften insbesondere Echtzeit, hohe Performance und Störsicherheit. Das Kommunikationssystem kommt schwerpunktmäßig in MotionControl basierten Systemen zum Einsatz. [7, Vgl. S.11] In SERCOS III wurden 2 Gerätebeschreibungssprachen entwickelt. Sie stellen Funktionen für die offline Konfiguration und für einfache Inbetriebnahme in der Entwicklungsumgebung bereit. Es gibt bei SERCOS III die Gerätebeschreibungssprache SDDML und SPDML. In dieser Arbeit wurde nur SDDML verwendet. Die Sprache ist eine auf XML- basierende Beschreibung für SERCOS III Geräte und orientiert sich an dessen Gerätemodell. Für die SERCOS Slaves wird für die Kommunikation eine FPGA basierender IP-Core benötigt. Bei der Masteranschaltung kann dieser IP-Core softwarebasierend im Basissystem programmiert werden. Bei den Slave-Modulen muss jedoch spezifische Hardware ver-

wendet werden. [7, Vgl. S.11]

Ein Sercos Netzwerk besteht aus einem Master für die Koordination der Prozesse und den Slavegeräten die für die Interaktion mit dem Prozess. Es sind Linien- und Ringtopologien von den Sercos-Systemen umsetzbar. Die an den Sercos-Modulen vorhandenen Kommunikationsanschlüsse dienen zur Anbindung der vor- und nachgelagerten Module. Die voll duplex Patchkabel ermöglichen eine logische Ringstruktur bei der Linien- und Ringtopologie.

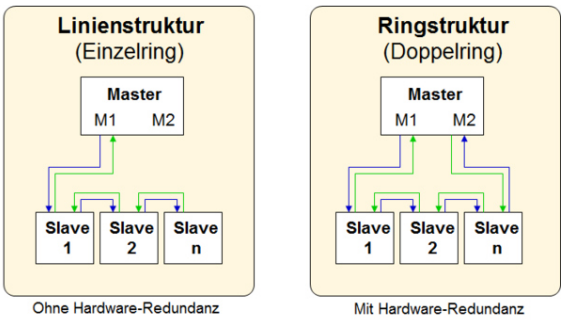


Abbildung 3.20: Sercos Topologievarianten [14]

In der Abbildung 3.20 sind die beiden Topologiearten von Sercos mit der logischen Ringstruktur dargestellt. Sercos III nutzt ein Summenverfahren und lässt somit die Sterntopologie von Standard Ethernet nicht zu. Es unterstützt somit auch keine Switches und Hubs im SERCOS Netzwerk. Durch den Verzicht dieser Komponenten wird die Laufzeit durch das Netzwerk minimiert. In diesem Performancevergleich wurde nur die Ringtopologie verwendet. Diese Topologie ermöglicht die Daten redundant zu übertragen. Dies heißt, dass die Kommunikationsfähigkeit im Netzwerk bei einem Kabelproblem erhalten bleibt.
Durch die beiden Kommunikationsschnittstellen an einem Sercos Gerät wird ein direkter Querverkehr zwischen den Teilnehmern unabhängig, von der verwendeten Topologie, ermöglicht. Somit können die Teilnehmer in einem Zyklus ohne Umweg über den Master miteinander kommunizieren. [7, Vgl. S.11]

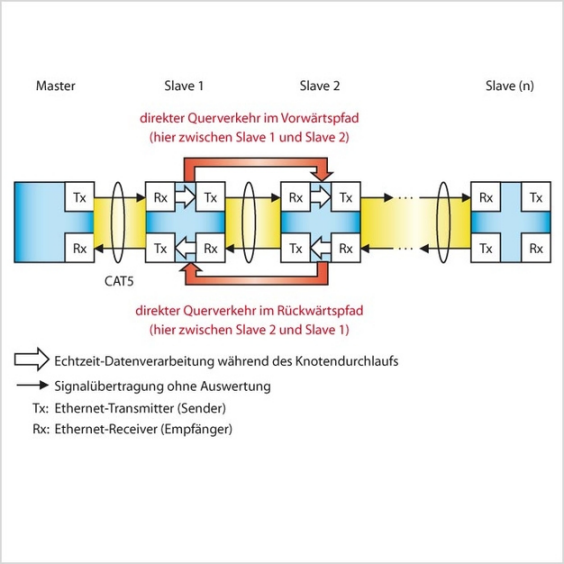


Abbildung 3.21: Sercos Querverkehr [15]

Die Abbildung 3.21 zeigt, dass pro Kommunikationszyklus jedes Echtzeittelegramm den Teilnehmer zweimal durchläuft. Dies wird in der Linientopologie durch den Hin- und Rückweg umgesetzt. In der Ringtopologie wird dies durch den Primär- und den Sekundärring realisiert. Hierdurch entsteht ein kurzer Querverkehr zwischen den Teilnehmern mit einer minimalen Verzögerung der Daten. Die Echtzeitdaten stehen somit synchron an jeder Position des Netzwerkes zur Verfügung.
SERCOS basiert auf einem Master Slave Prinzip mit einer zyklischen Übertragung der Daten. Hierbei wird ein Zeitschlitzverfahren angewendet, dass mithilfe einer Zeitsteuerung in zwei Kanäle unterteilt wird. Es gibt einen Echtzeit- und einen Nicht- Echtzeitkanal. Kollisionsfreiheit dieser Kanalunterteilung wird durch reservierte Bandbreiten gewährleistet. In dem Nicht- Echtzeitkanal können Frames als Einzeltelegramme über TCP/IP und UDP/IP verschickt werden. In dem Echtzeitkanal werden über Broadcast Sammeltelegramme übertragen. Aus diesem Grund wird in Sercos zwischen M/S , DCC/ AT und NRT Telegrammen im Kommunikationszyklus unterschieden. Das MDT Telegramm ist das Datentelegramm, das der Master an die Slaveteilnehmer sendet und das AT Telegramm ist das Telegramm, das die Slave Gräte an den zentralen Master senden. Dieses Telegramm wird auch für den Querverkehr zwischen den Slaves verwendet. Neben der zyklischen Übertragung gibt es in Sercos III auch eine azyklische Übertragung. Zu diesen Diensten zählt ein Service Kanal und das SERCOS Messaging Protokoll. Dieses Protokoll überträgt Funktionsdaten mehrerer Teilnehmer in einem Zeitschlitz mit Hilfe eines Multiplex-Verfahrens. [12, Vgl. S.232]
Das Sercos Telegramm ist von Ethertype 0x88CD basierend auf dem Standard Ethernet Frame. Dieses Telegramm hat eine Länge von 40-1538 Byte und besteht aus dem Ethernet Header, SERCOS Header, Datenteil und der Prüfsumme. In dem Sercos Header sind die Steuer- und Statusinformationen für die Teilnehmer enthalten. Aus diesem Header wird des Weiteren die Synchronisation abgeleitet. Nach dem Header folgt das

Sercos Datenfeld. Dieser Telegrammteil hat eine variable Datenlänge und enthält, neben den Servicedaten, auch die Echtzeitdaten.

3.4.1 BOSCH Rexroth Anwendungssystem

System 14: IndraControl L45 und inhomogene dezentrale Peripherie:
Dieser Systemaufbau besteht aus der BoschRexroth Steuerung IndraControl L45 und der dezentralen Peripherie PhoenixContact “AXL BK S3“. Der Aufbau ist ein inhomogenes System und ist in einer Doppelring Topologie vernetzt.

Die IndraControl L45 ist eine Kompaktsteuerung. Sie ist im oberen Leistungsbereich für anspruchsvolle SPS- und Motion Control Anwendungen anzusiedeln. Der Hochleistungsprozessor ermöglicht den Einsatz in SPS, MotionControl und CNC Anwendungen. Es werden hier die Vorteile der Embedded PC Architektur mit einem standardisierten E/A- System vereint. Sercos III ist die Standard Kommunikationsschnittstelle mit der vollständigen Kontrolle von MotionControl oder I/O Systemen. Zusätzlich ermöglicht eine Onboard MultiProtokoll Schnittstelle eine Feldbuserweiterung für folgende Bussysteme:

- Ethernet TCP/IP
- ProfiBus
- ProfiNet
- EtherNet/IP

An dieser Kompaktsteuerung befinden sich 8 schnelle Ein- und Ausgänge. Diese On-Board E/A’s können mit bis zu 63 Inline Modulen erweitert werden. Über die anwendungsorientierte Taskeinstellung des performanten Motion- Logic Laufsystems lassen sich sowohl schnelle I/O Signalverarbeitungen, als auch hochdynamische Motion Control Aufgaben, realisieren. An dieser Steuerung ist ein LCD Display vorhanden. Über dieses Display können grundlegende Einstellungen der Steuerung vorgenommen werden. Die Steuerung ist mit einem AMD Geode LX800 Prozessor mit 500 MHz versehen. Dieser leistungsfähige Prozessor ermöglicht eine Abarbeitung von 1000 Wort- oder Bit-Anweisungen in 30 μs. Der Speicher für Anwendungen und Applicationen beträgt 256 MB und der Flashspeicher beträgt 1 MByte. Die minimale Taskzykluszeit dieser Steuerung beträgt 1ms.

Für die Ausgabe der Zählerstände wurde der Buskoppler “AXL BK S3“ von PhoenixContact verwendet. Dieser Buskoppler ermöglicht die Anbindung des Axioline F Systems innerhalb eines Sercos III Netzwerks. Es werden Übertragungsraten von 100 MBit/s bei einer minimalen Sercos Zykluszeit von 31,25 μs unterstützt. Mit den zwei Rj45 Ethernet Ports des Controllers konnte eine Doppelringstruktur aufgebaut werden. Die verwendeten Axioline Buskoppler binden über den Lokalbus die Ein- und Ausgabemodule an das

SERCOS- Kommunikationsnetz. Als Ein- und Ausgabemodul wird das Modul “AXL F DI8/1 DO8/1“ verwendet. Die typische Zykluszeit des Systembusses beträgt circa 10μs. Im Koppler sind die sercosspezifischen Profile, Klassen und Funktionsgruppen implementiert. Hierzu zählen das Sercos Gerätemodell (GDP: Generic Device Profil), Sercos Kommunikationsmodell (SCP: Sercos Communication Profil) und das Sercos Funktionsmodell (FSP: Function Specific Profile- IO). Zur Einstellung der Sercos Adresse wurde die Entwicklungsumgebung IndraWorks verwendet. Hierbei werden die Adressen beim Boot des Mastergerätes vergeben und im Koppler remanent gespeichert.

3.5 POWERLINK- Kommunikation

Ethernet POWERLINK ist eine Protokollerweiterung des Ethernetstandards, um Echtzeitdaten im Mikrosekundenbereich zu übermitteln. Entwickelt wurde POWERLINK von der Firma B&R. Seit 2003 entwickelt diese Technologie die Nutzerorganisation Ethernet POWERLINK Standardization Group weiter. POWERLINK ist vollständig konform zum Ethernet Standard IEEE 802.3. Aus diesem Grund zählen die Protokolle TCP/IP und UDP/IP zu dem Übertragungsmechanismen. Diese Protokolle sind für die zeitlich unkritische Kommunikation geeignet. Für die in der Automatisierungstechnik angewandte deterministische Datenübertragung sind diese Protokolle weniger geeignet. [12, Vgl. S.188] Aus diesem Grund wurde bei Ethernet POWERLINK TCP/IP ein echtzeitfähiger POWERLINK Stack implementiert.

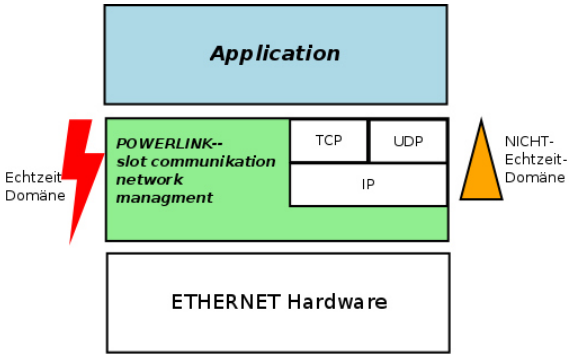


Abbildung 3.22: Powerlink SCNM im klassischen Ethernet

Die prinzipielle Funktionsweise von POWERLINK basiert auf einem isochronen Zeitschlitz- und Pollingverfahren. Durch die Integration von SCNM wird eine Vermeidung von Kollisionen zur optimalen Ausnutzung der vorhandenen Bandbreite erreicht. Es wird weiterhin der Datenaustausch zwischen den Geräten zeitlich organisiert. In einem POWERLINK Netzwerk gibt es somit genau einen Manager, der die Kommunikationssteuerung übernimmt. Dieser Teilnehmer übernimmt die Funktion des “Managing Node“ (MN) und steuert damit den Zeittakt zur Synchronisierung aller Teilnehmer. Die Steuerung der Kommunikation erfolgt mittels Vergabe von Senderechten. Somit können die anderen

Systeme nur senden, wenn sie die entsprechende Berechtigung vom Manager haben. Die so genannten Controller haben die Funktionalität des “Controlled Nodes“ (CN). Hat ein Controller eine Sendeberechtigung, werden diese in Broadcast versendet. Somit können die Daten von allen anderen Systemen mit empfangen werden. Der Datenverkehr zwischen dem Manager und den Controllern findet auf Basis von deterministischen, isochronen Zyklen statt. [9, Vgl. S.304] Bei dieser Arbeitsweise gibt es 4 Zyklusperioden.

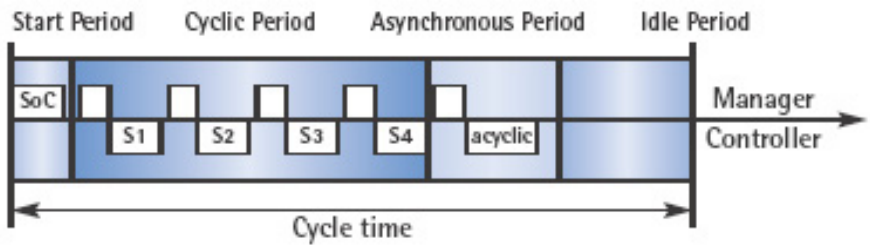


Abbildung 3.23: Zyklusperioden der POWERLINK Kommunikation

Wie in der Abbildung 3.23 besteht der POWERLINK Zeitzyklus aus folgenden Perioden:

- Start Periode
- Zyklische Periode
- Azyklische Periode
- IDLE Periode

Bei dem Startzyklus (Start Periode) werden die Frames für alle Systeme im Netzwerk synchronisiert. Diese Periode ist somit wesentlich für die Qualität des POWERLINK Netzwerks und für die Minimierung des Netzwerk- Jitters verantwortlich. Nach diesem Frame folgt die zyklische Periode. Hierbei ordnet der Manager sequenziell jedem aktiven Controller einen Kommunikationsslot zu und berechtigt somit die Station zum Senden von Daten. Die azyklische Periode ist für die asynchrone Kommunikation der Teilnehmer bestimmt. Dabei bekommen nur Stationen Senderecht, wenn diese vorhergehend eine Aufforderung ausgesendet haben. [9, Vgl. S.304] POWERLINK unterscheidet zwischen Echtzeit-Domänen und Nicht-Echtzeit-Domänen. Da die Datenübertragung der asynchronen Phase Standard-IP-Frames unterstützt, trennen Router (siehe Abbildung 3.22) die Daten transparent von den Echtzeit-Domänen. Die letzte Periode ist die sogenannte IDLE Periode. Diese Periode ist für eine Restzeit, die von der asynchronen Periode genutzt werden kann. Aufgrund der unterschiedlichen zeitlichen Differenzen der azyklischen Kommunikation kann die IDLE Time in ihrer Dauer schwanken.

POWERLINK verwendet die MAC Adressierung konform zu IEEE 802.3. Jedes Gerät hat eine eindeutige MAC Adresse. Zusätzlich wird jedem Gerät eine eindeutige Knoten-

nummer zugewiesen. Dieses zweite Adressschema hat einen 1 Byte langen 'POWERLINK NOTE ID', der mittels Knotenschalter eingestellt wird.

3.5.1 B&R Anwendungssystem

System 15: PP65 und homogene dezentrale Peripherie:
Unter POWERLINK wurde der dekadische Zähler durch ein System der Firma B&R umgesetzt. Das Gesamtsystem ist ein homogenes System, bestehend aus einer Panel SPS und fünf X20 Buskopplern. Die Panel SPS 'PP65' wird als Steuerung und Bedieneinheit eingesetzt. Alle Komponenten der Panel SPS sind in einem kompakten Gehäuse mit 5,7“ QVGA TFT Display untergebracht. Das Display ist ein resistiver Touchscreen. Für die Unterstützung von Busapplikationen und Visualisierungsoberflächen wird eine Compact Flashkarte benötigt. Das Power Panel verfügt über ein eigenes BIOS, 128 MByte DRAM und 232 kByte SRAM. Intern arbeitet ein Geode LX800 Prozessor mit 500MHz Taktfrequenz. Auf Grund der hohen Rechenleistung ist das Panel besonders für Mehrachsanwendungen geeignet. [32, Vgl.]

Die dezentrale Peripherie besteht aus einem Buscontroller X20BC0083, einem Versorgungsmodul X20PS9400, einem Eingangsmodul X20DI9371 und einem Ausgangsmodul X20DO9322. Diese Gesamteinheit ermöglicht eine Kommunikation mit dem Buscontroller über POWERLINK. Mit X2X-Link wird das Lesen und Schreiben der Ein- und Ausgänge der digitalen I/O Module ermöglicht.

Der X2X-Link Bus ist eine Lösungsvariante zur Eliminierung des Flaschenhalsproblems bei Feldbussen. Auf Grund der Tatsache, dass alle I/O Prozessdaten zentral verarbeitet werden, entsteht ein Problem hinsichtlich des Transportes der Daten über den Feldbus. So ist es Dank der Rechenleistung von modernen CPUs die Programmabarbeitung kein Problem, jedoch bilden die Feldbussysteme mit ihren Übertragungsgeschwindigkeiten einen Engpass. Zur Lösung müssen aufwendige Maßnahmen zur Dezentralisierung vorgenommen werden. Hier greift das X2X Link Busprinzip ein. Dies wird, durch ein im weitesten Sinne Herausführen des CPU-Rückwandbusses aus der Steuerung in das Patchkabel, erreicht.

Der Bus Controller X20BC0083 für die Zählerstandausgabe erlaubt die Kopplung von X2X Link I/O Knoten an POWERLINK V1/V2. Dabei gibt es die Möglichkeiten, den X2X Link Zyklus (1:1 oder über einen Prescale) synchron zum POWERLINK Zyklus zu betreiben. Die Schnittstellen des Controllers umfassen eine POWERLINK V1/V2 Schnittstelle mit integriertem dual Port Hub. Somit kann das System als Linientopologie realisiert werden. Am Buskoppler befinden sich zum einen ein Einspeisemodul und zum anderen 1 Byte Ausgabe- und 1 Byte Eingabekanal. Das Einspeisemodul X20PS9400 ist für

die interne I/O Versorgung des Buscontrollers zuständig. Des Weiteren dient es für die Spannungsversorgung des X2X- Link Busses. Dabei ist die Spannungsversorgung des Buscontrollers und die Busversorgung des X2X Link galvanisch voneinander getrennt. Das Modul X20DI9371 ist ein digitales Eingangsmodul mit 12 Eingängen. Das digitale Ausgangsmodul X20DO9322 besitzt 12 Ausgänge . Der maximale Ausgangsstrom beträgt 500 mA. Die Kommunikation mit anderen Modulen erfolgt mit X2X Link. [33, Vgl.]

System 16: CP20 und homogene dezentrale Peripherie:

Mit diesem System wird eine sehr leistungsfähige Konfiguration unter Powerlink umgesetzt. Das System besteht aus einer B&R X20 CPU und X20 Buskopplern. Die Buskoppler sind im Systemaufbau 15 näher beschrieben. Die X20CP1586 CPU findet Anwendungsbereiche bei Standardapplikationen bis hin zu anspruchsvollen Anwendungen. Dieser Controller ist ein Modell der Serie X20 von B&R. Bei dieser CPU können Zykluszeiten von 100μs realisiert werden. Die CPU verfügt über folgende Schnittstelen:

- Ethernet
- X2X Link
- USB

Die Standardkonfiguration ermöglicht einen Zugriff auf USB Geräte und auf weitere umfangreiche Netzwerkanbindungen. Für die Unterstützung von harter Echtzeit nutzt diese CPU den Ethernet Feldbus Powerlink. Hierüber kann zum einen die dezentrale Peripherie angeschlossen werden oder zum anderen auch die direkte Anbindung von Achsenmodulen durchgeführt werden. Neben diesen standardmäßig konfigurierten Schnittstellen verfügt der Controller auch über freie Steckplätze für zusätzliche Schnittstellenmodule. An den Controller können bis zu 250 X20 I/O Module angeschlossen werden. Dies entspricht einer Gesamtanzahl von 3000 Kanälen. Die X20 Zentraleinheit hat einen Intel Atom Prozessor mit 1,6 GHZ Taktung. Dieser leistungsstarke Prozessor verspricht kürzeste Taskzykluszeiten von 100μs und eine typische Befehlszykluszeit von 2,7ns. [31, Vgl.] Der Controller hat einen Arbeitsspeicher von 512 MByte und einen Daten RAM von 1 MByte. Eine nähere Beschreibung des im vollständigen Systemaufbau verwendeten X20 Buskoppler ist im Systemaufbau 15 vorzufinden.

3.6 PROFIBUS- Kommunikation

Profibus ist ein universelles, offenes, standardisiertes und digitales Kommunikationssystem. Es findet Anwendung in der Prozess- und Automatisierungsindustrie, der Verkehrstechnik und in der Energieerzeugung. Profibus wurde von der Organisation Profibus und dem Hersteller SIEMENS entwickelt. Es ist durch die Norm IEC 61158 stan-

dardisiert. Profibus ist ein serielles Bussystem. Dieses Bussystem orientiert sich aus technologischer Sicht an die Protokollarchitektur des OSI Referenzmodells. Es werden bei Profibus die Schichten 1, 2 und 7 verwendet. In der ersten Schicht (Physikal Layer) wird festgelegt, in welcher Weise die Daten übertragen werden. Bei Profibus werden die Daten über Kupferkabel, optisch oder über MBP übertragen. Bei der elektrischen Übertragung über Kupferkabel werden verdrillte Zweidrahtleitungen eingesetzt. Die optische Übertragung wird über Lichtwellenleiter in der Stern-, Linien und Ring Topologie realisiert. Bei der MBP -Installation erfolgt der Datentransfer und die Stromversorgung der Profibusteilnehmer über eine Leitung. In der 2. Schicht (Data Link Layer) wird bei Profibus das Buszugriffsverfahren festgelegt. Hierbei wird das Master- Slave Verfahren, mit der Erweiterung des Token Passing's für die Koordinierung mehrerer Master am Bus, verwendet. Das Token-Passing-Verfahren garantiert die Zuteilung der Buszugriffsberechtigung, dem Token, innerhalb eines genau festgelegten Zeitrahmens. Die Token-Nachricht ist ein besonderes Telegramm zur Übergabe der Sendeberechtigung von einem Master an den nächsten Master. Das Master-Slave-Verfahren ermöglicht es, dem Master (aktiver Teilnehmer), der gerade die Sendeberechtigung besitzt, die ihm zugeordneten Slave-Geräte (passive Teilnehmer) anzusprechen. Der Master hat hierbei die Möglichkeit, Nachrichten an die Slaves zu übermitteln bzw. Nachrichten von den Slaves abzuholen. [17] Neben diesem Verfahren werden in der zweiten Schicht auch die Funktionen der Datensicherung sowie die Abwicklung der Übertragungsprotokolle und der Telegramme festgelegt.

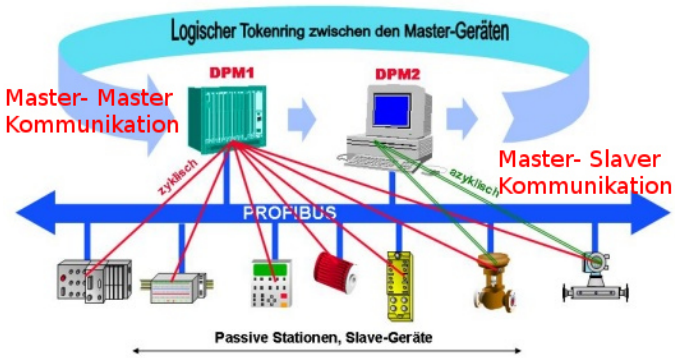


Abbildung 3.24: Kommunikation unter Profibus [17, modifiziert]

Die 7. Schicht (Application Layer) ist die Anwendungsschicht. Dies ist die Schnittstelle zum Anwenderprogramm. [12, Vgl. S.108]
Es haben sich bei Profibus 3 verschiedene Entwicklungen herauskristallisiert:

- Profibus- FMS (Fieldbus Message Specification) ist für den Einsatz in der klassischen Feldebene konzipiert.
- Profibus- DP (Dezentrale Peripherie) ist die Weiterentwicklung von Profibus-DP
- Profibus- PA (Prozess Automation) ist eine Spezialisierung des Bussystems für Kommunikationsaufgaben zum Prozessumfeld.

Alle Profibus-Varianten verwenden ein einheitliches Buszugriffsprotokoll (FDL). In diesem Performancevergleich wurde ausschließlich Profibus- DP verwendet. Es ist eine Erweiterung des Profibus- FMS für den objektnahen Einsatz in der Aktor Sensor Ebene. Mit Profibus- DP werden überwiegend Singlemastersysteme aufgrund der Geschwindigkeitsvorteile realisiert. Neben dem Singlemastersystem kann unter Profibus- DP auch das Multimastersystem realisiert werden. [16, Vgl. S. 371] Als Übertragungstechnik wurde eine RS-485 Schnittstelle verwendet. Sie erlaubt eine hohe Übertragungsrate und ist über ein verdrilltes und geschirmtes Kupferkabel verbunden.

Die Buszugriffssteuerung (MAC, Medium Access Control) legt das Verfahren fest, zu welchem Zeitpunkt ein Busteilnehmer Daten senden kann. Über die MAC muss sicherstellen werden, dass zu einem Zeitpunkt immer nur ein Teilnehmer die Sendeberechtigung besitzt. Das Profibus-Buszugriffsprotokoll arbeitet verbindungslos. Es ermöglicht neben der logischen Punkt-zu-Punkt-Datenübertragung auch Mehrpunktübertragung mit Broadcast- und Multicast-Kommunikation. [17] Hierbei wird unter folgenden Telegrammenarten unterschieden.

- Token Telegramme
- Telegramme ohne Datenfeld
- Telegramme mit variabler Länge oder fester Datenlänge
- Kurztelegramme

Kurztelegramme werden nur für kurze und positive Antworten genutzt. Telegramme mit fester Datenlänge sind generell 8 Byte groß. Telegramme mit variabler Datenlänge haben Nutzdaten in einer Bandbreite von 1 bis 246 Byte.

3.6.1 SIEMENS Anwendungssystem

System 17: S7-315F und dezentrale Peripherie ET200s:

In diesem System wurde ein homogener Aufbau unter ProfiBus realisiert. Der Systemaufbau besteht aus dem Mastergerät S7-315F 2-PN/DP und der dezentralen Peripherie ET200s. Der Hardwareaufbau des Mastersystems ist identisch mit dem Aufbau von System 1. In diesem Systemaufbau erfolgt die Kommunikation zu den E/A-Teilnehmern über Profibus. [36, Vgl.]

Abbildung 3.25: S7315F-2 PN/DP– ET200s Systemaufbau ProfiBus

Für die Anbindung der ET200s über die RS485 Schnittstelle an den ProfiBus DP wird das Interfacemodul IM151-1 Basic verwendet. Auf diesem Interfacemodul befindet sich ein DIP- Schalter zur Einstellung der Profibus DP- Stationsadresse. Für die Fehlerdiagnose befinden sich auf dem Modul Diagnoseanzeigen für Sammelfehler, Busfehler und zur Spannungsversorgung. Die maximale Parameterlänge dieses Moduls beträgt 198 Byte. Daraus folgt ein maximaler Adressumfang von 88 Byte für Eingänge und Ausgänge. Das ET200 Modul wird als DPV0 Slave betrieben. [37, Vgl.]

(9) zaehler1-DP-315F

Steckplatz	Baugruppe	Bestellnummer	E-Adresse	A-Adresse
1	PM-E DC24..48V	6ES7 138-4CA50-0AB0		
2	2DI DC24V HF	6ES7 131-4BB01-0AB0	2.0...2.1	
3	2DI DC24V HF	6ES7 131-4BB01-0AB0	2.2...2.3	
4	2DI DC24V HF	6ES7 131-4BB01-0AB0	2.4...2.5	
5	2DI DC24V HF	6ES7 131-4BB01-0AB0	2.6...2.7	
6	PM-E DC24..48V	6ES7 138-4CA50-0AB0		
7	2DO DC24V/0,5A HF	6ES7 132-4BB01-0AB0		2.0...2.1
8	2DO DC24V/0,5A HF	6ES7 132-4BB01-0AB0		2.2...2.3
9	2DO DC24V/0,5A HF	6ES7 132-4BB01-0AB0		2.4...2.5
10	2DO DC24V/0,5A HF	6ES7 132-4BB01-0AB0		2.6...2.7

Abbildung 3.26: ET200s Rackaufbau

Die Aufbau der Einsteckkarten entspricht dem von System 1. Es gibt 2 Powermodule und jeweils 1 Byte Eingangs- und Ausgangskanäle. Dies wird jeweils durch vier 2 Bit Kanal Einsteckkarten realisiert.

System 18: S7-400 und dezentrale Peripherie ET200s:

Dieser Systemaufbau besteht aus einer S7-413-2DP als Profibusmaster und fünf ET200s als Profibus Slavegeräte. Hierdurch wird ein homogener Systemaufbau realisiert, der wie folgt konfiguriert ist:

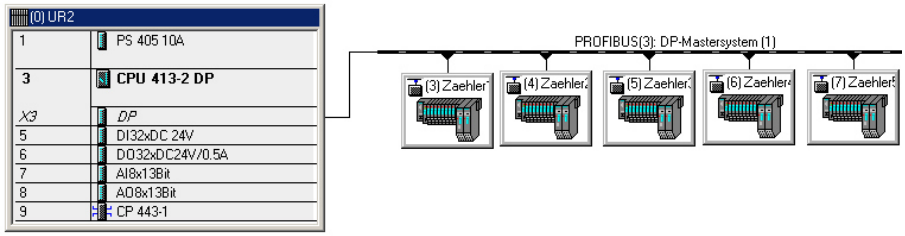


Abbildung 3.27: S7 413-2DP– ET200s Systemaufbau ProfiBus

Als dezentrale Peripherie für die Zählerstandausgabe wurden baugleich konfigurierte ET200s Module verwendet. Das Profibusmastergerät ist eine S7- 413 Steuerung. Diese Steuerungen sind in der Fertigungs- und Prozessindustrie sehr weit verbreitet. Aufgrund der hohen Kommunikationsleistung, wird die S7-400 als übergeordnete Steuerung von datenintensiven Aufgaben für die Kommunikation an untergelagerte Systeme genutzt. Die eingesetzte CPU ist durch eine schnelle Bearbeitungsgeschwindigkeit und deterministische Reaktionszeiten gekennzeichnet. Die S7-400 lässt sich modular aufbauen. Auf dem leistungsfähigen Rückwandbus der S7-400 sind neben dem Powermodul noch 4 digitale und analoge Einsteckkarten verbaut. Die digitalen I/O Karten besitzen jeweils 32 Eingangs- und Ausgangskanäle. Die analogen Karten besitzen jeweils 8x13 Bit Eingänge und Ausgänge. Des Weiteren ist in dem Systemaufbau ein CP 443-1 verbaut. Dieser Kommunikationscontroller ermöglicht den Zugriff über TCP/IP auf den Systemaufbau. Auf dem Controller befindet sich neben der Profibus Schnittstelle auch eine Multi-Point-Interface Schnittstelle. Sie kann zur Programmierung oder zum Anschluss von Bedieneinheiten genutzt werden. Der leistungsstarke Prozessor des Controllers hat eine Verarbeitungsgeschwindigkeit von 0.2ms pro 1000 Wort Anweisungen. [38, Vgl.] Der Controller besitzt einen Arbeitsspeicher von 72 kb. Für die Diagnose, Programmierung und Inbetriebnahme wird der SIMATIC Manager V5.5 verwendet.

System 19: S7-MicroBOX und dezentrale Peripherie ET200s:

Der Aufbau 19 besteht aus der SIMATIC Microbox PC 427B und der dezentralen Peripherie ET200s. Dieser Systemaufbau wird über ProfiBus DP angesteuert.

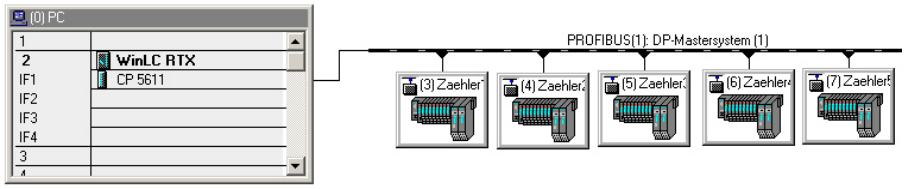


Abbildung 3.28: SIMATIC Microbox PC 427B– ET200s Systemaufbau ProfiBus

Der ProfiBus Master Microbox 427B wird durch den Kommunikationsprozessor CP5611 und die Soft SPS WinLC RTX konfiguriert. Den Kommunikationsprozessor benötigt man

für die Profibusanbindung. Die Zentralfunktionen des Systems wird von der SIMATIC PC-based Control mit WinLC RTX wahrgenommen. Sie ist mit den aufeinander abgestimmten Hard- und Softwarekomponenten für PC- based Automation abgestimmt. Auf diesem Industrie embedded PC lassen sich alle anfallenden Automatisierungsaufgaben wie Steuern, Regeln, Bedienen & Beobachten und MotionControl auf einer Plattform realisieren. Die Soft SPS WinLC RTX ist für Aufgaben, die eine hohe Flexibilität und Integrationsfähigkeit erfordern, optimiert. Der Microbox PC umfasst einen Embedded PC mit der SIMATIC Automatisierungssoftware WinLC RTX. Als Betriebssystem kommt bei diesem System Windows XP Embedded zum Einsatz. Hierbei wird der Windows Logic Controller für Windows 2000/XP mit Ardence RTX Echtzeitumgebung verwendet. Dieses Gerät wird als DPV1 Profibus Gerät betrieben. Neben dem Profibus DP Anschluss weist die Microbox noch die Schnittstelle für die S7- Kommunikation aus. Sie zeichnet sich durch Äquidistanz und Taktsynchronität aus. In dem Gerät ist ein Intel Pentium M mit 1,40 MHz vorhanden. Der Hauptspeicher im Gerät beträgt 1 GB. [39, Vgl.] Als dezentrale Peripherie für die Zählerstandausgabe wurden konfigurationsgleiche ET200s Module, wie im Systemaufbau 17 und 18, verwendet. Für die Diagnose, Programmierung und Inbetriebnahme wird der SIMATIC Manager V5.5 verwendet.

3.7 Gesamtübersicht

Die Tabelle 3.1 gibt einen Gesamtüberblick über alle verwendeten Systeme. In den unterschiedlichen Testaufbauten ist ein breites Spektrum von Feldbusarten und verschiedenen Steuerungungsarten vertreten.
Diese Aufbauten ermöglichen einen Gesamtvergleich der Systeme untereinander. Zusätzlich können Performanceuntersuchungen der unterschiedlichen Feldbusse, der Steuerungssystemen und der dezentralen Peripherie getätigt werden.

Basis für den Vergleich sind unterschiedliche Steuerungssysteme. Dabei können die unterschiedlichen Steuerungsarten untereinander verglichen werden. Hier stehen zur Betrachtung Hard SPS und Embedded Steuerungen. Es wurden Steuerungen unterschiedlicher Leistungsklassen herangezogen. Sie unterscheiden sich in der CPU Architektur. Manche Systeme, beispielsweise die SIEMENS Geräte, haben eine Doppelkernstruktur für eine Strukturierung zwischen Kommunikation- und Datenprozess und andere Systeme haben eine Einkernarchitektur für diese Prozessverarbeitung. Hierbei ist es auch interessant, wie sich die Performence der Zählstruktur bei den Systemen aus unterschiedlichen Generationen entwickelt hat.

Es wurde verglichen, inwieweit ein homogener oder inhomogener Aufbau der Peripherie Einfluss auf das Zählverhalten und dementsprechend der Performence des Systems hat. Hierfür kann der Systemaufbau 1 und 4 herangezogen werden. In den beiden Systemen wird unter anderem das gleiche Mastergerät verwendet, aber unterschiedliche Peripherie Geräte. Es werden zum einen SIEMENS ET200s Geräte verwendet und zum

anderen WAGO 750-370 Buskoppler. Diese Geräte unterscheiden sich schon von der Leistungsklasse. Die ET200s Geräte sind High Feature Baugruppen die eine Aktualisierungszeit von 1ms erlauben, bei den Wagobaugruppen haben wir hier eine Aktualisierungszeit von 16ms.

Ein anderer Vergleich ist der Vergleich der unterschiedlichen Feldbussysteme bei gleichen Systemaufbau. Für diese Betrachtung sind die Ergebnisse von System 6 und 12 nutzbar. Durch die Multiprotokollfähigkeit der E/A-Komponenten der Firma TURCK ist es möglich, einen Performancevergleich zwischen ProfiNet und Modbus durchzuführen. Dieser Vergleich ist auch bei dem Systemaufbau 1 und 17 möglich. Hierbei wird das gleiche Steuerungssystem (S7-315F) verwendet, das sich hinsichtlich der Ansteuerung zur Peripherie über ProfiNet oder ProfiBus unterscheidet. Dafür wurden ET200s-Module verwendet, die eine ProfiNet-Schnittstelle (Modell IM 151-3) und Module die eine Profibus-Schnittstelle (Modell IM 151-1) besitzen.

System	Master	Slave 1	Slave 2	Slave 3	Slave 4	Slave 5	Protokoll
1.	S7-315F	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiNET
2.	S7-1511	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiNET
3.	S7-1516	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiNET
4.	S7-315F	WAGO 750-370	WAGO 750-370	WAGO 750-370	WAGO 750-370	WAGO 750-370	ProfiNET
5.	S7-317	ILB PN 24	ILB PN 24	FL IL 24	WAGO 750-340	WAGO 750-340	ProfiNET
6.	VT250	BL20	BL20	FEN20	TBEN-L1	TBEN-L1	ProfiNET
7.	ILC350	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiNET
8.	AXC1050	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiNET
9.	AXC1050	AXL F BK PN	AXL F BK PN	AXL F BK PN	AXL F BK PN	AXL F BK PN	ProfiNET
10.	Modicon TSX	BL20	BL20	FEN20	MOMENTUM	MOMENTUM	ModbusTCP
11.	AXC1050	AXL F BK ETH	AXL F BK ETH	AXL F BK ETH	AXL F BK ETH	AXL F BK ETH	ModbusTCP
12.	VT250	BL20	BL20	FEN20	TBEN-L1	TBEN-L1	ModbusTCP
13.	EvaKit EL9840	AXL F BK EC	AXL F BK EC	AXL F BK EC	AXL F BK EC	AXL F BK EC	EtherCAT
14.	IntraControlL45	AXL BK S3	AXL BK S3	AXL BK S3	AXL BK S3	AXL BK S3	Sercos III
15.	PP65	X20	X20	X20	X20	x20	PowerLink
16.	X20CP	X20	X20	X20	X20	x20	PowerLink
17.	S7-315F	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiBUS
18.	S7-400	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiBUS
19.	S7-MicroBOX	ET200s	ET200s	ET200s	ET200s	ET200s	ProfiBUS

Tabelle 3.1: Gesamtübersicht der Systeme

4 Systembetrachtungen

In diesem Kapitel werden allgemeine Betrachtungen zu den Testsystemen vorgenommen.
In den realisierten Systemaufbauten läuft jeweils nur ein Programm- Task ab. Dieser Task steuert den zeitlichen Ablauf der Programme. Das Programm ist diesem Task zugeordnet. Die CPU führt die Anweisungen aus, die in dem angewählten Arbeitsspeicher abgearbeitet werden. Der Zugriff für die konfigurierten Variablen wird über den internen Daten- und Adressbus von der CPU zum Arbeitsspeicher gesteuert. Grundlegend ist die zyklische Abarbeitung des Programms in folgende Abschnitte eingeteilt.

- Einlesen der Prozessdaten
- Bearbeitung des zyklischen Programms
- Schreiben der Prozessdaten

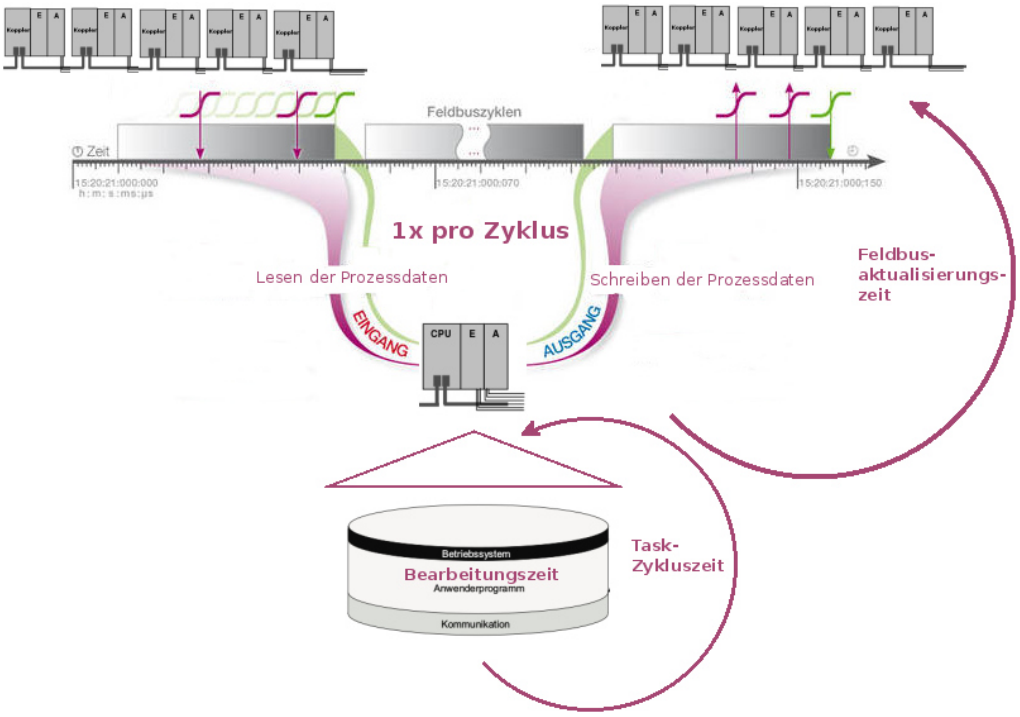


Abbildung 4.1: Einflussfaktoren auf das Performanceverhalten des Systemaufbaus

Wie in der Abbildung zu sehen ist, ist die Performence der Systeme abhängig sowohl von der Zykluszeit des Feldbusses als auch der Zykluszeit der CPU. Die Zykluszeit der SPS ist die Zeit für die Abarbeitung eines Zyklus, einschließlich aller Kommunikationsaufgaben. Sie ist abhängig von der Rechenzeit des Programms, also abhängig von

der Zahl der Anweisungen. Die Aktualisierungszeit der Teilprozessabbilder ist abhängig von der Anzahl der zugeordneten zentralen und dezentralen Peripheriedaten. Das Betriebssystem der CPU stellt für die Kommunikation CPU Verarbeitungsleistung zur Verfügung. Wenn die Verarbeitungsleistung für die Kommunikation nicht benötigt wird, steht die Verarbeitungsleistung dem Betriebssystem als freie Ressource zur Verfügung. Das Anwenderprogramm hat ohne Unterbrechungen eine gewisse Laufzeit. Die Länge der Programmlaufzeit des Anwenderprogramms ist ausschließlich von der Anzahl der Operationen und verwendeten Funktionen abhängig.

System	Master	Protokoll	Zykluszeit	Bearbeitungszeit
1.	S7-315F	ProfiNET	∼0.320ms	90ns
2.	S7-1511	ProfiNET	∼0.150ms	72ns
3.	S7-1516	ProfiNET	∼0.100ms	12ns
4.	S7-315F	ProfiNET	∼0.320ms	90ns
5.	S7-317	ProfiNET	∼0.380ms	0.05ms /kAW
6.	VT250	ProfiNET	∼2ms	###
7.	ILC350	ProfiNET	3ms	0.5ms /kAMix
8.	AXC1050	ProfiNET	4ms	1,3ms /kAMix
9.	AXC1050	ProfiNET	4ms	1,3ms /kAMix
10.	Modicon TSX	ModbusTCP	1ms	45-60ns
11.	AXC1050	ModbusTCP	4ms	1,3ms /kAMix
12.	VT250	ModbusTCP	∼3.9ms	###
13.	EvaKit EL9840	EtherCAT	∼0.350ms	###
14.	IntraControlL45	Sercos III	1ms	0,03 ms /kAMix
15.	PP65	PowerLink	0.4ms	###
16.	X20CP	PowerLink	0.1ms	2,7ns
17.	S7-315F	ProfiBUS	∼0.320ms	90ns
18.	S7-400	ProfiBUS	∼0.900ms	0.2ms /kAW
19.	S7-MicroBOX	ProfiBUS	∼0.022ms	4ns

Tabelle 4.1: Systeme und deren Performancevorraussetzungen

Die Tabelle 4.1 zeigt die jeweiligen Systeme mit den Zyklus- und den Bearbeitungszeiten für Wortoperationen. Es ist erkennbar, dass es schwer ist, für die einzelnen Bearbeitungszeiten der verschiedenen Systeme eine vergleichbare Basis anhand der Datenblätter zu finden. Die Bearbeitungszeiten die mit Rautezeichen “#“ gekennzeichnet sind, konnten anhand der Herstellerdatenblätter nicht herausgefunden werden. Es ist jedoch zu erkennen, dass sich die Zykluszeiten der Steuerungen anhand der Bearbeitungszeiten dimensionieren.

Die Zykluszeiten von den Siemens Systemen konnte nicht ohne weiteres ermittelt werden. Bei diesen Systemen ist ein Task mit keiner Mindestlaufzeit realisiert worden. Bei allen anderen Systemen ist trotz freilaufenden Task eine nicht deaktivierbare Mindestlaufzeit der Task System seitig vorgegeben. Bei den Siemens Systemen musste die

Taskzykluszeit mithilfe des OB35 ermittelt werden. Der Zeitbaustein wird beim Start des Systems aktiviert und sobald die Ausgänge an der Peripherie Baugruppe 3 den Wert 10 erreichen, also das erste und dritte Bit gesetzt ist, wird der Zeitbaustein durch ein Rücksprung aus den OB35 gestoppt. Für diesen Vorgang wird der OB1 auskommentiert. Anhand dieses Zeitwerts (ms) kann die ungefähre Zykluszeit des jeweiligen Systems ermittelt werden. Die Zykluszeit wird anhand einer Division durch die Anzahl der CPU Zyklen bis zum Erhalt des entsprechenden Zählwerts an der dritten Dekade errechnet. Dieser Wert ist nur ein Richtwert für eine Vorbetrachtung der Zählerentwicklung der einzelnen Dekaden.

Neben den Bearbeitungs- und Zykluszeiten spielt die interne Architektur der CPU eine entscheidende Rolle. Die Siemens Steuerungen besitzen 2 Prozessorkerne. Einen Kern für die Kommunikation und einen Kern für die Datenverarbeitung. Das Betriebssystem der CPU stellt für die Kommunikation CPU Verarbeitungsleistung zur Verfügung. Wird diese auf verschiedene Kerne aufgeteilt, kann die Performance des Gesamtsystems erhöht werden. Über die Architektur kann bei den einzelnen Systemen kaum eine Aussage getroffen werden, da dies aus den Datenblättern der Hersteller nicht ersichtlich ist. Des weiteren ist der Umstand zu beachten, dass die Soft SPS basierenden Steuerungen keinen Rückwandbus besitzen. Dieser Fakt ist vor allem an der enormen Performence der SIEMENS Microbox zu erkennen. Anhand der Zykluszeiten der einzelnen Systeme lassen sich die Zeiten für die einzelnen Dekaden über einfache Hochrechnung prognostizieren.

System	Master	Dekade 1 /ms	Dekade 2 /s	Dekade 3 /h	Dekade 4 /d	Dekade 5 /Y
1	S7-315F	81.6	20.81	1.47	15.62	10.91
2	S7-1511	38.25	9.75	0.69	7.33	5.12
3	S7-1516	25.5	6.5	0.46	4.89	3.42
4	S7-315F	81.6	20.81	1.47	15.62	10.91
5	S7-317	96.9	24.71	1.75	18.59	12.99
6	VT250	765	195.08	13.82	146.84	102.59
7	ILC350	765	195.08	13.82	146.84	102.59
8	AXC1050	1020	260.1	18.42	195.71	136.73
9	AXC1050	1020	260.1	18.42	195.71	136.73
10	Modicon	255	65.03	4.61	48.98	34.22
11	AXC1050	1020	260.1	18.42	195.71	136.73
12	VT250	994.5	253.6	17.96	190.83	133.32
13	EvaKit	89.25	22.76	1.61	17.11	11.95
14	IntraControl	255	65.03	4.61	48.98	34.22
15	PP65	102	26.01	1.84	19.55	13.66
16	X20CP	25.5	6.5	0.46	4.89	3.42
17	S7-315F	81.6	20.81	1.47	15.62	10.91
18	S7-400	229.5	58.52	4.15	44.09	30.8
19	MicroBox	5.61	1.43	0.1	1.06	270.3

Tabelle 4.2: Prognostizierte Zählerentwicklungen anhand der Zykluszeiten

Bemerkung: Die Zeitbasis der 5 Dekade vom System 19 ist **Tage "/d"**!
Die Zeitbasis der unterschiedlichen Steuerungssysteme musste entsprechend der verschiedenen Systemwelten angepasst werden. Ohne diese Umsetzung wären die Ergebnisse kaum aussagekräftig. Aus diesem Grunde wurde die Zeitbasis der einzelnen Dekaden übergreifend normiert.

In der Tabelle 4.2 sind Zeiten für die einzelnen Dekaden anhand der Zykluszeiten prognostiziert. Diese Werte wurden auf Grundlage der Zykluszeiten der Steuerungen aus der Tabelle 4.1 berechnet. Durch die dekadische Struktur kann dies durch mathematisches addieren realisiert werden. Hieraus wird ersichtlich, dass es wesentliche Unterschiede zwischen den einzelnen Systemen bestehen. Durch die Dekadenstruktur haben nur geringe Differenzen von wenigen μs der einzelnen Zykluszeiten einen großen Einfluss auf die Zählerstandentwicklung des jeweiligen Systems.

Es ist hierbei aber schon zu beachten, dass alle Systeme, außer die Siemens Steuerungen, eine Mindesttasklaufzeit trotz freilaufenden Task besitzen. Die Systeme können meist eine weit geringere Zykluszeit realisieren, die unter Mindesttasklaufzeit liegt.

5 Implementierung

Im folgenden Kapitel wird beschrieben, wie die einzelnen Systeme implementiert wurden. Dabei werden zunächst die Emwicklungsumgebungen der einzelnen Hersteller vorgestellt. Danach wird auf die einzelnen Systeme eingegangen, die mithilfe der entsprechenden Entwicklungsumgebung realisiert wurden. Dazu zählen auch Probleme, die dabei aufgetreten sind.

5.1 SIEMENS Implementierung

Für die Implementierung der Steuerungssysteme des Herstellers SIEMENS wurden die Inbetriebnahmen mit Step7 V5.5 oder dem TiaPortal V13 durchgeführt. Das TIAPortal ist der Nachfolger der klassischen Step7- Programmieroberfläche. Es hat im Gegensatz zum klassischen Step7 eine gemeinsame Datenhaltung. Dies soll das Arbeiten effektiver gestalten und die Dateneingabe vereinfachen. Die Entwicklungsumgebung für die Inbetriebnahme der Steuerungssysteme ist von den eingesetzten Hardwareständen abhängig. Das TiaPortal unterstützt ausschließlich Komponenten, die vor dem 1.10.2007 im Siemens Hardwarekatalog verfügbar waren. Aus diesem Grund müssen die Steuerungen der CPU Generation S7-1500 mit dem TIA Portal implementiert werden. Die unterschiedliche Datenhaltung in der Programmieroberflächen TIA und das klassische Step7 sind inkompatibel zueinander. Es ist also nicht möglich, eine Migration von Projekten dieser verschiedenen Umgebungen durchzuführen. Das klassische Step7 Programmierpaket ist sozusagen abgekündigt. Es wird keine Weiterentwicklung dieser Software geben. Der Support für Step7 ist nur noch bedingt verfügbar. Ein Grund dafür ist, dass das klassische Step7 in Zukunft nicht mehr zuverlässig auf den neuen Betriebssystemen arbeiteten wird. Das TiaPortal setzt unter anderem auf dem .NET- Framework auf. Während der Implementierung der Systeme und der Arbeit mit diesen beiden Entwicklungsumgebungen wurden folgende Feststellungen gemacht:

In dem TiaPortal gibt es nicht mehr, wie im klassischen Step7, viele unterschiedliche Editoren für die Bearbeitung der Systeme, sondern nur eine Oberfläche für alle Projektbestandteile. Diese Oberfläche ist deshalb sehr überladen und unübersichtlich. Das Bedienkonzept, aus dem klassischen Step7, ist nicht wiederzufinden. Alle Programmteile sind in einer gesamten Projektdatei gespeichert. Stürzt das TiaPortal nach einer längeren Sitzung ab, sind alle Programmänderungen seit dem Start der Sitzung verloren. Das bezieht sich auf alle Programmteile wie Bausteine, Funktionen und Hardwarekonfigurationen.

Die Inbetriebnahmen der Systeme 1 bis 4 wurden mit dem TIA Portal projektiert und die Systeme 5, 17, 18 und 19 mit dem klassischen Step7. Bei der Umsetzung von homogenen Systemen gab es keine Probleme. Bei diesen Systemen wurde zunächst die Hardwarekonfiguration vorgenommen. Danach wurde jeweils der OB1 für den Zyklischen Task programmiert. Der OB1 Baustein ruft dann zyklisch den Funktionsbaustein FB1 auf. Für diesen Funktionsbaustein musste ein Datenbaustein angelegt werden. Es werden bei der Hardwarekonfiguration die Festlegungen aus dem Kapitel 2, Seite 3, umgesetzt. Bei der Konfiguration der Controller gibt es zwischen den beiden Entwicklungsumgebungen unterschiedliche Standardeinstellungen.
Der bei allen Systemen für den Funktionsaufruf verwendete OB1 ist ein freilaufender zyklischer Baustein. Hierbei ist aber zu beachten, dass dies nur in dem klassischen Step7 realisiert ist. Bei dem TiaPortal wird dieser “freilaufende“ OB1 durch eine konfigurierbare Minimalzykluszeit begrenzt. Da in dem Performancevergleich auf allen Systemen ein freilaufender Task realisiert werden soll, wird diese Option deaktiviert. Hierdurch können die Zyklen bei geringer Kommunikationslast schneller durchlaufen werden.

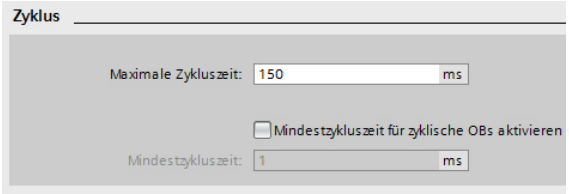


Abbildung 5.1: Mindestzykluszeit

In der TiaPortal Umgebung ist eine Kommunikationslast von 50% standardmäßig festgelegt. Diese wird bei allen Systemen, basierend auf der Ausgangsbetrachtungen (Kapitel 2, Seite 3), auf 20% gesetzt. Die Einstellung bewirkt, das der zeitliche Anteil von Kommunikationsprozessen im Zyklus begrenzt ist.

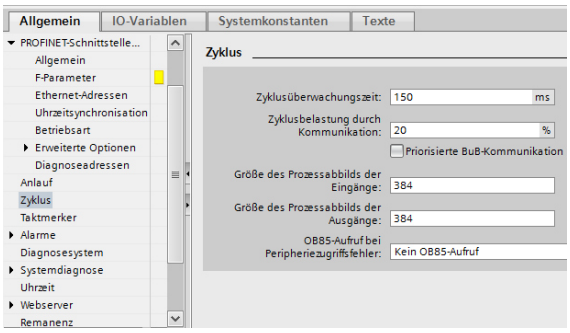


Abbildung 5.2: Kommunikationslasteinstellungen bei Controllern im TiaPortal

Das TiaPortal ist für die symbolische Programmierung ausgelegt und ermöglicht damit für die Controllergeneration S7-1200 und S7-1500 eine effizientere Arbeitsweise. Diese durchgängige symbolische Programmierung schafft einen konsistenten, typsicheren,

schnellen und flexiblen Zugriff auf sämtliche Daten. Es wird somit keine absolute Adressierung verwendet, wie im klassischen Step7. Die CPU stellt die geeignete Datenablage sicher, indem sie die Bausteine abhängig vom jeweiligen Steuerungstyp, Platz- und Geschwindigkeitsabhängig organisiert. Aus diesem Grund gibt es im TiaPortal die Möglichkeit des optimierten Bausteinzugriffs. [40, Vgl.]

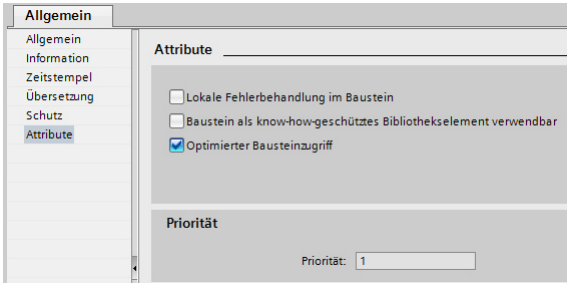


Abbildung 5.3: Optimierter Bausteinzugriff

Die neue Controllergeneration S7-1200/ S7-1500 ist symbolisch optimiert. Die älteren Steuerungsgenerationen sind auf die direkte Adressierung optimiert. Aus diesem Grund wurde nur für die S7-1500 Steuerungen der optimierte Bausteinzugriff verwendet. Auf diese Weise sollen die jeweiligen Steuerungen mit einer höheren Performance arbeiten können. Generell wurde die Programmerstellung mit symbolischer Programmierung auf den einzelnen Systemen realisiert.
Bei der Einbindung der dezentralen Peripherie ist folgende Vorgehensweise umgesetzt worden. Als erstes wurden die Adressen der ET200s-Buskoppler gepackt, um einen byteweisen Zugriff auf die Ausgänge zu ermöglichen. Im Anschluss wurde die Aktualisierungszeit des PROFINET IO Busses konfiguriert. Der Minimalwert der Aktualisierungszeit ist abhängig vom eingestellten Kommunikationsanteil für PROFINET IO, von der Anzahl der IO-Devices und von der Anzahl der projektierten Nutzdaten. Die Aktualisierungszeit für die Buskoppler wird auf eine Millisekunde gesetzt. Aufgrund der High Feature Baugruppen wird dies von den ET200s Buskopplern unterstützt.

Die Realisierung von inhomogenen Netzen ist mit den Siemens Entwicklungsumgebungen schwer oder nur teilweise umsetzbar. Zunächst müssen die einzelnen GSD- Dateien in die Bibliotheken importiert werden. Nach dem Import der Dateien ist in den seltensten Fällen der volle Funktionsumfang, wie bei einer SIEMENS Komponente, gegeben. Beispielsweise sind Funktionen, wie die des Adressen Packens, nicht mehr möglich und müssen per Hand ausgeführt werden. Im konkreten Fall wurden zwei baugleiche E/A Module im Komponentenaufbau integriert. Das erste E/A- Modul reserviert eine Belegung von Bits für digitale Module. Diese reservierte Belegung wird dann mit den Bitmodulen belegt. Das zweite Modul ist für die Bitbelegung konfiguriert. Entsprechend dieser Belegung werden die Elemente dann gepackt angesprochen.
Ein Problem, dass nicht vollständig gelöst werden konnte, ist die Implementierung von PhoenixContact Inline Modulen in die SIEMENS Umgebung. Nach dem Import der ent-

sprechenden Inlinemodule in die Step7 Bibliothek, konnten die Module konfiguriert werden. Jedoch nach dem Überspielen der Parametrierung auf das Zielgerät gibt die Steuerung einen Systemfehler an. Der Diagnosepuffer gab an, dass eine Moduldifferenz zwischen dem Soll- und Istaufbau vorliegt.

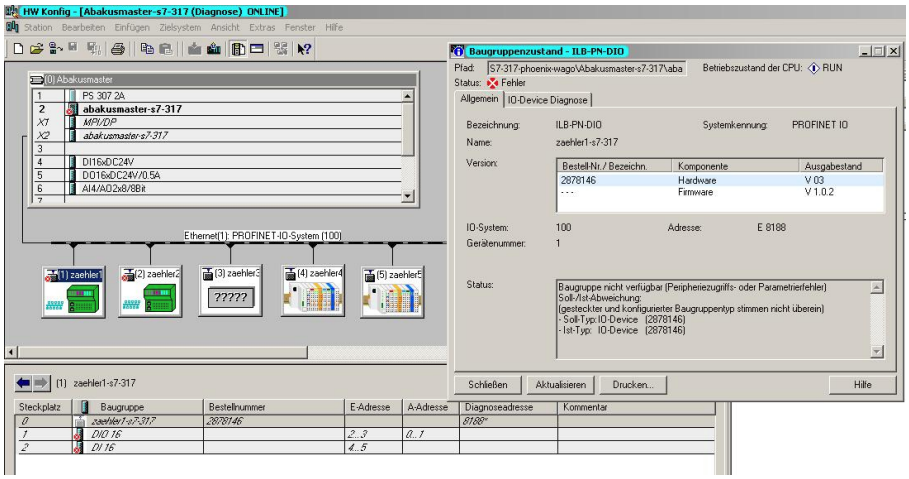


Abbildung 5.4: PhoenixContact Inline Problem in Step7

In der Abbildung 5.4 ist die Moduldifferenz im Anlauf ersichtlich. Die Module Melden demnach keine Fehlermeldung. Die Prozessdatenaktualisierung wird aber von der Steuerung aus gesperrt. Nach Betrachtung des Anlaufs unter Wireshark ist zu sehen, dass der Modudiffblock nur einmalig auftaucht und am Ende der Parametrierung nicht mehr anliegt und auch nicht mehr gemeldet wird. Scheinbar merkt sich aber die Steuerung den ersten Moduldiffbolck, obwohl dieser nicht mehr anliegt. Der Wiresharkmitschnitt für den Anlauf ist in der Anlage B Seite 95 vorzufinden. Der Systemfehler konnte durch das Programmieren des Fehlerbausteins OB86 abgefangen werden. Dieser Baustein wird abgearbeitet, sobald beim Hochlaufen der Steuerung ein Fehler anliegt. Durch diese Umsetzung kann nach dem Anlauf der Systemfehler und somit der CPU STOP zumindest unterdrückt werden. Das Programm wird zwar abgearbeitet, jedoch bleibt die Prozessdatenausgabe an den Inlinemodulen gesperrt.

Ein weiteres Problem bei den Fremdbuskopplern ist, dass diese nur mit einer 1 Port Schnittstelle deklariert werden, obwohl sie einen dual Port Switch besitzen. Dies ist besonders bei dem sehr objektorientierten TiaPortal ein Problem. Man konnte das Problem nur lösen, indem man die Buskoppler nur dem Mastergerät und nicht in der Topologie zuordnet.

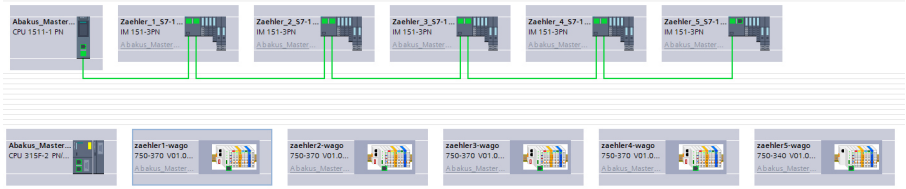


Abbildung 5.5: Implementierung von inhomogenen Komponenten

Die Abbildung 5.5 zeigt zum einen die SIEMENS ET200s und zum anderen die Komponenten “Wago 750-370“ in der Topologieansicht. Hier ist zu erkennen, dass beide Systeme einen dual Port Switch besitzen, obwohl bei den Wogo Komponenten nur ein Singleport implementiert wird.
Grundlegend ist festzustellen, dass die Implementierung von systemeigenen Siemens-Komponenten in die Entwicklungsumgebung gut realisierbar ist. Es ist hierbei aber zu beachten, dass die Standarteinstellungen zwischen dem klassischen Step7 und dem TiaPortal sehr unterschiedlich sind. Die Implementierung von systemfremden Herstellern, wie beispielsweise WAGO oder PhoenixContact, ist in die SIEMENS Umgebung nur eingeschränkt und mit intensivem Arbeitsaufwand möglich. Desweiteren ist eine enorme Umorientierung zwischen dem TiaPortal und dem klassischen Step7 nötig.

5.2 PCworx Implementierung

Für das System 7, 8, 9 und 11 wurde die Entwicklungsumgebung PcWorx des Herstellers PhoenixContact verwendet. PCworx ist eine Umgebung für die Feldbuskonfiguration, Diagnose und Programmierung nach IEC 61131 von PhonixContact Geräten. Diese Software wurde in drei grundlegende Arbeitsbereiche unterteilt. Die Arbeitsbereiche sind die IEC- Programmieroberfläche, Buskonfiguration und Prozessdatenzuordnung der anzusteuernden E/A-Module. Die Prozessoren dieser Steuerungen unterstützen eine Multitasking-Struktur mit globalen und lokalen Variablenzugriff. Dieses wird über die Programmieroberfläche PCWorx umgesetzt. Der Programmcode wird in den 5 konformen IEC 61131 Programmiersprachen erstellt. In der Buskonfigurationsoberfläche dient der Parametrierung und Auswahl des verwendeten Bussystems. In den Testumgebungen wurden die Systeme in PcWorks mit ProfiNet und ModbusTCP implementiert. Neben der Konfiguration des Bussystems werden in dieser Arbeitsfläche auch die Teilnehmer dieses Netzwerks konfiguriert. Bei ProfiNet-Netzwerken werden hier die DeviceNet- Namen vergeben, sowie die Zuordnung der IP/Mac-Adressen und die Lokalbusteilnehmer konfiguriert. Bei Modbus-Netzwerken erfolgt hier die Festlegung der Kommunikationsregister.

Funktionscode	Funktion	Beschreibung
FC3	Read holding registers	Lesen von Wörtern für Aus- und Eingänge
FC4	Read input registers	Lesen von Wörtern von Eingängen
FC6	Write single registers	Schreiben eines Worts für Ausgangsdaten
FC16	Write multiple registers	Schreiben mehrerer Ausgangsworte
FC23	Read/write multiple registers	Lesen und Schreiben mehrerer Prozessdaten für Ein- und Ausgänge

Abbildung 5.6: Unterstützte MODBUS- Funktionen

Die Abbildung 5.6 zeigt die unterstützten Modbusfunktionen der Geräte unter PcWorx. Es ist erforderlich, für den lesenden und schreibenden Zugriff im Ethernet- Netzwerk, spezielle Funktionsregister zu definieren. In der Arbeit wurde ausschließlich der Modbus Funktioncode FC16-WRITE MULTIPLE REGISTER verwendet.

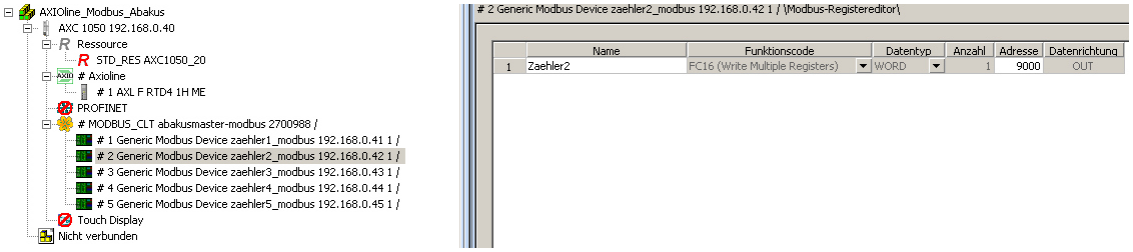


Abbildung 5.7: Modbusregistereditor

Neben dem Funktionscode muss auch die Registeradresse für die Ausgangsprozessdaten angegeben werden. Sie ist bei den Buskopplern ab Register 9000 abgebildet. Auf der Abbildung 5.7 ist zu erkennen, dass auf den Modulkatalog für die Buskonfiguration kein spezifischer Buskoppler verwendet wird, sondern ein Allgemeines Modbus- Gerät. In dem dritten Arbeitsbereich wird die Prozessdatenzuordnung hergestellt. Hierbei werden die Prozessdaten der Buskoppler mit den Variablen verheiratet.
Bei der Realisierung von homogenen Systemen gab es keine Schwierigkeiten. Es mussten aber Einarbeitungen in der Registerbeschreibung unter Modbus vorgenommen werden. Probleme gab es hinsichtlich der Versionsverwaltung unter PcWorx. Es wurde bei der Inline- Steuerung nur die neueste Version unterstützt. Erst nach einem Update der Steuerung über FTP war die Inbetriebnahme über PcWorks möglich.
Hinsichtlich der Inbetriebnahme der Axiolinesteuerung gab es Defizite in der Aktualisierungszeit der Buskoppler. Die Axioline-SPS hat eine CPU für kleine bis mittlere Automatisierungsapplikationen mit bis zu 8 Peripheriebaugruppen. Im Datenblatt ist eine Aktualisierungszeit von 4ms angegeben. Dies war aber schon ab 2 deklarierten Buskopplern nicht mehr gegeben. Bei einer zu klein konfigurierten Aktualisierungszeit meldet die Axioline-Steuerung einen IO Fehler. Sie schafft es nicht, mehr Datenmenge zu übertragen. Das Problem hierbei sind die fehlenden Systemressourcen des Controllers für die

benötigte Aktualisierung der Buskoppler. Dieses Problem wird dadurch noch verstärkt, dass der Controller bei jedem Buskoppler von der maximalen Größe des Prozessabbildes ausgeht. Die Größe des Prozessabbildes ist in PcWorx nicht konfigurierbar. Durch diesen Fakt gehen Leistungsreserven des Controllers verloren. Letztendlich wurde bei allen 5 Buskopplern, die mit der Axiolinesteuerung realisiert wurden, eine Aktualisierungszeit von 16ms deklariert. Der Controller ILC350 ist eine wesentlich leistungsstärkere Steuerungseinheit. Diese CPU unterstützt Aktualisierungszeiten bis zu 1ms für den Feldbus.
Für die Registerbeschreibung der Modbus-Module waren größere Konfigurationsumfänge notwendig. Die Modbus Buskoppler haben einen sogenannten Plug&Play Modus. Dieser Modus ermöglicht eine Auto-Konfiguration des Lokalbusses ohne Konfiguration mit Hilfe der Parametrierungssoftware. Dieser Modus ist werkseitig aktiv. Dadurch wird das Schreiben und Lesen der Prozessdaten vom Buskoppler gesperrt. Für die Deaktivierung des Plug& Play Modus muss das Kommandoregister beschrieben werden. Ein weiterer Fakt, der bei der Inbetriebnahme zu beachten ist, ist das die Buskoppler werksseitig ein Eingangsmodul deklariert haben. Um an den Buskoppler ein I/O Modul für die Prozessdatenausgabe zu konfigurieren, muss zunächst das Register mit der Adresse 9000 und dem Funktionscode FC16 mit Null beschrieben werden. Im Anschluss daran wird das Register 2006 mit 0008hex beschrieben. Erst nach dieser Vorgehensweise ist das I/O Modul für den Buskoppler verfügbar.

Die Realisierung von inhomogenen Systemen unter PcWorx war die Schwierigkeit, das deklarieren von Byte Modulen und die Entsprechende Prozessdatenzuordnung. Es wurden in PcWorx Siemens ET200s Buskoppler eingebunden. Die Funktionalität des Adressen packens wurde seitens PCWorks nicht unterstützt. Deshalb wurden wiederum zwei verschiedene Zuordnungen im Komponentenaufbau der Prozessdaten getätigt. Das eine Modul realisiert hierbei eine Belegung vom Type Byte. Diese reservierte Byte- Belegung wird dann mit Reservierung vom Typ Bit belegt. Bei beiden Modulen handelt es sich um die digitalen Ausgangskanäle 132-4BB01. Die Abbildung 5.8 stellt diese Konfiguration und die entsprechende Prozessdatenzuordnung dar.

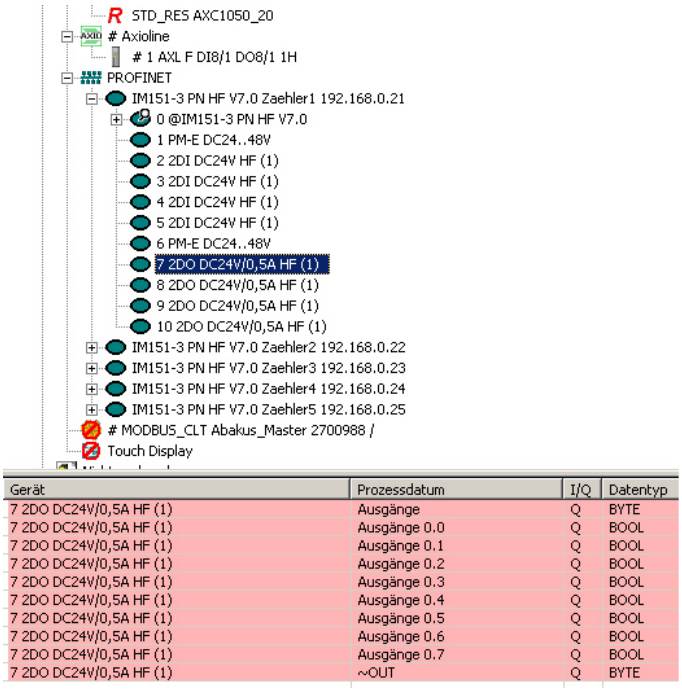


Abbildung 5.8: Prozessdatenzuordnung

Die Implementierung unter PcWorx ist nach einer gewissen Einarbeitung sehr logisch und komfortabel aufgebaut. Dieses Programm läuft unter Win7 und WinXP sehr stabil. Die Copy& Paste Funktion und das Migrieren von unterschiedlichen Projekten ist ohne Probleme möglich. Etwas schwieriger ist die Inbetriebnahme der Modbus-Buskoppler. Nach der Konfiguration dieser I/O Devices ist das Schreiben und Lesen der Prozessdaten über die einzelnen Register schnell deklariert.

5.3 Unity Implementierung

Diese Programmiersoftware nach IEC 61131-3 wird für die Steuerungen Modicon M340, Premium, Atrium und Quantum eingesetzt. Für die Konfiguration, Programmierung und Projektierung des Systems 10 wurde diese Automatisierungssoftware verwendet. Unity unterstützt alle IEC 61131 Sprachen und Strukturen. Ein weiteres Feature ist eine Multitasking Struktur für alle Steuerungstypen.

Für die Implementierung des dekadischen Zählers unter dem Schneidersystem müssen folgende Bereiche bearbeitet werden:

- Buskonfiguration
- Variablen und FB- Instanzen
- Kommunikation
- Programm

In der Hardwarekonfiguaration des Steuerungssystems werden der Bus, das Rack, die E/A-Module, die Speicherbaugruppe und die Zykluszeiten des Prozessors parametriert. Als erstes werden sämtliche Hardwarewarekomponenten in der Aufbauansicht der Steuerung eingefügt und konfiguriert.

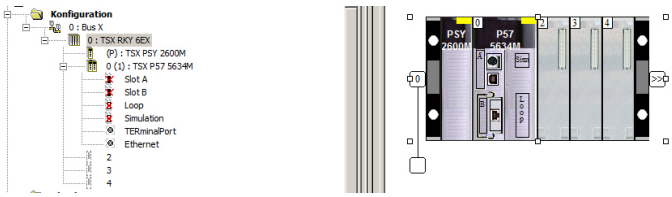


Abbildung 5.9: Modicon TSX Busaufbau

Für die Feldbus-Kommunikation muss dem Ethernet-Controller ein Kanal zugeordnet werden. Um den Kanal der Modicon TSX für die Kommunikation über Ethernet benutzen zu können, müssen zunächst die Konfigurationsparameter für die Nachrichtenübertragung eingestellt werden. Der Kanal definiert die Funktion und die Netzwerkverbindung für den Netzwerktyp Modbus/TCP “ETH TCP IP“. Des weiteren muss dem Kanal eine Netzwerkverbindung zugeordnet werden. Hierfür wird zunächst im Bereich Kommunikation ein Ethernet TCP Netzwerk angelegt. In diesem Netzwerk wird der Kommunikationskanal deklariert und die erforderlichen Ethernet Port Parameter konfiguriert. Dazu zählen die Ethernet Parametrierung des Mastergeräts sowie die Implementierung aller dezentralen E/A-Peripheriegeräte. In dem vorhandenen Adressserver werden alle Turck- und MOMENTUM Buskoppler eingetragen. Diese Client/ Server Adresstabelle verwaltet alle dezentralen Stationen, die der DHCP Server für den Anlauf benötigt. Der Server vergibt im Anlauf anhand der MAC Adresse dem Buskopplern die IP- Adresse und die ihm zugeordnete Subnetzmaske.

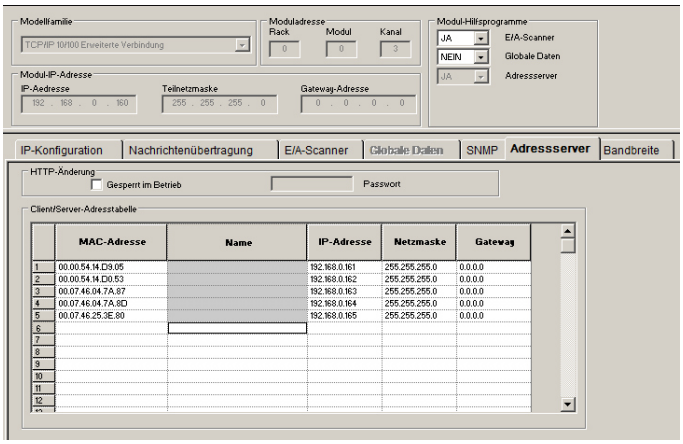


Abbildung 5.10: Client/Server Adresstabelle

Für die Übertragung von Daten zwischen den Master und Slave Geräten aus der Adresstabelle muss die E/A- Abfrage konfiguriert werden. In dieser Tabelle werden anhand

der IP Adressen die Prozessdaten zugeordnet. Entscheidend ist hierbei, dass die Quelladresse der Master- PLC in dem Write Master Objekt implementiert wird. Die Merkerwort-Adresse gibt an, wohin die Daten des jeweiligen dezentralen Gerätes geschrieben werden sollen. Schreibvorgänge werden hierbei immer auf der Wortebene ausgeführt. Die Anzahl der zuschreibenden Worte wird in der Spalte WR-Länge vorgegeben.

IP-Konfiguration | Nachrichtenübertragung | E/A-Scanner | Globale Daten | SNMP | Adressserver | Bandbreite

Gerätesteuerungsblock (%MW): vor bis Wiederholungsratenschritt: 10

Gescannte Peripheriegeräte

	IP-Adresse	Gerätename	Einheit-ID	Slave-Syntax	Funktionsfähigkeit-Timeout(ms	Wiederholungsrate (ms)	RD Master-Objekt	RD Slave-Ref	RD Länge	Letzter Wert (Eingang)	WR Master-Objekt	WR Slave-Ref	WR Länge	Gateway/ Bridge-Gerät	Beschreibung
1	192.168.0.161		255	Index	1500	10	%MW0	0	1	Letzten Wert beibe	%MW10	0	1		MOMENTUM
2	192.168.0.162		255	Index	1500	10	%MW2	0	1	Letzten Wert beibe	%MW12	0	1		MOMENTUM
3	192.168.0.163		255	Index	1500	10	%MW4	0	1	Letzten Wert beibe	%MW14	0	1		TURCK BL20
4	192.168.0.164		255	Index	1500	10	%MW6	0	1	Letzten Wert beibe	%MW16	0	1		TURCK BL20
5	192.168.0.165		255	Index	1500	10	%MW8	0	1	Letzten Wert beibe	%MW18	0	1		TURCK FEN20

Abbildung 5.11: Prozessdatenzuordnung über E/A Scanner

Zu beachten ist, dass unter Unity sämtliche Konfigurationen validiert werden müssen. Dies wird über den Button in der Symbolleiste umgesetzt. Der abschließende Schritt der Implementierung des Kommunikationsnetzwerks ist die logische Zuweisung des Ethernetkanals zum Controller. Nach dieser Zuordnung wird das logische Netzwerk aktiviert und die Rack-, Modul- und Kanalnummer im Konfigurationsbildschirm des logischen Netzwerks aktualisiert.
Nach der Implementierung des Netzwerks erfolgt die Variablendeklaration und die Programmerstellung. Die Multitasking-Struktur der Steuerung ermöglicht eine Unterscheidung zwischen MAST- und FAST-Task. Bei der Umsetzung dieser Arbeit wurde eine FAST Task verwendet. Sie ist eine periodische Task mit hoher Priorität und sehr kurzer Laufzeit. Das erstellte Projekt musste vor der Übertragung generiert werden und im Anschluss per USB auf die Steuerung geladen werden.
Die Programmierung in Unity zeichnet sich durch eine sehr praktische, offene und effektive Implementierung aus. Es kann sehr einfach ein Netzwerk konfiguriert werden. Über die Client/ Server Adresstabelle und dem E/A Scanner kann mithilfe der MAC Adressen schnell ein solches Netzwerk aufgebaut werden. Auch herstellerneutrale Systeme, wie die TURCK Buskoppler, können problemlos eingebunden werden.

5.4 CodeSys Implementierung

Die Programmierung der TURCK VT250 Module wurde mit Codesys V3 durchgeführt. Die der Visualisierung mit Hilfe der Softwareoberfläche QVIS. Mit CodeSys können Steuerungsapplikationen für unterschiedlichste herstellerspezifische Hardwarekomponenten in Betrieb genommen werden. Die Version 3 vereint die Funktionalität der Version 2 mit den Möglichkeiten neuster Technologien für objektorientierte SPS- Programmierung.
Für die Konfiguration, Programmierung und Projektierung der Systeme 6 und 12 stellt

TURCK eine modifizierte CodeSys Version “CodeSys V3.5 SP1Hotfix1“ zur Verfügung. TURCK empfiehlt, vor der Erstellung eines Projektes die vordefinierten Feature Sets von “Standard“ auf “Professional“ zu aktivieren. Unter Professional ist die Benutzeroberfläche komplexer und die Möglichkeiten der Systemkonfiguration sind größer. Bei der Erstellung des CodeSys Projektes muss das TURCK Gerät und die jeweilige Programmiersprache gewählt werden. Die Koordinierung der Benutzeroberfläche findet über einen Projektbaum statt.
Für die Kommunikation zwischen Gerät und Programmiersystem muss zunächst ein Gateway angelegt werden. Hierbei wird das lokale CodeSys Kommunikations- Gateway des Programmierrechners verwendet. Diesem Gateway muss das TURCK Mastergerät zugeordnet werden. Nachdem das Gateway konfiguriert wurde, wird dieses automatisch gestartet. Über die Knotenadresse steht jetzt eine Verbindung von Programmiergerät zum Mastergerät zur Verfügung. Der TURCK-Testaufbau für die Diplomarbeit wurde kommunikationstechnisch mit Modbus/TCP und ProfiNet angesteuert. Für die jeweiligen Ansteuerungen wurden zwei unterschiedliche Ethernetadapter verwendet.

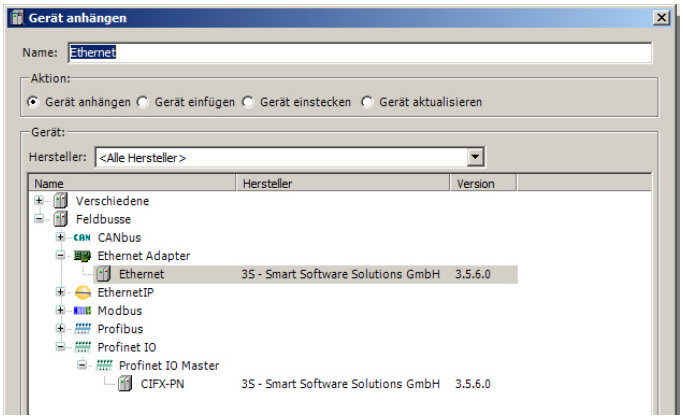


Abbildung 5.12: Ethernetadapter für ProfiNet oder Modbus

Unter ProfiNet wird ein ProfiNet IO Master von netX als Gerät konfiguriert. Für diesen Adapter muss die IP Adresse für den ProfiNet Master parametriert werden. Die IP- Adressen der Geräte des Netzwerks werden über das TURCK IP- Adress- Tool vergeben. Die IP- Adressen müssen so konfiguriert werden, das sie mit der tatsächlichen Adresse des Geräts übereinstimmen. Nachdem der ProfiNet Ethernet Adapter konfiguriert ist, kann im ProfiNet Netzwerk nach Teilnehmern gesucht werden. Hierbei ist es wichtig, dass nur dann die ProfiNet-Slaves bei einem Scan gefunden werden, wenn die herstellerspezifischen GSDML Dateien in die Geräte Repository von CodeSys importiert wurden. Zusätzlich zu der IP-Adressierung wird noch ein Gerätenamen vergeben, da ProfiNet-Geräte nicht über die IP-Adresse identifiziert werden, sondern über ihren Gerätenamen. In dem Gerätescann Fenster kann diese Eigenschaft für die einzelnen Busteilnehmer konfiguriert werden. Bei nicht Erkennen von Busteilnehmern im Scan-Zyklus können diese auch manuell eingefügt werden. Nachdem die Geräte pa-

rametriert sind, werden diese in das Projekt implementiert. Diese Geräte sind nun im Projekt eingebunden und können mit den entsprechenden Modulen bestückt werden. Dies geschieht über die Funktion “Gerät Anhängen“. Es wird folgender ProfiNet Netzwerkaufbau realisiert.

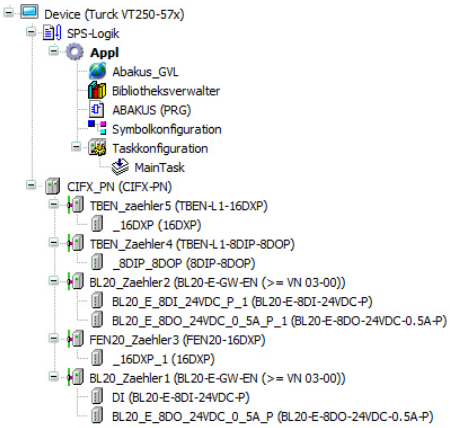


Abbildung 5.13: VT250– ProfiNet Netzwerk

Für die Ansteuerung über ein Modbus-Netzwerks muss ein allgemeiner Ethernet Adapter in die Steuerungskonfiguration hinzugefügt werden. Diesem Adapter wird die Modbus-Masterfunktionalität zugewiesen und anschließend alle Slave-Geräte zugeordnet. Dem Buskopplern kann nach dem Anhängen an das Mastergerät die IP- Adresse, sowie der Modbusport “502“ zugeordnet werden.

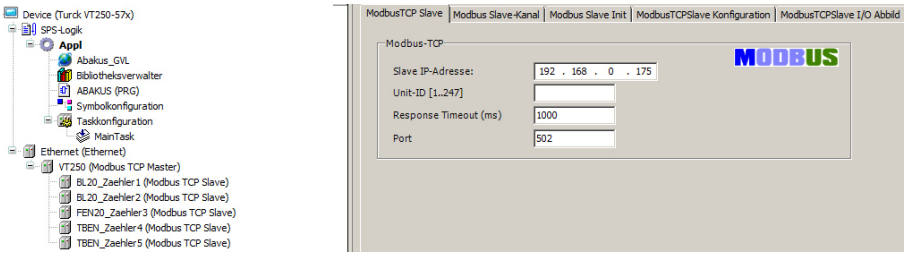


Abbildung 5.14: VT250– Modbus Netzwerk

Die Kommunikation zwischen dem Modbus TCP Master und dem Modbus Slaves erfolgt über definierte Modbuskanäle. Hierfür wird der Write Modbus Funktionscode 6 verwendet. Als Write Register wird das Register 0x0800 verwendet. Diese Einstellung wird in den jeweiligen Buskopplern des Modbus-Slave Kanals vorgenommen.

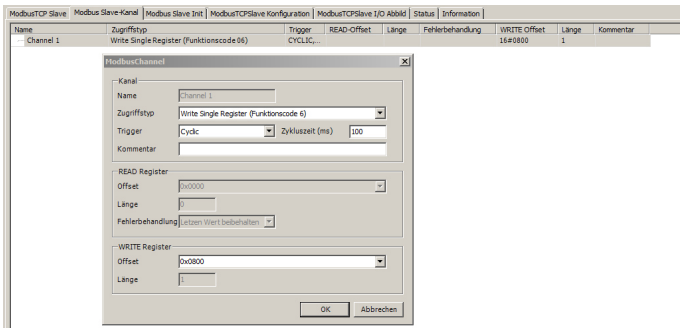


Abbildung 5.15: Buskoppler– Modbus Slave-Kanal

Das Mappen des Zählerwertes auf das Ausgangsregister des Buskopplers wird im ModbusTCP Slave I/O Abbild umgesetzt.

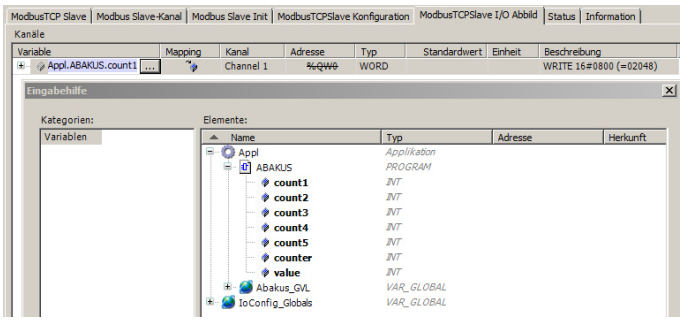


Abbildung 5.16: ModbusTCP Slave I/O Abbild

Nachdem die Konfiguration des Netzwerks abgeschlossen ist, wird die Programmierung in der Steuerung umgesetzt. Generell ist die Implementierung des Programmcodes bei beiden Applikationen gleich. Sie unterscheiden sich ausschließlich hinsichtlich der kommunikationstechnischen Ansteuerung über Modbus oder ProfiNet. Hierfür wurden im Projektbaum unter der Applikation die benötigten Variablen deklariert und danach der Programmcode erstellt. Das Programm wurde einer freilaufenden Task zugeordnet. Bei der Variablendeklaration müssen die Variablen in der Symbolkonfiguration freigeben werden. Wird dies nicht realisiert, kann auf die Variablen über den OPC Server oder die Visualisierung nicht zugegriffen werden. Die Visualisierung für die VT250 Panel SPS wird über die Software QVIS von HILSCHER programmiert. Die programmierte Visualisierung für die Panel SPS hat jedoch keine Bedeutung auf den Performancevergleich. Aus diesem Grund wurde sie in diesem Kapitel nicht näher betrachtet. Eine Abbildung der Visualisierung ist im Anhang C Seite 96 dargestellt.

Bei der Implementierung der TURCK Systeme ist von Vorteil, dass CodeSys eine sehr etablierte Programmierumgebung ist. Es ist hierbei darauf zu achten, dass die jeweiligen hardwarespezifischen Treiber von TURCK und HILSCHER in die Programmieroberfläche eingebunden sind. Visualisierungen können einfach über QVIS erstellt werden, da

dies eine sehr übersichtliche Software ist.

5.5 IndraWorks Implementierung

Die Implementierung des BoschRexroth Systems wurde mit Hilfe der Entwicklungsumgebung IndraWorks umgesetzt. IndraWorks ist das Programmiersystem, in welches sich die BoschRexroth Engineering Tools integrieren. Es ist für die SPS- und Motion-Logic- Programmierung auf Basis von CoDeSys V3 mit objektorientierte Spracherweiterung geeignet. Hinzu kommt eine umfangreiche Softwarebibliothek, die konform zu IEC 61131-3 und PLCopen ist. IndraWorks bietet somit eine durchgängige Bedienung auf Basis aktueller Windows Technologien. Es stehen umfangreiche Wizards für die Projektierung von Steuerungen, Antrieben und deren Peripherie zur Verfügung. Basis für die Entwicklungsumgebung ist CoDeSys. Aus diesem Grund ist der Aufbau und die Bedienung sehr ähnlich zum Standard Softwaretool der Firma 3S-Smart Software Resolutions. Die Bedienung ist, wie in CoDeSys üblich, an dem Projektbaum orientiert.

Bei der Implementierung des Projektes ist es sehr wichtig, dass die richtige Firmware-Version und das entsprechende Firmware-Release der Steuerung projektiert ist. Bei dieser Arbeit wurde ausschließlich die Sercos Schnittstelle verwendet. Nach der Implementierung der Steuerung in das Projekt wird automatisch die Sercos Schnittstelle eingefügt und aktiviert. Über die Funktion “Buskonfiguration scannen“ können automatisch die Buskoppler des Sercos-Rings hinzugefügt werden. Manuell ist dieses über die Gerätebibliothek möglich. Hierfür müssen die Axioline Buskoppler in Betrieb und in der Ringstruktur integriert sein. Desweiteren müssen die auf XML basierenden SDDML- Gerätebeschreibungsdateien in die IndraWorks Bibliotheks Respository eingebunden sein. Die deklarierte Busstruktur muss der realen Ringstruktur entsprechen. Diese Struktur kann in der Sercos I/O Diagnose angepasst werden. Den eingebundenen Buskopplern muss für die Prozessdatenausgabe ein I/O Modul zugeordnet werden. Diese Module befinden sich, nach der Installation der SDDML- Datei, in der IndraWorks Bibliothek. Nach dieser Implementierung ergibt sich folgende Projektstruktur:

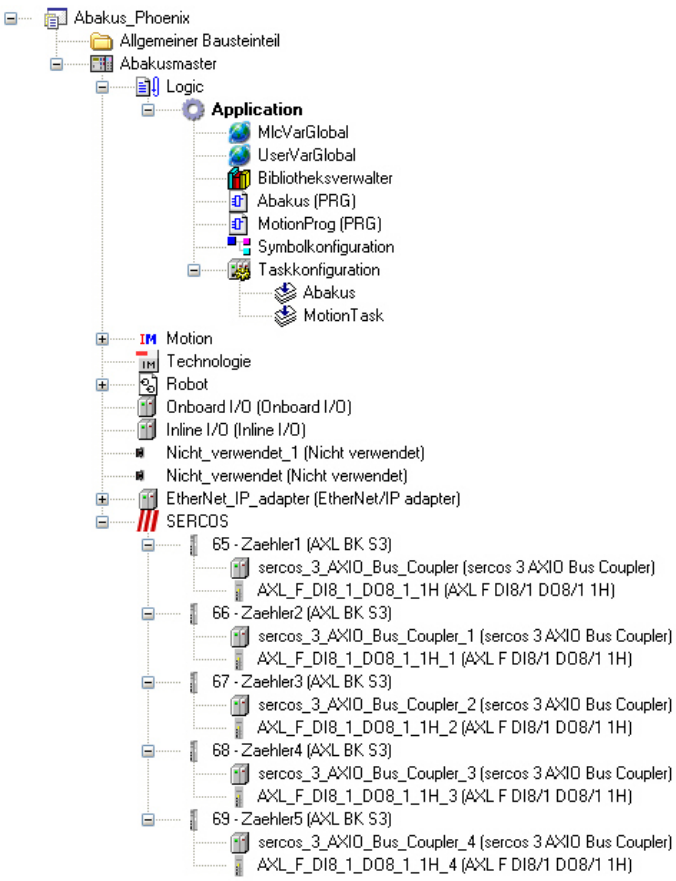


Abbildung 5.17: IndraWorks Projektstruktur

Die Umsetzung der Programmstruktur erfolgt nach CoDeSys üblichen Methoden. Die Bezugsadressen für die Prozessdatenausgabe auf den Buskopplern ist im Sercos III Modul I/O Abbild beschrieben. Diese Adressen werden den Programmvariablen in der Variablentabelle zugeordnet. Durch einen Download wird das programmierte Projekt auf der Steuerung aktiviert.

Die Implementierung des BoschRexroth Systems gestaltete sich am Anfang schwierig. Die Probleme traten bei der Einbindung der Steuerung auf. Ursache hierfür waren die unterschiedlichen Firmwarestände der steuerungstechnischen Hardwarekomponenten. Nach einem Firmwareupdate konnten diese jedoch problemlos in die IndraWorks Umgebung eingefügt werden. Die ähnlichen Softwarestukturen von CoDeSys und IndraWorks wirkten sich sehr vorteilhaft auf die Inbetriebnahme aus.

5.6 TwinCat Implementierung

Die Automatisierungssoftware TwinCAT der Firma Beckhoff ist für SPS-, CNC-, und MotionControl Anwendungen gedacht. Mit dieser Softwareumgebung können PC basierte Steuerungssysteme als Echtzeitsteuerungen mit Multi-SPS-Systeme, Achsregelun-

gen, Programmierumgebungen und Bedienstationen verwendet werden. TwinCAT besteht aus Laufzeitsystemen zur Echtzeitausführung von Steuerungsprogrammen und den Entwicklungsumgebungen zur Programmierung, Diagnose und Konfiguration. Die in dieser Arbeit benutzte Version TwinCAT 3 wurde von Beckhoff in die Entwicklungsumgebung Microsoft VisualStudio integriert. [46, Vgl.] In dieser Entwicklungsumgebung sind alle wesentlichen Programmiersprachen, wie C/C++, die Sprachen aus der IEC 61131 (ST, AWL, AS, FUP, KOP), inklusive der objektorientierten Erweiterungen, sowie eine Matlab/ Simulink Anbindung integriert. [44, Vgl. S. 298-299]
Um den PC als Soft-SPS Betrieb zu verwenden, muss die Netzwerkkarte als TwinCAT Schnittstelle konfiguriert werden. Hierfür muss der von TwinCat mitgelieferte Netzwerktreiber installiert werden. Nach erfolgreicher Installation des Treibers kann die EtherCAT Hardware mit PC-System kommunizieren.
Für die Installation des Treibers ist zu beachten, dass BECKHOFF nur bestimmte Netzwerkkarten unterstützt.

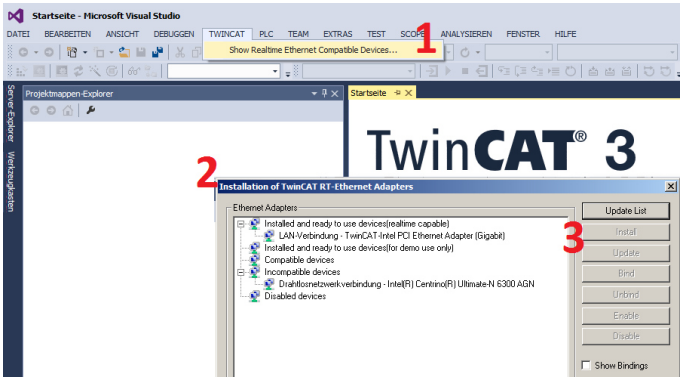


Abbildung 5.18: Compatible TwinCAT RT-Ethernet Adapter

Die Busstruktur kann in TwinCAT zum einen automatisch und zum anderen manuell eingelesen werden. In diesem Projekt wurde die Busstruktur automatisch eingelesen. Um das EVA-Kit und die Buskoppler für EtherCAT verwenden zu können, müssen die *.XML Dateien der Geräte in TwinCAT importiert werden. Diese Dateien beschreiben die Funktionen der EtherCAT Hardware für TwinCAT. Die erfolgreich eingelesenen Module werden in der Reihenfolge, wie sie physikalisch angeschlossen sind, in die EtherCAT Struktur integriert. Es ergibt sich folgende Busstruktur. Als Boxen werden die Axioline Module erkannt und der FB1130 Controller.

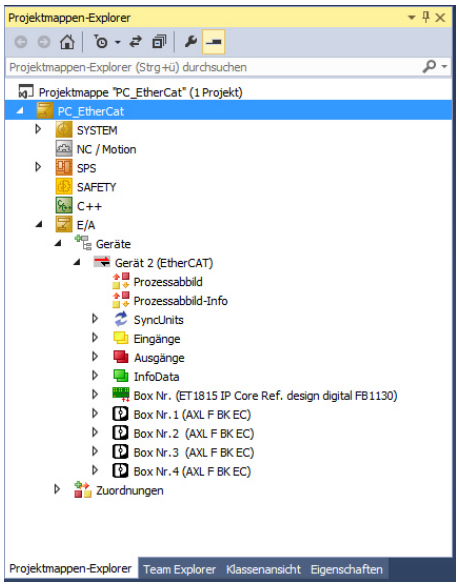


Abbildung 5.19: Busstruktur

Nachdem die Busstruktur eingelesen wurde, kann der 'Free RUN Mode' aktiviert werden. In diesem Modus kommuniziert der PC über die Netzwerkkarte mit dem EtherCAT Teilnehmer und kann dadurch direkt Prozessdaten austauschen. Nach Aktivierung des Modus werden die EtherCAT Geräte in den Freilauf-Modus versetzt. E/A-Variablen von Busklemmen können danach auch ohne konfigurierter und aktivierter Task (z.B. ohne SPS-Projekt) geschrieben werden. [46, Vgl.] Die eindeutige Identifikation eines Geräts erfolgt in über:

- NetId
- Port

Die AMS Net Id Adresse ist die Adresse des lokalen Rechners im TwinCat Netwerk. Die 'AMS Net ID' wird aus der IP Adresse des Systems + '1.1' generiert. [47, Vgl.] Nachdem die Busstruktur implementiert ist, muss das bestehende Projekt durch ein SPS Projekt erweitert werden. Dabei wird der Explorer Baum um das angelegte SPS Projekt erweitert. Dieses hat für die einzelnen Anwendungsgebiete wie Variablenlisten (GVLs), Visualisierungen (VISUs) und Programme (POUs) einzelne Ordner. [47, Vgl.] Für das Umsetzen des dekadischen Zählers werden unter POUs das FUP Programm und unter GVLs die entsprechende Variablenliste erstellt. Nachdem der Programmcode und die Variablenliste erstellt wurden, kann eine Taskreferenz deklariert werden. Hierfür wird dem SPS Projekt ein Task zugewiesen. Nachdem der Task für die entsprechenden Programme erstellt wurde, kann das SPS Modul kompiliert werden. Die erstellte Instanz enthält das Prozessabbild des SPS Projekts mitsamt aller deklarierten Ein- und Ausgänge.
Die Variablenliste muss mit den Modulen der Busstruktur verknüpft werden. Die in dem SPS Programm deklarierten globalen Variablen lassen sich mit der Hardware in der

Busstruktur verknüpfen. [45, Vgl.] Um mehre Variablen z.B ein Byte mit dem entsprechenden Prozessdatenbyte zu verknüpfen, müssen die Variablen markiert werden und mittels "*Multiverknüpfung ändern...*" verbunden werden. [47, Vgl.] Nachdem nun die Busstruktur deklariert wurde und der Programmcode erstellt ist, kann aus dem TwinCat Konfig Modus in den Run- Modus gewechselt werden. Hierbei wird die Soft SPS gestartet und das Projekt übertragen. Durch Einloggen auf die Steuerung kann das Programm gestartet werden.
Die Entwicklungsumgebung VisualStudio von Microsoft ist eine sehr weit verbreitete Programmiersoftware. Durch die Integration von TwinCat3 in VisualStudio spricht BECKHOFF somit viele Nutzer an. VisualStudio ist ein sehr mächtiges Programm, das eine Vielzahl von Hochsprachen unterstützt. Diese Softwareplattform ermöglicht somit die Programmierung von komplexen Technologie Bausteinen für die BECKHOFF IPC Geräte. Es ist zu beachten, dass von TwinCat nur bestimmte NetzwerkController unterstützt werden.

5.7 AutomationStudio Implementierung

Der B&R AutomationStudio ist eine Entwicklungsumgebung für die Implementierung von Steuerungen, Antrieben, Kommunikationssystemen und Visualisierungen. Das AutomationStudio erlaubt die Programmerstellung in allen SPS- Sprachen der IEC 61131-3, in CFC und C, sowie objektorientiertes Programmieren in C++. Das AutomationStudio unterstützt zahlreiche Schnittstellen zu Fremdsystemen. Dazu zählen zu Beispiel eine XML offene Projektablage, PLCopen, Simulink über MATLAB Interface, SQL Interface und eine Vielzahl von Feldbussen. Das AutomationStudio beinhaltet das Komplettportfolio sämtlicher Softwarepakete der Firma B&R. Es gibt somit keine weiteren Extrapakete, wie beispielsweise eine direkte Anbindung an die Leitebene über OPC. Alle Pakete sind in der Entwicklungsumgebung integriert. [41, Vgl.]

Mit dieser Entwicklungsumgebung wurde das System 15 (PowerPanel 65) und das System 16 (X20CP) implementiert. Das zentrale Fenster im AutomationStudio ist in drei Reiter unterteilt. Die Implementierung wird über "Logical View", "Physical View" und "Configuration View" gegliedert. Der "Logical View" enthält alle Programmteile und Variablendeklarationen. Im "Physikal View" befindet sich die gesamte Hardware, die einer CPU zugeordnet ist. Der "Configuration View" spiegelt alle Beziehungen der Hardware, wie I/O Mapping, wider. Die Implementierung des Systems wird zunächst über das Projekt Wizard erstellt. Über den Wizard wird dem Projekt ein Namen und Projektpfad vergeben und darauffolgend eine Steuerung zugeordnet. Bei der Projekterstellung ist darauf zu achten, dass die Hardware als neue Konfiguration angelegt wird. Nach der Beendigung des Projektwizard werden die Objekte dem Projekt hinzugefügt. In der Diplomarbeit wird dem Projekt ein Programm Objekt, Visualisierungsobjekt und ein OPC Objekt hinzugefügt. Bei der Deklaration des Programmobjektes ist es empfehlenswert, eine "Data type declaration" und eine "Variable declaration" mit anzulegen. Die Optio-

nen “Init programm“ und “Merge init, cyclic and exit programm in to one file“ sollten mit ausgewählt werden, damit das Programm in der Programmierumgebung unter “Logical View“ als eine Datei erscheint und nicht in Segmente wie Initialisierungsdatei, Programmzyklus und Programmabschluss untergliedert wird. Das deklarierte Programm wird nach der Einbindung der CPU zugeordnet.
In der Programmieroberfläche von B&R Systemen wurde das Programm für den dekadischen Zähler in StructedText (ST) geschrieben. Die Umsetzung in FUP war nicht im AutomationStudio möglich, da hierfür die Weiterschaltungsbedingungen EN/ ENQ an den Funktionsblöcken fehlten. Funktionell unterscheidet sich jedoch das programmierte ST- Programm nicht von den anderen Testaufbauten. Dieses Programm läuft in einer schnellen Taskklasse mit der höchsten Priorität.
Nachdem der softwareseitige Aufbau implementiert wurde, wird im “Physical View“ der hardwareseitige Aufbau umgesetzt. In der Ansicht wird die konfigurierte CPU mit den entsprechen Schnittstellen dargestellt. Zur Anbindung der X20 Buskoppler muss in der POWERLINK Ansicht dem POWERLINK Modul die X20 Buskoppler angehängt werden. Die Einbindung der Buskoppler erfolgte über die Vergabe der Knotennummer. Den Buskopplern wird dann über den Lokalbus– X2X Link, das Versorgungsmodul sowie das Eingangs- und Ausgangsmodul, angehängt. Die X2X Link Knotennummer wird automatisch für jedes weitere X2X Link Modul, das diesem Buscontroller zugeordnet wird, erhöht. Die Prozessdatenzuordnung der einzelnen Variablen mit dem Ausgangsmodul erfolgt über eine I/O Mapping Datei.
Um die CPU auf die höchste Performance zu konfigurieren, wurde das Zeitverhalten der CPU verändert. Hierbei wurde der Softwaretaktgeber abgeschaltet und der interne Takt des POWERLINK’s der CPU neu zugeordnet.

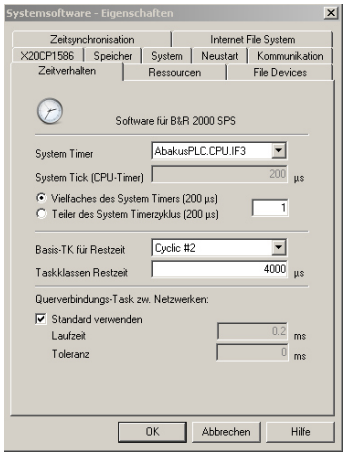


Abbildung 5.20: X20CP– CPU Zeitverhalten konfigurieren

Der “System Tick“ ist die Zeitbasis für den Taktgeber, von der die Zykluszeit abgeleitet wird. Bei Einsatz eines externer Taktgeber (beispielsweise POWERLINK) wird die Zeitbasis dieses Taktgebers herangezogen. Aus diesem Grund muss die Zeitbasis des POERLINK’s angepasst werden.

IF3		X20CP1586 (POWERLINK)
Modultyp	Typ 4	Beschreibt Funktionsumfang des Moduls
Betriebsmodus	POWERLINK V2	
MTU Grösse	300	
Baudrate	100 MBit Halbduplex	
POWERLINK Parameter		
POWERLINK Kommunikation aktivieren	Ein	
Gerätename	<InterfaceAddress>	
Zykluszeit [us]	200	

Abbildung 5.21: POWERLINK Zykluszeit konfigurieren

Für das PowerPanel wurde neben dem Programmcode eine Visualisierung erstellt. Auf die Erstellung dieser Visualisierung wird hier nicht weiter eingegangen, da dies für den Performancevergleich nicht relevant ist. Die erstellte Visualisierung ist in der Anlage D Seite 97 hinterlegt. Die Konfiguration kann über das Erstellen einer Compact Flash Karte oder über eine Ethernet- Verbindung auf das Gerät geladen werden. Hierfür muss das Ethernetmodul der Steuerung in der “Physical View“ entsprechend der Vorgabe für die IP- Adresse und der Subnetzmaske konfiguriert sein.

Die Implementierung der B&R Systeme ist nach einer gewissen Einarbeitungszeit in das AutomationStudio durch die strukturierte Unterteilung in “Logical View“, “Physical View“ und “Configuration View“ praxisnah und logisch gegliedert. Die Projekterstellung über den Wizard gestaltet sich als sehr effektiv. Ein weiterer Vorteil von AutomationStudio liegt in der Integration sämtlicher benötigten Funktionen für die Inbetriebnahme der Steuerung in einem Komplettsoftwarepaket.

6 Performance

In diesem Kapitel werden die Ergebnisse des Testszenarios näher betrachtet. Hierfür wird zunächst das Mess- und Monitoringsystem vorgestellt. Im zweiten Teil dieses Kapitels werden die Ergebnisse mit den prognostizierten Systemverhalten verglichen. Dabei wird unter anderem versucht, die Verhaltensweisen zwischen dem prognostizierten und dem tatsächlichen Systemverhalten herauszuarbeiten.

6.1 Monitoringsystem

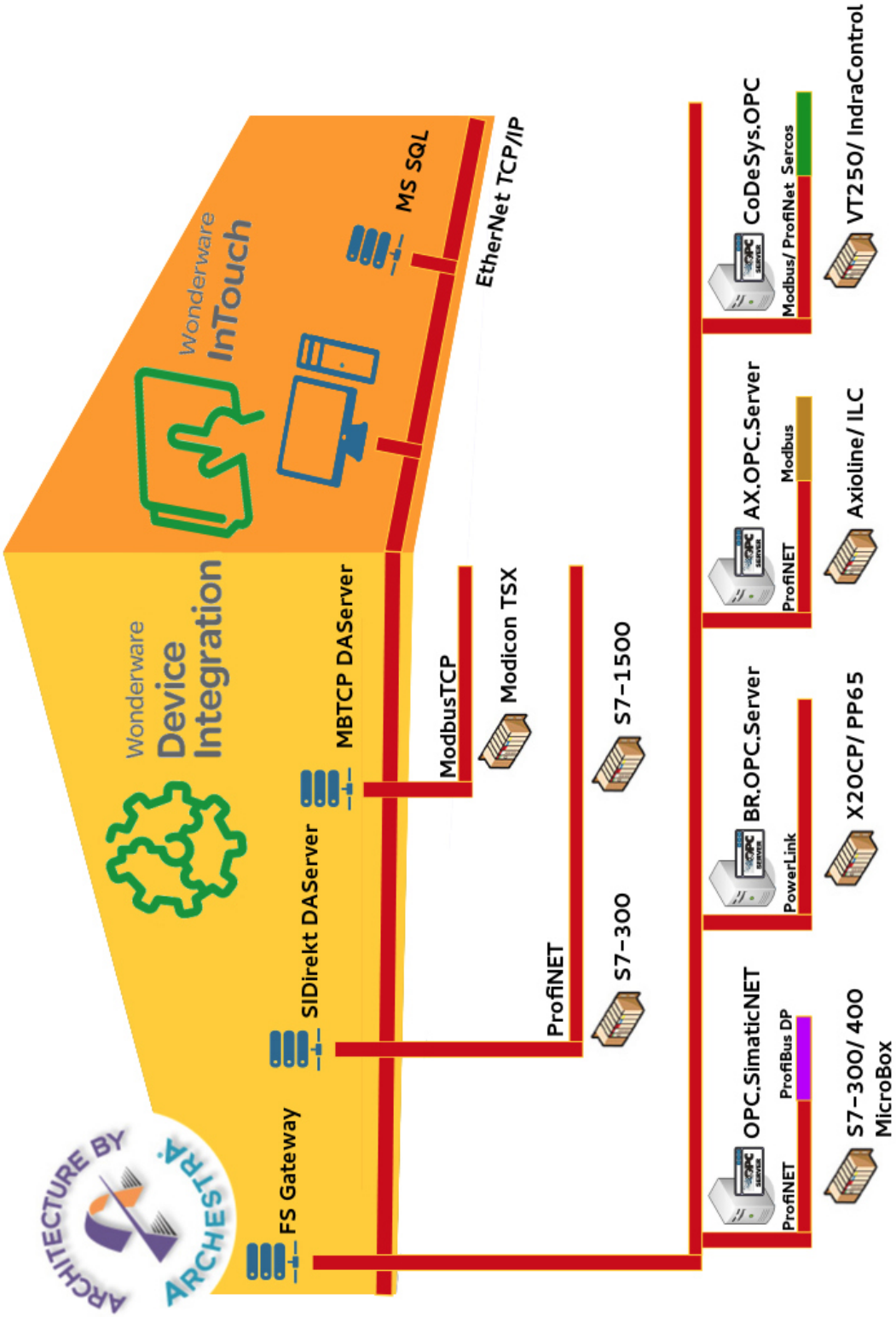
Für das Bedienen, Beobachten und Aufzeichnen der einzelnen Systeme wurde ein Monitoringsystem entwickelt. Über dieses System können die einzelnen Komponenten des Performancevergleichs mithilfe einer Visualisierung bedient und beobachtet werden. Als Anzeige- und Bedienkomponente wurde ein Rechner mit der Software InTouch von der Firma Wonderware zur Prozessvisualisierung verwendet. Wonderware stellt Lösungen für HMI/ SCADA Systeme zur Verfügung. Es ist eine herstellerunabhängige Softwareumgebung für die Entwicklung von Prozessleitsystemen. Mithilfe von verschiedenen Client- Server Konzepten wurde eine Ankopplung des Monitoringsystems an die Steuerungssysteme realisiert. Damit wird ermöglicht, die Zählzustände visuell darzustellen und alle Systeme zu steuern. Ein weiterer entscheidender Grund für die Entwicklung dieses Monitoringsystems ist die Möglichkeit, dass hierüber alle Systeme zu einem definierten Zeitpunkt gestartet werden können. Die ermittelten Messwerte werden mit Hilfe einer Datenbankanbindung aufgezeichnet. Dies ermöglicht eine detaillierte Auswertung des Testszenarios.

Für die Kopplung des Visualisierungssystems an die Steuerungssysteme wurde die OPC Technologie verwendet. OPC ist eine standardisierte Schnittstelle für den Datenaustausch zwischen herstellerunabhängigen Automatisierungsanwendungen. Die Schnittstelle basierte ursprünglich auf der OLE Technologie von Windows. Durch die Weiterentwicklung dieser Schnittstelle für den dynamischen Datenaustausch entstanden die Technologien DCOM und .NET. Diese Windows Schnittstellen wurde für die Bedürfnisse des Datenzugriffs in der Automatisierung erweitert. Dort wird diese Schnittstelle vorwiegend zum Lesen und Schreiben von Werten aus der Steuerung verwendet. Man unterscheidet hierbei zwischen OPC Clients und OPC Server. OPC Clients sind typischerweise Visualisierungen, die vom Anwender so konfiguriert werden, dass die anzuzeigenden Daten vom Server gelesen und die Daten für die Bedienung in den Server geschrieben werden. Im OPC Server werden die SPS Systeme und Feldbuskarten systemseitig zur Verfügung gestellt. Jeder Steuerungshersteller bietet heutzutage für

seine Steuerungen einen eignen Server an. Diese Server entsprechen den Vorgaben der OPC Foundation, damit jeder OPC Client auf die bereitgestellten Daten zugreifen kann. [35, Vgl. S.40-45]

Der OPC Server ist ein ausführbares Programm, das durch den OPC Client gestartet wird. Dies geschieht automatisch, sobald der OPC Client eine Verbindung zur Steuerung aufgebaut hat. Der OPC Server stellt alle Variablen, die auf der Steuerung aktiv sind, in einer Item List dem OPC Client zur Verfügung. Der OPC Server holt die Prozessdaten von der SPS und stellt sie im OPC Server als OPC Objekte zur Verfügung. Für den Zugriff des Servers auf die Items müssen die Variablen in der Entwicklungsumgebung entsprechend freigegeben und deren Zugriffsrechte konfiguriert sein. Verwaltet werden diese Items in einem Data Cache des OPC Servers. Dies ermöglicht einen schnellen Zugriff auf die Item-Werte. Damit stellt der Server die aktuelle Daten der Item Liste den Clients als Gateway im TCP/IP Protokoll zur Verfügung. [35, Vgl. S.40-45] Die Schnittstelle für den OPC Client ist durch die OPC Foundation standardisiert. In dieser wird festgelegt von welchem Server, welche Variablen/ Items geschrieben und gelesen werden.

Bei vielen Herstellern von HardSPS, wie beispielsweise SIEMENS, sind Server und Gateway in einem Produkt integriert. Dies bedeutet aber, das für die unterschiedlichen Feldbusarten auch unterschiedliche OPC Server angeboten werden. Da beispielsweise CodeSys auf unterschiedlichen Hardwareplattformen verwendet wird, müssen hier das Gateway und der Server getrennt verwendet. Im folgenden wird näher auf das Monitoringsystem für die Zählerdatenerfassung eingegangen.


Wonderware
InTouch
MS SQL
EtherNet TCP/IP
Wonderware
Device
Integration
MBTCP DAServer
SIDirekt DAServer
FS Gateway
ModbusTCP
Modicon TSX
ProfiNET
S7-1500
S7-300
CoDeSys.OPC
Modbus/ ProfiNet Sercos
VT250/ IndraControl
AX.OPC.Server
ProfiNET
Modbus
Axioline/ ILC
BR.OPC.Server
PowerLink
X20CP/ PP65
OPC.SimaticNET
ProfiNET
ProfiBus DP
S7-300/ 400
MicroBox
ARCHITECTURE BY
ARCHESTRA

In der Abbildung 6.1 ist schematisch der Aufbau des Gesamtsystems mit dem Monitoringsystem dargestellt. Zur Realisierung wurde die Systemoberfläche ArchestrA Technology von Wonderware verwendet. ArchestrA stellt eine komplette Automatisierungs- und Informationssoftware Architektur dar, die auf offene Industriestandards und Softwaretechnologien aufbaut. ArchestrA passt sozusagen Microsoft .NET und andere Microsoft Technologien an die industriespezifischen Applikationen an. Aus dieser ArchestrA Technologie wurde die Wonderware InTouch HMI zur Visualisierung und Bedienung verwendet, sowie die Wonderware Device Integration Servers zur Kommunikation mit dem Prozess. Die Verbindung zu dem Prozess ist die Voraussetzung für effektive Informationsverwaltung in Echtzeit. Wonderware bietet hierzu die umfangreichsten Kommunikationsmöglichkeiten für industrielle Automatisierungs- und Informationssysteme an. Für die Kommunikationsverbindung der unterschiedlichen Systeme wurden das FactorySuite Gateway, der ModbusTCP DAServer und der SIDirect DAServer verwendet. Diese Systeme werden in der System Management Console (SMC) verwaltet.

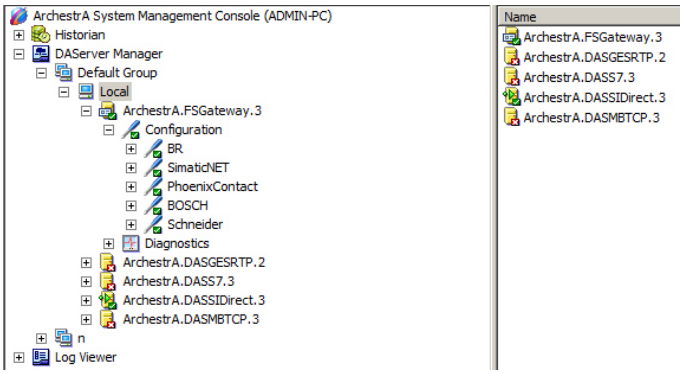


Abbildung 6.1: System Management Console mit Integration Server

Factory Suite-Gateway (FS Gateway) ist ein Microsoft Windows-Anwendungsprogramm, das als ein Kommunikationsprotokoll-Wandler wirkt. Es basiert auf dem ArchestrA DAS Toolkit. FS-Gateway verbindet verschiedene Server und Datenquellen, die über verschiedenen Datenzugriffsprotokolle kommunizieren. Die Client-Applicationen können eine Verbindung zum FS Gateway mit OPC-, SuiteLink- und DDE/FastDDE- Protokollen aufbauen. Bei diesem Projekt wurden ausschließlich die Treiber für Unterstützung der OPC Schnittstelle verwendet. FS-Gateway kann unter Verwendung von Standard-COM-Aktivierungsmechanismen COM-Server (OPC-Server) aktivieren. Durch diese Protokollunterstützung erfolgt auch der Datenaustausch zu Wonderware InTouch. FSGateway wird ausschließlich als Client der einzelnen OPC Server verwendet. Es wurden folgende OPC Server in das FSGateway eingebunden:

- OPC.SimaticNET
- CoDeSys.OPC
- BR.OPC
- AX.OPC.Server

An den jeweiligen OPC Servern sind die einzelnen Steuerungen aus dem Testszenario deklariert. Der OPC Server SimaticNET stellt die Daten von den S7-300 Steuerungen (System 4,5), der S7-400 Steuerung (System 18) und der S7-Microbox (System 19) zur Verfügung. Die S7-400 ist über einen Kommunikationsprozessor in das Ethernet TCP/IP Netz integriert. Die S7- Microbox ist über eine universelle (MPI/ DP) PCI-Karte am OPC Server PC eingebunden. Für die Verwendung dieses OPC Servers muss der OPC.SimaticNET im Siemens-Komponenten-Konfiguration deklariert sein. Die Verbindung zu den einzelnen Steuerungen wird im NetPro über S7-Verbindungen hergestellt. Die einzelnen Einstellungen wurden über dem Step7 Manager in die Baugruppen geladen. An den CoDeSys OPC Server sind die Systeme der Firma TURCK (System 7, 12) und BOSCHRexroth (System 14) eingebunden. Hierfür werden die Steuerungen in das Ethernet TCP/IP Netz integriert und über die entsprechende Knotenadresse der Steuerung in der OPC Konfiguration deklariert. An dem PhoenixContact OPC AX Server sind die Axioline Steuerungen (System 8, 9, 11) und die ILC Steuerung (System 7) eingebunden. Für den Datenaustausch zwischen der Steuerung und dem OPC Server müssen die einzelnen Variablen in PcWorx als OPCItems deklariert sein. Der OPC Server von B&R läuft direkt integriert auf den einzelnen Steuerungen unter dem Echtzeit Betriebssystem. Bei der Projektierung der Steuerungsapplikation wird im Automation Studio, durch den direkten Zugriff auf die Prozessvariablen, die OPC Konfiguration mit erstellt und verwaltet. Diese OPC Server werden im FS Gateway über OPC Objekte eingebunden. Hierfür wird der Server Name angegeben und der OPC Item ID Prefix für das Ansprechen der Variablen.

Der Wonderware MBTCP DAServer (MBDAServer) ist ein Microsoft Windows Anwendungsprogramm, das als ein Kommunikationsprotokoll Server agiert. Es ermöglicht anderen Windows-Anwendungsprogrammen den Zugriff auf Daten in der Modbus und Schneider TSX Momentum Steuerungen, die mit dem Ethernet TCP/IP verbunden sind. Der DAServer stellt die Prozessvariablen der Steuerungen über Ethernet-Ports unter Verwendung des Modbus TCP / IP Protokoll zur Verfügung. In der Arbeit werden diese Daten ausschließlich für Wonderware InTouch über SuiteLink freigegeben. An diesen DAServer ist die Modicon TSX Steuerung implementiert. Der DAServer wird in der Management Console deklariert. Hierfür muss der Modbus Port 502 verwendet, sowie das “TSX Premium“ Modbus Objekt an diesen Port angefügt werden. In diesem Objekt werden dann die Bitorder und das Format konfiguriert. Die entsprechenden Variablen werden in InTouch über Register angesprochen.

Der SIDirect DAServer ist ein Microsoft Windows-Anwendungsprogramm, das als ein SIEMENS Kommunikationsserver agiert. Der DAServer ermöglicht windowsbasierende Anwendungsprogrammen den Zugriff auf Daten der Siemens S7-200/ 300/ 400/ 1200/ 1500 -Steuerungsfamilie. Dieser Industrie- Applicationsserver bietet über eine off-the-shelf Standard-Ethernet-Netzwerkkarte einen Zugriff auf SIEMENS Steuerungen. Der DAServer wird in der Diplomarbeit für die S7-300 (System 1) und S7-1500 (System 2,3) Steuerungen verwendet. Die Prozessdaten werden in diesem Projekt über die Suite-

Link Verbindung der InTouch Anwendung freigegeben. Für die Konfiguration des SIDirect DAServers muss in der Managment Console ein TCP/IP Port freigeben werden, an dem das S7 PLC Objekt angehängt ist. Die Identifikation der entsprechenden S7 Steuerung erfolgt über die IP Adresse, die Racknummer und der Modulnummer. Die Prozessvariablen der Steuerung werden dann über die jeweiligen Merker direkt angesprochen.

Die Prozessvariablen der Steuerungen werden in InTouch über die Archestra Tools und Device Server zur Verfügung gestellt. Hierfür müssen für die einzelnen Steuerungen Zugriffsnamen festgelegt werden. In diesen Zugriffsnamen wird der Anwendungsname (FSGateway, DASSIServer, MBTCPDAServer) sowie der Topic- Name der einzelnen Steuerung definiert. Der Anwendungsname ist der Server (FSGateway, DASSIServer, MBTCPDAServer) in dem die Steuerung deklariert ist. Der Topic-Name ist der Zugriffspfad auf die Steuerung. Als Protokoll wird jeweils SuiteLink verwendet. Die Parametrierung der Variablen findet in der Variablenliste statt. Hier wird der Variable der Zugriffsname und die Itemadresse zugeordnet. Die Variablen werden über das Item Prefix, Register oder direkt über die Merkeradressen angesprochen. Die Variablen der einzelnen Zähldekaden wurden auf diese Weise in InTouch implementiert. Somit können die aktuellen Zählerstände der einzelnen Dekaden auf der Visualisierungsoberfläche von InTouch angezeigt werden.

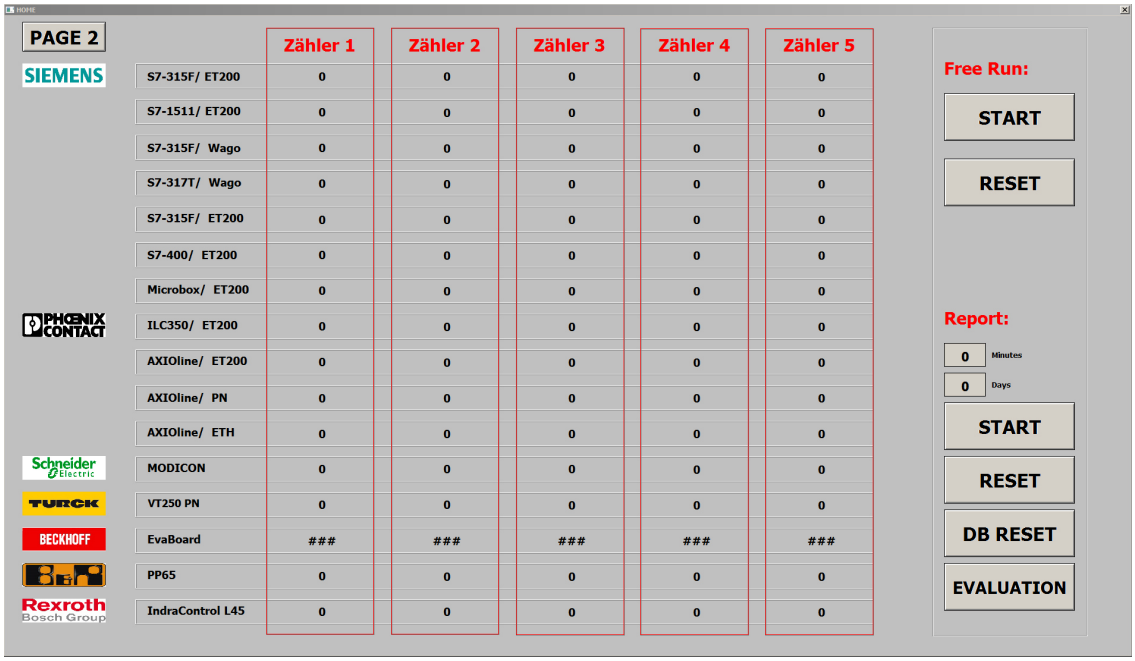


Abbildung 6.2: HOME– InTouchoberfläche

Die Abbildung 6.2 zeigt die Visualisierung zur Darstellung der Zählerstände auf den einzelnen Steuerungssystemen. Auf dieser Oberfläche werden links die herstellerspezifischen Systeme abgebildet. Im zentralen Teil der Visualisierung sind die einzelnen

Zählerstände, nach Dekaden getrennt, wiedergegeben. Auf der rechten Seite befindet sich ein Navigationspanel. Die Variablen der Zählerstände werden in einem 100ms Zyklus aktualisiert und für die Dekaden 1 bis 5 dargestellt. Die Aktualisierungszeit wird bei den Archestra Tools und Device Servern angegeben. Auf diese Art werden die Zählerstände für alle 19 Systeme visualisiert.

Über das Navigationspanel wird das Testszenario gesteuert. Darüber kann der dekadische Zähler durch das Startsignal auf allen Systemen zu einem definierten Zeitpunkt gestartet werden. Dies ist im “Free Run“- Modus oder im “Report“- Modus möglich. Im “Free Run“- Modus werden die dekadischen Zähler der einzelnen Steuerungen zur gleichen Zeit gestartet. Über einen Reset Button werden die Zählerstände Systeme zurückgesetzt. Der “Report“- Modus ermöglicht, das Zählverhalten der verschiedenen Systeme aufzuzeichnen. Hierfür ist es notwendig die Zeit anzugeben, wie lange die Aufzeichnung erfolgen soll. Über den “Start“ Button wird die Aufzeichnung gestartet und kann über den “Reset“ Button manuell zurückgesetzt werden. Für die Aufzeichnung der Zählerstände werden die Daten in eine Datenbank geschrieben. Für das Datenbankmanagement wird das Programm Microsoft SQL– Server Management Studio verwendet. Der MS SQL Server ist ein relationales Datenbankmanagmentsystem, das zur elektronischen Datenverwaltung in Computersystemen verwendet wird. Ein Datenbankmodell ist die theoretische Grundlage für eine Datenbank und bestimmt auf welcher Art und Weise die Daten in dem Datenbanksystem gespeichert und bearbeitet werden. Die entwickelte SQL Datenbank dient allein der BDE Datenerfassung. Es gibt für jedes System eine Tabelle, die jeweils die Zählwerte der einzelnen Dekaden, den Startzeitpunkt und die aktuelle Zeit abspeichert. Die Zählwerte der Zaehler 1 bis 5 werden im Format “tinyint“ und die Startzeit/ Endzeit im Format “datetime“ gespeichert. Es wird neben diesen Werten auch die Zeitdifferenz zum Startzeitpunkt mithilfe eines SQL Skripts berechnet. Wegen der großen zeitlichen Unterschiede zwischen dem Ablauf der einzelnen Dekaden wird hier in Sekunden, Minuten, Stunden und Tage unterteilt. Mithilfe des SQL Skripts werden diese Differenzen berechnet sobald ein neuer Datensatz eingetragen wird. Dadurch wird eine schnelle Auswertung der Datensatze ermöglicht.

Column Name	Data Type	Allow Nulls
ID	bigint	☐
Zaehler1	tinyint	☑
Zaehler2	tinyint	☑
Zaehler3	tinyint	☑
Zaehler4	tinyint	☑
Zaehler5	tinyint	☑
StartTime	datetime	☑
EndTime	datetime	☑
DIFFday		☑
DIFFhour		☑
DIFFmin		☑
DIFFsec		☑
		☐

Abbildung 6.3: Tabellen Design

Die Verbindung zwischen InTouch und MS SQL wird über die ODBC Schnittstelle umgesetzt. ODBC ist eine standardisierte Datenbankschnittstelle, die SQL als Datenbanksprache verwendet. Im SQL Access Manager von InTouch müssen die entsprechenden Bindelisten für die Datenbankanbindung definiert werden. Es gibt hier für jedes System eine Import- und eine Exportbindeliste. Die Exportbindeliste enthält die Datensätze die in die SQL Datenbank geschrieben werden müssen und die Import Bindeliste enthält die Datensätze aus der Datenbank. Die einzelnen Datensätze werden über ein Globales Skript in die Datenbank geschrieben, sobald der Startbutton gedrückt wurde. Hierfür wird die Funktion SQLInsert() verwendet. Mit dieser Funktion können neue Datensätze in die Datenbank geschrieben werden. Die Werte, die in die einzelnen Spalten des Datensatzes geschrieben werden, werden den InTouch Variablen entnommen, die in der angegebenen Bindeliste definiert sind. Für das Ausführen der Funktion muss zunächst über SQLConnect eine Verbindung zur Datenbank hergestellt werden. Das Globale Skript wird dann in dem vordefinierten Zeitfenster abgearbeitet. Nach beendeter Aufzeichnung wird die Verbindung zur SQL Datenbank geschlossen und eine weitere InTouch Seite freigegeben. Sie visualisiert eine Auswertung des Systemtests. Diese Oberfläche sieht wie folgt aus:

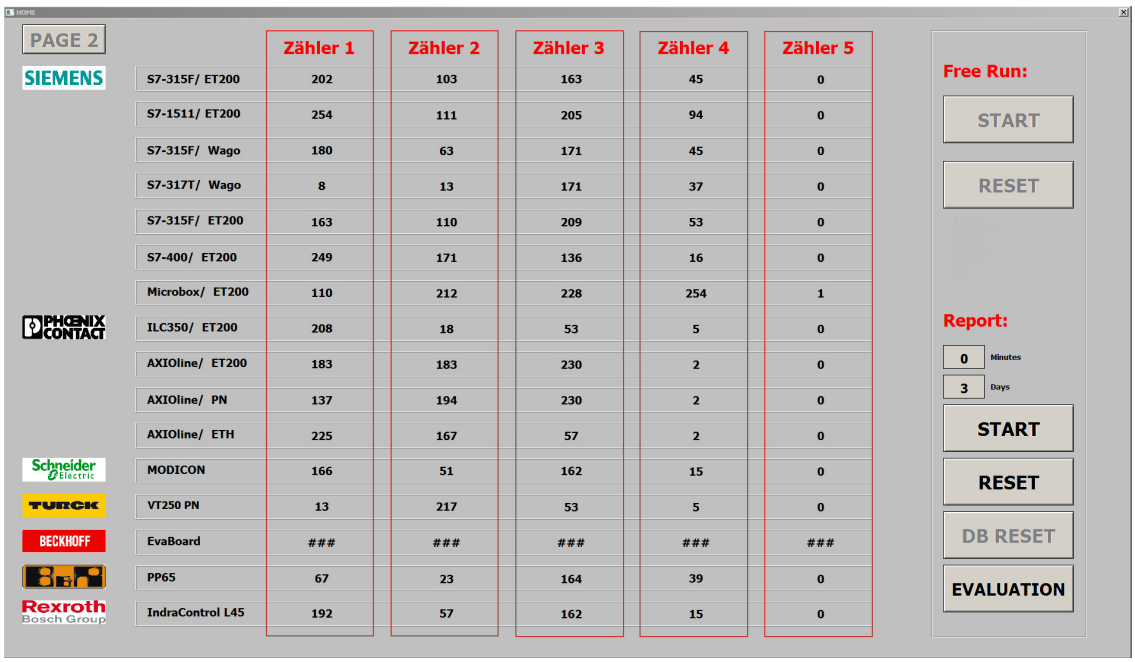
Abbildung 6.4: Auswertung– InTouchoberfläche

Auf dieser Seite werden die Ergebnisse des Testszenarios für die verschiedenen Systeme dargestellt. Für jedes System werden die Zeiten gezeigt, die diese bis zur Beendigung der einzelnen Dekade benötigen. Die Zeiteinheiten sind für jeder Dekade unterschiedlich. Ursache hierfür sind die verschiedenen Zeitfenster bis zur Beendigung der Dekade. Die erste Dekade wird bei der Auswertung nicht beachtet, da die Zeitdifferenz zu gering ist. Beim System 19 (S7-MicroBox) ist die Zeitdifferenz auch in der zweiten

Dekade noch zu gering und wird nicht bewertet. Die Differenzen werden über ein SQL Skript berechnet und dann über die Importbindelisten der Systeme in der InTouch Oberfläche dargestellt. Für das Abrufen der einzelnen Werte wird ein globales Skript mit der Funktion “SQLSelect“ verwendet. Diese Funktion ruft Datensätze anhand bestimmter Kriterien aus einer Tabelle ab. Das Kriterium ist hierbei, das die jeweilige Dekade den Wert 255 erreicht hat. Die abgerufenen Datensätze werden in einer temporären Ergebnistabelle im Arbeitsspeicher abgelegt. Aus diesem Auswahlpuffer werden die Werte dann über Variablenverknüpfungen auf der Oberfläche angezeigt. Eine detaillierte Ansicht der einzelnen Visualisierungen ist in der Anlage E Seite 98 zu finden.

6.2 Auswertung

Auf Grundlage der beschriebenen Systemen, Betrachtungen und Bedingungen wurde ein Systemtest von 3 Tagen mithilfe des Monitoringsystems gefahren. Diese Messung wurde 3 mal wiederholt, um eventuelle Abweichungen festzustellen. Der Vergleich der Messreihen ergab keine Abweichungen untereinander. Differenzen wurden nur in der ersten Zähldekade festgestellt. Diese Dekade wird sehr schnell abgearbeitet und kann somit nicht so schnell aufgezeichnet werden. Nach Ablauf des Testzeitraums konnten folgende Programmabarbeitungen der Systeme aufgezeichnet werden.



	Zähler 1	Zähler 2	Zähler 3	Zähler 4	Zähler 5
S7-315F/ ET200	202	103	163	45	0
S7-1511/ ET200	254	111	205	94	0
S7-315F/ Wago	180	63	171	45	0
S7-317T/ Wago	8	13	171	37	0
S7-315F/ ET200	163	110	209	53	0
S7-400/ ET200	249	171	136	16	0
Microbox/ ET200	110	212	228	254	1
ILC350/ ET200	208	18	53	5	0
AXIOline/ ET200	183	183	230	2	0
AXIOline/ PN	137	194	230	2	0
AXIOline/ ETH	225	167	57	2	0
MODICON	166	51	162	15	0
VT250 PN	13	217	53	5	0
EvaBoard	###	###	###	###	###
PP65	67	23	164	39	0
IndraControl L45	192	57	162	15	0

Abbildung 6.5: Zählerstandentwicklung nach einer Laufzeit von 3 Tagen

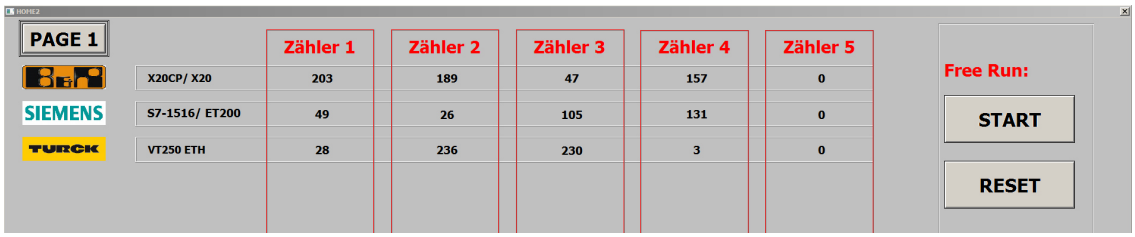
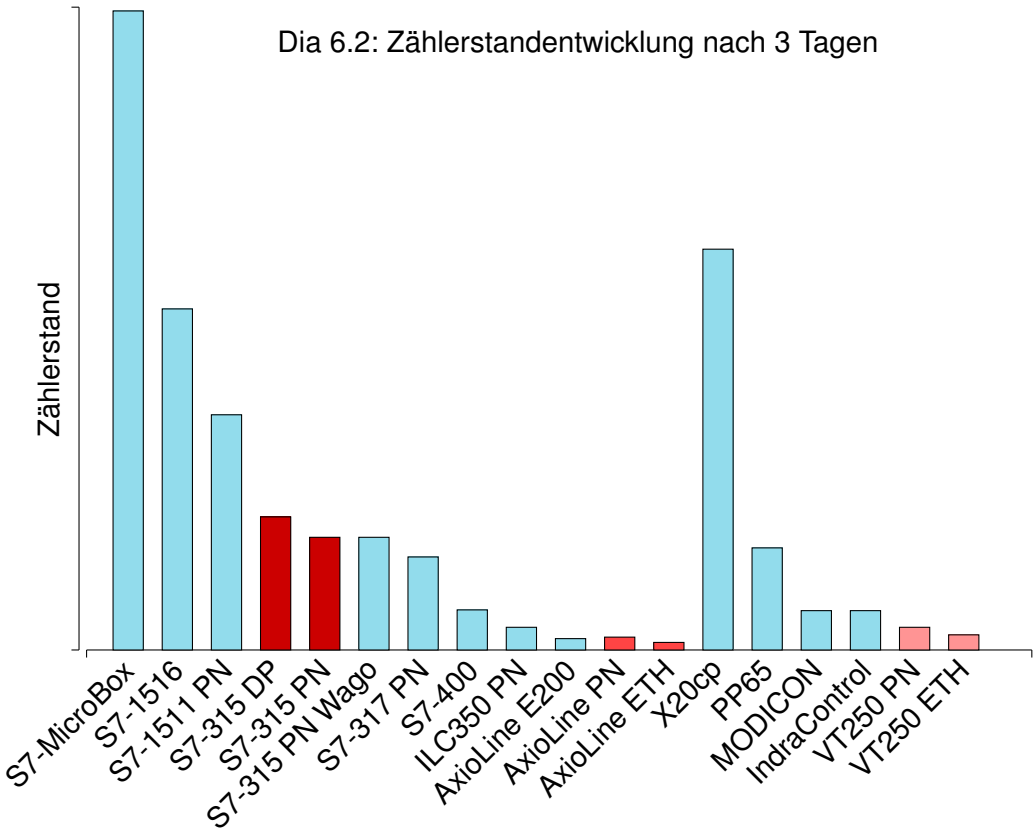


Abbildung 6.6: Zählerstandentwicklung nach einer Laufzeit von 3 Tagen

In den Abbildungen 6.5 und 6.6 ist ersichtlich, dass nur das System 19 mit der S7-MicroBox die vierte Dekade beendet hat. Innerhalb der 3 tägigen Testzeit haben alle Systeme die 1., 2. und 3. Dekade beendet. Wie in den Vorbetrachtungen schon festgestellt wurde, ist die Zählerstandentwicklung der einzelnen Dekaden sehr unterschiedlich. Ursache hierfür sind einerseits die unterschiedlichen Zykluszeiten der Steuerungen und andererseits die Übertragungsgeschwindigkeit des jeweiligen Feldbussystems. Es ist ersichtlich, dass die SIEMENS Systeme sehr hohe Zählerstände erreicht haben. Dort erfolgte die Programmabarbeitung in einer freilaufenden Task ohne Minimalzeit. Alle anderen Systeme haben eine festgelegte Zykluszeit (B&R- und SchneiderElectric Geräte) oder einen freilaufenden Task. Dieser Task ist aber durch eine Minimallaufzeit begrenzt. Nähere Informationen der einzelnen Zykluszeiten der Systeme sind im Kapitel 4, Seite 44 zu finden.



In dem Diagramm 6.2 ist die gesamte Zählbreite der einzelnen Systemaufbauten dar-

gestellt, die sich über die drei Tage entwickelt hat. Systeme, die auf gleichem Systemaufbau basieren, jedoch über einen unterschiedlichen Feldbus kommunizieren, wurden im Diagramm mit Rot markiert. Das sind zum einen Systeme mit ProfiNet/ProfiBus und zum anderen die Systeme unter ProfiNet/Modbus.

Ersichtlich ist der gravierende Unterschied zwischen den seriellen Feldbussen (ProfiBus) und den Ethernet basierenden Feldbussen. Die Differenzen entstehen bei den ProfiNet und ProfiBus System durch die Nettodatenrate. Diese Rate ist der eigentliche Anteil an Informationen in einem Datenpaket. Dadurch kann ein Protokoll eines seriellen Feldbusses mit einer geringeren Bautrate schneller sein als ein Ethernet basierender Feldbus mit einer weitaus höheren Bautrate. Der Grund dafür ist auch, dass die Ethernet basierenden Feldbusse je nach Art, sehr viel Overhead benötigen. Overhead sind Zusatzinformationen der zu übertragenden Daten für den Sender und Empfänger. Aus diesem Grund kann ein Feldbus mit einer hohen Nutzdatenrate wesentlich mehr echte Informationen, als ein gleich schneller oder schnellerer Bus mit großen Overhead, übertragen. [42, Vgl.]
Profibus hat in diesem Fall eine Übertragungsrate von 1,5 MBit/s und einen maximalen Datendurchsatz von 32 Byte. Das Telegramm hat eine maximale Länge von 240 Byte. Hiervon ist der Overhead bei Profibus bei jedem Zeichen (UART) 3 Bit groß. Hierbei sind keine Nutzdaten enthalten. Jedes Telegrammformat besteht aus unterschiedlichen Steuerzeichen. Abhängig von diesem Format variiert der Overhead zwischen 1 bis 12 Bytes. Nachteilig im Zeitverhalten wirken sich dazu die relativ langen Abarbeitungszeiten der Standard Protokolle TCP/IP und UDP/IP aus. Die Übertragungszeiten sind sehr stark von dem Overhead abhängig. Das Ethernet Minimaltelegramm mit 46 Bytes hat beispielsweise einen Overhead von 38 Bytes. Hingegen hat das ProfiNet Protokoll 44 Bytes Overhead und einen Nutzdateninhalt von minimal 36 Bytes. Das eigentliche ProfiNet- Telegramm hat eine Übertragungsdatenrate von 100 MBit/s und einen maximalen Datendatendurchsatz von 254 Byte. Trotz dieser besseren Leistungsdaten kann es vorkommen, dass es bei Konstellationen mit wenig Nutzdatenanteil, der ProfiBus eine höhere Performance als ProfiNet hat. Das liegt an dem Aufbau und der Funktionalität von Ethernet. Der Grund hierfür ist zum einem der Overhead und zum anderem die Mindesttelegrammlänge. Jedes Ethernettelegramm muss eine bestimmte Mindestlänge von 64 Byte besitzen. Die Nutzdaten sind in diesem Test so gering, dass die Mindestlänge mit erreicht wird. In diesem Fall wird das Telegramm mit einem sogenannten PAD-Füllfeld aufgefüllt, damit die Mindestlänge erreicht wird. Dies führt dazu, dass das System unter Profibus mit einer höheren Performence bei geringen Nutzdaten arbeitet, als das System unter ProfiNet. Bei anderen Telegrammarten, wie beispielsweise EtherCat, wird dieses prinzipiell vermieden, da hier ein Summenrahmentelegrammen generiert wird. Hierbei werden die Daten von mehreren Teilnehmern in einem zusammengefasst. Dies hat aber wiederum den Nachteil, das jedes Telegramm zu jedem Teilnehmer gesendet werden muss, mit der entsprechenden Durchlaufzeit für das Telegramm.

Es wurde in dem Testszenario zum einen das Turck System und zum anderen das

Axioline System mit unterschiedlichen Feldbussen, aber gleichen Systemaufbau, realisiert. Wie in dem Diagramm 6.2 zu sehen ist, gibt es bei dieser Gegenüberstellung große Unterschiede zwischen dem jeweiligen Systemaufbau unter Modbus und ProfiNet. Dies liegt an der Grundfunktionalität dieser beiden Bussysteme. Modbus ist ein TCP/IP basiertes Protokoll und ProfiNet ist eine MAC-basierte Kommunikation. Somit setzt Modbus auf Standard TCP/IP- Software- Stacks und nutzt Standard- Ethernet-Hardware. Bei diesem Bussystem ist die Echtzeitfähigkeit durch das Zeitverhalten der TCP/IP Stacks sehr begrenzt. Bei Modbus wird die Adressierung über die IP Adresse realisiert. ProfiNet verwendet die TCP/IP Stacks nicht und arbeitet somit direkt auf dem Ethernet Frame. Die Adressierung unter ProfiNet erfolgt über die MAC Adressen der Teilnehmer. Dadurch stehen Kanäle für IP-basierte Dienste zur Verfügung, sowie für die MAC Adressen orientierte Echtzeitkommunikation. Dies ermöglicht einen geringeren Overhead als bei dem Modbusprotokoll. Das bessere Zeitverhalten ist aus dem Testszenario erkennbar. [12, Vgl. S. 246-247] Die Performanceunterschiede durch die verschiedenen Arbeitsweisen der beiden Protokolle lassen sich am besten über Wireshark Mitschnitte darstellen und analysieren.

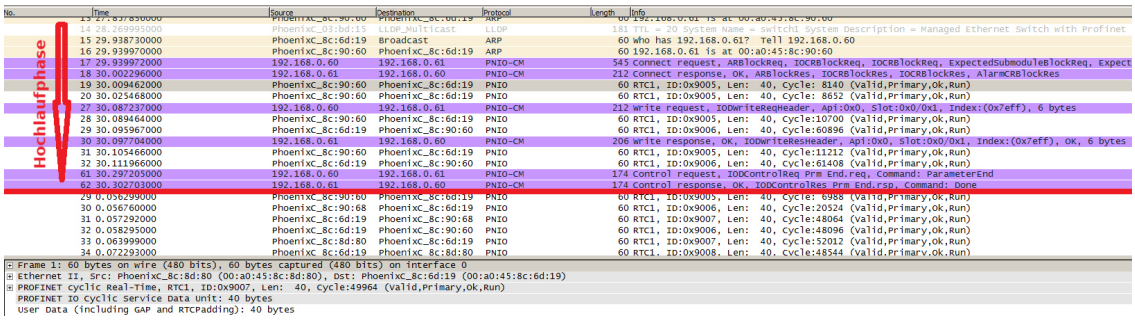


Abbildung 6.7: ProfiNet Kommunikation in Wireshark

Die Abbildung 6.7 zeigt einen Wireshark Mitschnitt von ProfiNet. Dieser beginnt zunächst mit der Systemhochlaufphase. Hierbei findet über ARP (Address Resolution Protokoll) die Namens- und Adressauflösung statt. Danach erfolgt die IP- Adressenvergabe. Nachdem die IP-Adresse vergeben wurde, beginnt der Datenaustausch mit verschiedenen Aufrufen. Dies sind der Connect-, Write- und Control Aufruf. Ein solcher Aufruf besteht immer aus einem Request und einem Response. Im Connect Request leitet der ProfiNet Master einen Verbindungsaufbau mit dem ProfiNet Device ein. Durch den Connect Response wird der Verbindungswunsch nach einer Überprüfung von Device beantwortet. Nach dem Connect Aufruf folgt der Write Request. Hierbei werden im Hochlauf die Parameterdaten entsprechend der GSD Datei dem IO Device übertragen. Der Empfang dieser Daten wird von dem Write Response quittiert. Der Control Request folgt nach dem Write Aufruf. Mit dem Control Request wird signalisiert, dass die Parametrierung abgeschlossen ist und die Verbindung steht. Der Nutzdatenaustausch findet nun statt. Der Hochlauf wird immer im Non-Real-Time Kanal über UDP/IP abgewickelt, der Nutzdatenverkehr erfolgt dann über den Real-Time Kanal. [43, Vgl. S.51-66] Neben dieser Hochlaufphase und dem darauffolgenden Nutzdatenaustausch ist in dem

Mitschnitt die Telegrammlänge von jeweils 60 Byte ersichtlich. Die Source und Destination Spalte des Mitschnitts zeigt die MAC Adressen basierte Echtzeitkommunikation bei ProfiNet.

No.	Time	Source	Destination	Protocol	Length	Info
2	0.012567000	192.168.0.40	192.168.0.41	Modbus/TCP	69	Query: Trans: 10658; Unit: 1, Func: 16: Write Multiple Registers
3	0.013300000	192.168.0.41	192.168.0.40	Modbus/TCP	66	Response: Trans: 10658; Unit: 1, Func: 16: Write Multiple Registers
5	0.023568000	192.168.0.40	192.168.0.44	Modbus/TCP	69	Query: Trans: 10659; Unit: 1, Func: 16: Write Multiple Registers
6	0.024301000	192.168.0.44	192.168.0.40	Modbus/TCP	66	Response: Trans: 10659; Unit: 1, Func: 16: Write Multiple Registers
8	0.032806000	192.168.0.40	192.168.0.41	Modbus/TCP	69	Query: Trans: 10660; Unit: 1, Func: 16: Write Multiple Registers
9	0.033569000	192.168.0.41	192.168.0.40	Modbus/TCP	66	Response: Trans: 10660; Unit: 1, Func: 16: Write Multiple Registers
11	0.050805000	192.168.0.40	192.168.0.41	Modbus/TCP	69	Query: Trans: 10661; Unit: 1, Func: 16: Write Multiple Registers
12	0.051569000	192.168.0.41	192.168.0.40	Modbus/TCP	66	Response: Trans: 10661; Unit: 1, Func: 16: Write Multiple Registers

```
⊞ Frame 2: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0
⊞ Ethernet II, Src: PhoenixC_8c:6f:29 (00:a0:45:8c:6f:29), Dst: PhoenixC_8c:5e:31 (00:a0:45:8c:5e:31)
⊞ Internet Protocol Version 4, Src: 192.168.0.40 (192.168.0.40), Dst: 192.168.0.41 (192.168.0.41)
⊞ Transmission Control Protocol, Src Port: 1040 (1040), Dst Port: 502 (502), Seq: 1, Ack: 1, Len: 15
⊟ Modbus/TCP
    Transaction Identifier: 10658
    Protocol Identifier: 0
    Length: 9
    Unit Identifier: 1
⊟ Modbus
    Function Code: Write Multiple Registers (16)
    Reference Number: 9000
    Word Count: 1
    Byte Count: 2
    Register 0 (UINT16): 72
```

Abbildung 6.8: Modbus Kommunikation in Wireshark

Im Vergleich hierzu zeigt die Abbildung 6.8 einen Wireshark- Mitschnitt unter Modbus. Zunächst ist das Client/ Server Protokoll “Modbus/TCP“ zu erkennen. Es ist ersichtlich, dass die Modbus Kommunikation auf TCP/IP basiert. Die Adressierung zwischen den Source und Destination Telegrammen wird jeweils über entsprechende IP- Adresse realisiert. Jeder Datenaustausch zwischen dem Master und dem Client findet in einem Query- Response Zyklus statt (rot makiert). Der Master arbeitet nur auf Anfrage des Clients. Der Zyklus beginnt mit einem Request Telegramm. Somit stellt der Client eine Anfrage an den Server. Dieser analysiert die Anfrage, führt die gewünschte Operation aus und sendet eine Antwort (Response Telegramm) an den Client zurück. Das Response Telegramm quittiert somit die Anfrage und enthält die Nutzdaten bezüglich der Anfrage des Clients. In diesem Telegrammen wird über den Funktionscode angegeben, welche Aktion vom Server ausgeführt werden soll. Die Länge des Request Telegramms ist 69 Byte und des Response Telegramms ist 66 Byte lang.

Die Arbeitsweise von Modbus zeigt, dass bei dieser Kommunikation die Netzlast höher ist als bei ProfiNet. Der Grund hierfür ist, dass bei ProfiNet nach dem Hochlauf ein direkter Datenaustausch und bei Modbus ein gesicherter Datenaustausch, bestehend aus Request- und Response- Telegramm, stattfindet. Die Größe der Telegramme und die unterschiedlichen Arbeitsweisen der Kommunikationen sind die Gründe für die unterschiedlichen Zählerstandentwicklungen der Systemaufbauten.

In diesem Vergleich haben einige sehr leistungsfähige Systeme einen relativ geringen Zählerstand erreicht. Das liegt zum einen daran, das nicht alle Systeme einen freilaufenden Task besitzen (B&R, SchneiderElectric) oder zum anderen an die Begrenzung der freilaufenden Task durch eine Minimallaufzeit. Betroffen davon sind alle CoDeSys Systeme der Firmen TURCK und BoschRexroth.

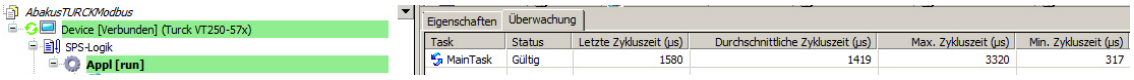


Abbildung 6.9: Zykluszeit der Turck VT250

Abbildung 6.10: Zykluszeit der BoschRexroth IndraControl L45

Die Abbildungen 6.9 und 6.10 bilden die Taskzykluszeiten im eingeloggten Modus dar. Die durchschnittliche Zykluszeit des TURCK Systems beträgt hierbei 1419 μs Zykluszeit und bei dem BoschRexroth System 48 μs. Die minimale Zykluszeit des TURCK Systems beträgt 3ms und des BoschRexroth Systems 1ms. Daraus folgt eine Ruhephase bei dem TURCK-System von 50% bezogen auf die Zykluszeit. Bei dem System der Firma BoschRexroth beträgt sie über 90% bezogen auf die Zykluszeit. Dies zeigt zum einen ungenutzte Ressourcen der Systeme aber zum anderen muss beachtet werden, ob bei Ausnutzung der Ressourcen die I/O Device dies richtig erkennen und auch in dieser Zykluszeit aktualisiert werden können.

System	**Master**	**Dekade 2 /s**	**Vorbetrachtung /s**
1	S7-315F	23	20.81
2	S7-1511	11	9.75
3	S7-1516	8	6.5
4	S7-315F	23	20.81
5	S7-317	28	24.71
6	VT250 PN	195	195.08
7	ILC350	187	195.08
8	AXC1050	259	260.1
9	AXC1050	259	260.1
10	Modicon	65	65.03
11	AXC1050	313	260.1
12	VT250 ETH	254	253.6
13	EvaKit	24	22.76
14	IntraControl	65	65.03
15	PP65	27	26.01
16	X20CP	7	6.5
17	S7-315F	20	20.81
18	S7-400	62	58.52
19	MicroBox	2	1.43

Tabelle 6.1: Bearbeitungszeiten für Dekade 2

Die Tabelle 6.1 zeigt eine Gegenüberstellung der in den Vorbetrachtungen aus den Zy-

kluszeiten berechneten Bearbeitungszeiten und der Ergebnissen aus dem dreitägigen Performancetest. Es ist zu sehen, dass es Abweichungen zueinander gibt. Dies hängt zum einen daran, dass die Zeitdifferenzen zwischen der Startzeit und Endzeit im SQL Skript auf Basis von Ganzzahlen berechnet werden und zum anderen hat die Netzwerkkarte durch die Frameaktualisierung einen gewissen Zeitversatz erzeugt. Des Weiteren muss der Fakt betrachtet werden, dass die Zykluszeiten für die Vorbetrachtungen über einen Zeitbaustein gemessen wurden. Hierbei war die Kommunikationslast um ein Vielfaches geringer, weil der OPC Server die Kommunikation der CPU nicht so stark belastet hat.

In den vorliegenden Untersuchungen wurde unter anderen auch betrachtet, inwiefern das Zählverhalten des Systems durch herstellerinhomogene Buskoppler, beeinflusst wird. Dafür wurden die Systeme 1 und 4 sowie die Systeme 8 und 9 untersucht. Die Systeme 1 und 4 haben jeweils die CPU S7-315F als Mastergerät. Das System 1 hat Buskoppler vom Typ Siemens ET200s und das System 4 hat WAGO 750-370 als E/A-Module. Bei Betrachtung der Bearbeitungszeit für die zweite Dekade (siehe Tabelle 6.1), sind keine Unterschiede festzustellen. Die Unterschiede sind erst bei Betrachtung der kompletten Dekadenstruktur des kaskadischen Zählers ersichtlich. Ab der dritten Dekade (siehe Abbildung 6.5) treten bei den beiden Siemens Geräten große Unterschiede auf. Bei den Systemen 8 und 9 sind die Unterschiede sehr gering. Das Verhalten, dass das SIEMENS System mit den ET200s Buskopplern langsamer ist, als das SIEMENS System mit den Wago Buskopplern, ist mit den Aktualisierungszeiten zu begründen. Die Aktualisierungszeit des Systems mit den ET200s beträgt 2ms. Die Aktualisierungszeit des Systems mit den Wago Buskoppler beträgt 16ms. Dieses entspricht einer 8-fach höheren Kommunikationslast der beiden Systemaufbauten untereinander. Hierbei wird die CPU des Systems mit den Wago Baugruppen weniger belastet. Auf Grund der Tatsache, dass beide Steuerungssysteme in einer Linienstruktur aufgebaut wurden, sind zusätzliche Latenzzeiten durch die Switche vorhanden. So gesehen ist bei einer solchen Struktur jeder Buskoppler ein Switch, der eine gewisse Zeitverzögerung mit sich bringt. Dies sind beispielsweise bei den SIEMENS Geräten $\sim$3.9μs. Bei einer Aktualisierungszeit von 2ms hat diese Zeitverzögerung öfter einen Einfluss auf das Systemverhalten, als bei einer Aktualisierungszeit von 16ms.
Bei dem System mit der Axioline SPS und dem ET200s als Buskoppler oder die Axiocontrol Buskoppler (System 8 und 9), sind die Unterschiede marginal. Wegen der geringeren CPU Leistungsfähigkeit der Axioline SPS wird nur eine Aktualisierungszeit von 16ms für 5 Buskoppler unterstützt. So ist die Aktualisierungszeit bei dem System mit den ET200s Buskopplern und dem System mit den Axioline Buskopplern gleich.

Bei der Auswertung wurde das EtherCat System mit den PhoenixContact Busmodulen und der Beckhoff SPS nicht betrachtet. Die Beckhoff SPS ist ein Evaluation Board ist und besitzt nicht die Funktionalität einer klassischen SPS. Nach dem Überspielen des Programms auf das Evaluation Board geht das System nach einer Stunde in eine Art Sleep Modus. Danach werden keine Prozessdaten mehr an die Buskoppler übertra-

gen. Aus diesem Grunde konnten zwar Betrachtungen der Zählerentwicklung gemacht werden, aber eine Aufzeichnung über 3 Tage war nicht möglich. Trotzdem konnten Erfahrungen mit der Arbeitsweise von EtherCat und mit dem Umgang von TwinCat gemacht werden. (siehe Kapitel 5.6)

Der Aufbau der Testszenarios hat gezeigt, dass mache Systeme den dekadischen Zähler mit einer sehr schnellen Programmabarbeitung realisieren. Hier sticht besonders das System mit der S7- Microbox hervor. Es sollte hierbei aber auch berücksichtigt werden, wie schnell die I/O Devices aktualisiert werden können. Dabei ist zu beachten, dass die Antwortzeiten eines Profinet Slaves nicht zwingend bedeuten, dass diese auch aktualisiert wurden. Es kann durchaus sein, dass ein Master alle 1ms neue Datensätze an den Slave senden kann, aber die I/O Zustände intern nur alle 5ms aktualisiert werden.

7 Fazit

Die Diplomarbeit beschäftigte sich mit der Grundlagenforschung von industriellen Kommunikationsnetzwerken und Steuerungssystemen. Sie umfasst eine Untersuchung der Performance und der Flexibilität von verschiedenen modernen Steuerungsmodellen. Es wurden Betrachtungen bezüglich des Feldbusses sowie der CPU Performence auf Grundlage desselben Zählprogramms realisiert. Damit ist die Basis für die Vergleichbarkeit der unterschiedlichen Systeme gegeben. Dieses Programm sollte auf jedem System zu den gleichen Bedingungen umgesetzt werden. Eine Ausnahme bildet lediglich das B&R System, welches in Structed Text programmiert ist. Das Programm wurde zwar in einer anderen Programmiersprache umgesetzt, realisiert aber den gleichen Funktionsumfang.

Für die Umsetzung dieser Arbeit wurden zunächst die unterschiedlichen Systeme aufgebaut und in die verschieden Entwicklungsumgebungen implementiert. Die Systeme wurden mit ihren für die Performance entscheidenden Parametern vorgestellt und im Anschluss wurde auf die Implementierung näher eingegangen. Hierbei wurde vor allem auf Probleme, Hindernisse und Herangehensweisen näher eingegangen. Die visuelle Darstellung, Aufzeichnung, Archivierung und Auswertung der Ergebnisse erfolgte mittels eines Monitoringsystems.

Auf Basis der vom Monitoringsystem gelieferten Ergebnisse, sowie der theoretischen Vorbetrachtungen, konnten Schlussfolgerungen bezüglich der zu betrachtenden Punkte gezogen werden. Hierbei wurde zunächst festgestellt, dass die Performence sehr stark von der CPU Leistungsfähigkeit abhängig ist. Da das entwickelte Programm relativ wenig CPU Leistung benötigt, sind die leistungsstarken Systeme mit einer Minimalzykluszeit bei diesem Test deutlich erkennbar. Aus dem Grund der wenigen Nutzdaten bei dem Datenaustausch und der unterschiedlichen Leistungsdaten der Bussysteme, wurden Performenceunterschiede der Feldbussysteme ersichtlich. Bei der Betrachtung der Performance in Bezug auf herstellerinhomogene Systemaufbauten hat sich herausgestellt, dass hierbei die Aktualisierungszeit der Steuerung im herstellernahen Zusammenspiel der Buskoppler eine entscheidende Rolle spielt. Dabei ist die Aktualisierungszeit, die die Steuerung sowie der Buskoppler unterstützt, entscheidend.

Die verschiedenen Steuerungsarten, die bei diesem Performencevergleich betrachtet wurden, zeigten sehr unterschiedliche Ergebnisse. Der Grund hierfür ist die Entwicklung der Steuerungen für sehr unterschiedliche Einsatzgebiete. Hierzu zählt beispielsweise das TURCK System. Dieses System ist eine sehr gute Kompaktlösung für mittlere Automatisierungsanwendungen. Bei diesem System sticht besonders die universelle

Einsetzbarkeit hervor. Durch die Multiprotokoll Unterstützung kann das System sehr einfach in bestehende Anlagen integriert werden.
Weiterhin haben die Systeme der Firma B&R bei dieser Untersuchung eine sehr hohe Performance erzielt, und können somit in sehr unterschiedliche Automatisierungsgebiete eingesetzt werden. Ein weiterer Vorteil dieses Systems ist die Entwicklungsumgebung “ AutomationStudio“. Das AutomationStudio bietet die Möglichkeit, Steuerungsabläufe, Kommunikationssysteme, reglungstechnische Probleme, Sicherheitstechnik, Bewegungen und Visualisierungen innerhalb nur einer Entwicklungsumgebung zu programmieren.

Diese Arbeit soll einen Grundstein für weiterführende Untersuchungen und Projekte gelegt haben. Hierbei sollte vor allen der ganze Testszenario mit einen umfangreicheren Programm durchgeführt werden. Dieses Programm sollte dem typischen praxisnahen Mengengerüst entsprechen, für das die Steuerungen ausgelegt sind. Ein typisches Programm für Automatisierunganlagen hat ein Mengengerüst von 1000 Zeilen Programmcode mit einem entsprechenden Mix aus Bit-, Word-, Festpunkt- und Gleitpunktoperationen. Ein solches Programm könnte in folgenden Projekten umgesetzt werden. Hiermit würden die Systeme mit einer Minimalzykluszeit nicht mehr benachteiligt sein.

Diese Arbeit zeigte unter anderem, dass bei der Umsetzung von kleinen Automatisierungsaufgaben, die unter dem typischen Mengengerüst eines Programms implementiert werden, es eine entscheidende Rolle spielt, welcher Feldbus und welche dezentrale Peripherie verwendet wird und wie die interne Konfiguration der Steuerung die Performance des Gesamtsystem beeinflusst.

Literaturverzeichnis

[1] IEC Webstore (2015): “http://www.beuth.de/de/norm/din-en-61131-3/67272302“. (13.05.2015).

[2] Grundwissen SPS- Technik: “https://homepages.thm.de/ mlth53/wp-content/ uploads/2010/04/sps.pdf“. Technische Hochschule Hessen.

[3] Modicon Programmierhandbuch: “http://www.berger-positec.at/files/ programmierhandbuchm238.pdf“. SchneiderElectric. (05/2010).

[4] Automation: Hard, Soft, Slot - SPS-Konzepte im Vergleich, “http:// www.automationnet.de/index.cfm?pid=1702pk=66027.VVSkfTeKO5-“. (14.06.2015).

[5] IT- Automation: “http://www.it-production.com/index.php?seite=einzel_artikel _ansicht&id=60920“. (14.06.2015).

[6] Norbert Becker (2. Auflage 2014): “Automatisierungstechnik“. Würzburg: Vogel Buchverlag.

[7] EPSG Office (2nd Edition): “INDUSTRIAL ETHERNET FACTS“. Berlin.

[8] PROFIBUS Nutzungsorganisation e.V.(PNO): “PROFINET Systembeschreibung – Technologie und Anwendung“. Karlsruhe.

[9] Gehard Schnell, Bernhard Wiedermann (8. Auflage, 2012): “Bussysteme in der Automatisierungs- und Prozesstechnik“. Wiesbaden: Springer Vieweg.

[10] Sven Schmeisser, Dietmar Römer: “Kommunikation mit Ethernet“. Fakultät Elektro- und Informationstechnik. Hochschule Mittweida.

[11] Anybus: Modbus TCP, “http://www.anybus.de/technologie/modbustcp.shtml“. (15.05.2015).

[12] Frithjof Klasen, Volker Oestreich, Michael Volz (2010-11): “Industrielle Kommunikation mit Feldbus und Ethernet“. Berlin: VDE Verlag.

[13] Ethernet Powerlink (EPSG): “http://www.ethernet-powerlink.org/de/ powerlink/technologie/wie-funktioniert-powerlink/“. (16.05.2015).

[14] SERCOS – the automation bus: “http://www.sercos.de/technologie/sercos -iii/topologie/“. (17.05.2015).

[15] ElektronikNet: “http://www.elektroniknet.de/automation/sonstiges/artikel/1405/2/“. (17.05.2015).

[16] Reinhard Langmann (2. Auflage 2010): “Taschenbuch der Automatisierung“. München: Hanser.

[17] Feldbusse: “http://www.feldbusse.de/Profibus/Buszugriffsprotokoll.shtml“. 18.05.2015.

[18] Siemens: “https://mall.industry.siemens.com/mall/ES/WW/Catalog/ Products/10030191“. 20.05.2015.

[19] Siemens: “https://mall.industry.siemens.com/mall/de/WW/Catalog/ Products/10021996?tree=CatalogTree“. 20.05.2015.

[20] Siemens: “https://mall.industry.siemens.com/mall/de/WW/Catalog/ Products/10204209?tree=CatalogTree“. 20.05.2015.

[21] Siemens: “https://mall.industry.siemens.com/mall/de/WW/Catalog/ Products/10204211?tree=CatalogTree“. 20.05.2015.

[22] Wago: “http://www.wago.com/wagoweb/documentation/750/ger_dat/d07500370_ 00000000_0de.pdf“. 20.05.2015.

[23] HILSCHER: “http://www.hilscher.com/fileadmin/cms_upload/de.pdf“. 21.05.2015.

[24] TURCK: “http://www.turck.de/static/media/downloads/more_TURCK_2_09.pdf“. 21.05.2015.

[25] Elektronik Vogel: “http://www.elektrotechnik.vogel.de/feldkommunikation/ articles/386974/“. 21.05.2015.

[26] HILSCHER: “http://www.elektrotechnik.vogel.de/feldkommunikation/ articles/386974/“. 21.05.2015.

[27] Computer- Automation: “http://www.computer-automation.de/ steuerungsebene/bedienen-beobachten/artikel/73725/2/“. 21.09.2015.

[28] PhoenixContact: “https://www.phoenixcontact.com/online/portal/de?urile=pxc-oc-itemdetail:pid=2876928“. 21.05.2015.

[29] SPS Magazin: “http://www.sps-magazin.de/?inc=artikel/article_shownr=77699“. 21.09.2015.

[30] PhoenixContact: “https://www.phoenixcontact.com/online/portal/de?urile=pxc-oc-itemdetail:pid=2688459“. 21.05.2015.

[31] B&R: “http://www.br-automation.com/de-at/produkte/steuerungssysteme/x20-system-coated/x20-cpus/x20ccp1586/techdata“. 23.05.2015.

[32] B&R: “https://www.br-automation.com/de-de/produkte/visualisieren-und-bedienen/power-panel-65/“. 23.09.2015.

[33] B&R: “https://www.br-automation.com/de-de/produkte/netzwerke-und-feldbus-module/x2x-link/“. 23.09.2015.

[34] BA Produktionstechnik: “http://www.ba-produktionstechnik.de/attachments/084_Kap1.5_Grundwissen_SPS.pdf“. 24.05.2015.

[35] Matthias Seitz (3. Auflage 2012): “Speicherprogrammierbare Steuerungen für Fabrik- und Prozessautomation“. München: Carl Hanser Verlag.

[36] SIEMENS: “https://mall.industry.siemens.com/mall/DE/WW/Catalog/Products/10012761“. 27.05.2015.

[37] SIEMENS: “https://mall.industry.siemens.com/mall/de/WW/Catalog/Products/5000594?tree=CatalogTree“. 27.05.2015.

[38] SIEMENS: “https://mall.industry.siemens.com/mall/de/WW/Catalog/Products/5000014?tree=CatalogTree“. 27.05.2015.

[39] SIEMENS: “https://mall.industry.siemens.com/mall/de/WW/Catalog/Products/10042911?tree=CatalogTree“. 27.05.2015.

[40] SPS-Magazin: “http://www.sps-magazin.de/?inc=artikel/article_show&nr=90727“. 28.05.2015.

[41] B&R: “https://www.br-automation.com/de-de/produkte/software/“. 02.06.2015.

[42] Industrieanzeiger: “http://www.industrieanzeiger.de/html/portlet/ext/articles-portlet/articles_content/print.jsproupId=22469894&articleId=26093686&companyId=1&accountId=7&version=1.0&p_l_i_d=12515&articlePk=26093685&layId=3&langId=de_DE“. 05.06.2015.

[43] Manfred Popp, Karl Weber(2004): “Der Schnelleinstieg in PROFINET“. Karlsruhe: ProfiBus Nutzerorganisation e.V.

[44] Prof. Dr.-Ing. habil. Habiger, Ernst (2013): “openautomation Fachlexikon 2013/14“. Berlin: VDE VERLAG GMBH.

[45] PhoenixContact (2014):’Anwenderhandbuch UM DE ALX F BK EC’. ’ Inbetriebnahme des Axioline F-Buskopplers für EtherCat’ (23.01.2015).

[46] BECKHOFF (2015): “http://www.infosys.beckhoff.com/index.php“. (24.02.2015).

[47] BECKHOFF (2015): “http://infosys.beckhoff.de/index.php?content=../content/1031/tcplccontrol/html/tcplcctrl_intro.htm&id=“. (24.02.2015).

[48] BECKHOFF (2015): “http://infosys.beckhoff.de/index.php?content=../content/1031/tcplclibstandard/html/tcplclibstandard_ctu.htm“ “. (2.03.2015).

[49] BECKHOFF (2015): “http://infosys.beckhoff.de/index.php?content=../content/1031/tcplclibstandard/html/tcplclibstandard_ton.htm&id. (2.03.2015).

[50] EtherCAT (2015): “http://www.ethercat.org/de/technology.html“. (2.03.2015).

[51] EtherCAT (2015): “http://www.ethercat.de/download/documents/EtherCAT _EAP_DE.pdf“. (2.03.2015).

[52] Feldbusse (2015): “http://www.feldbusse.de/EtherCAT/ethercatprotokoll.shtml“. (2.03.2015).

[53] BECKHOFF (2015): “http://infosys.beckhoff.com/index.php?content=../content/1031/ethercatsystem/html/bt_ethercat_master_cable_redundancy.htm&id=“. (2.03.2015).

[54] BECKHOFF (2015): “download.beckhoff.com/download/Document/ethercat/Development _products/EL9840d_Ver1.2.pdf“. (2.03.2015).

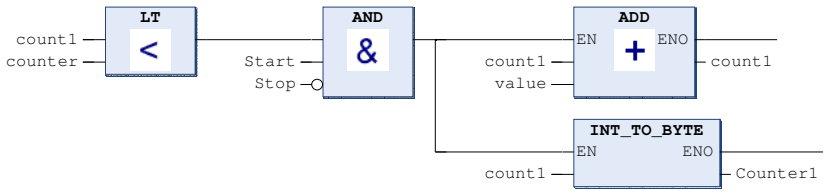
[55] PhoenixContact (2015): “Datenblatt: AXL F BK EC“. PHOENIX CONTACT. 2014-10-13

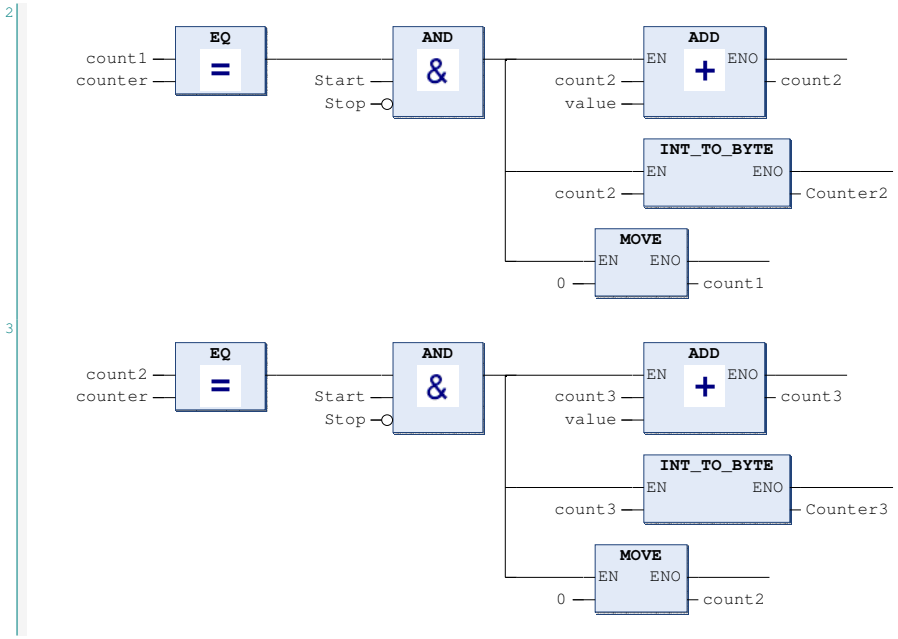
[56] PhoenixContact (2015): “Anwenderhandbuch: AXL F BK EC“. PHOENIX CONTACT. 2014-10-13

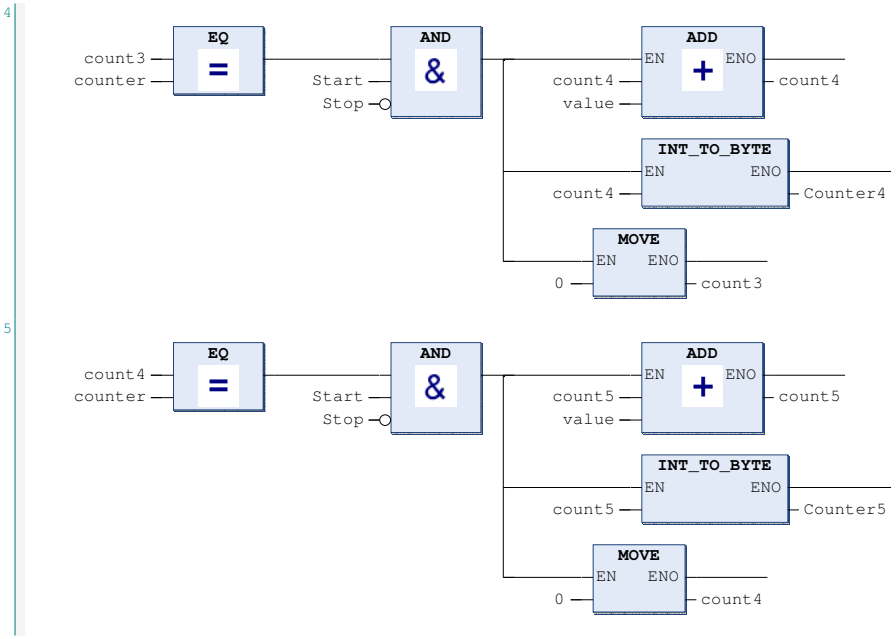
Anhang A: FUP Programm

```
1     PROGRAM  ABAKUS
2     VAR
3         count1   : INT ;
4         count2   : INT ;
5         count3   : INT ;
6         count4   : INT ;
7         count5   : INT ;
8         counter      : INT  := 255 ;
9         value        :  INT  := 1 ;
10
11    END_VAR
12
```

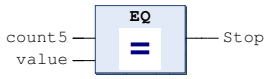
1



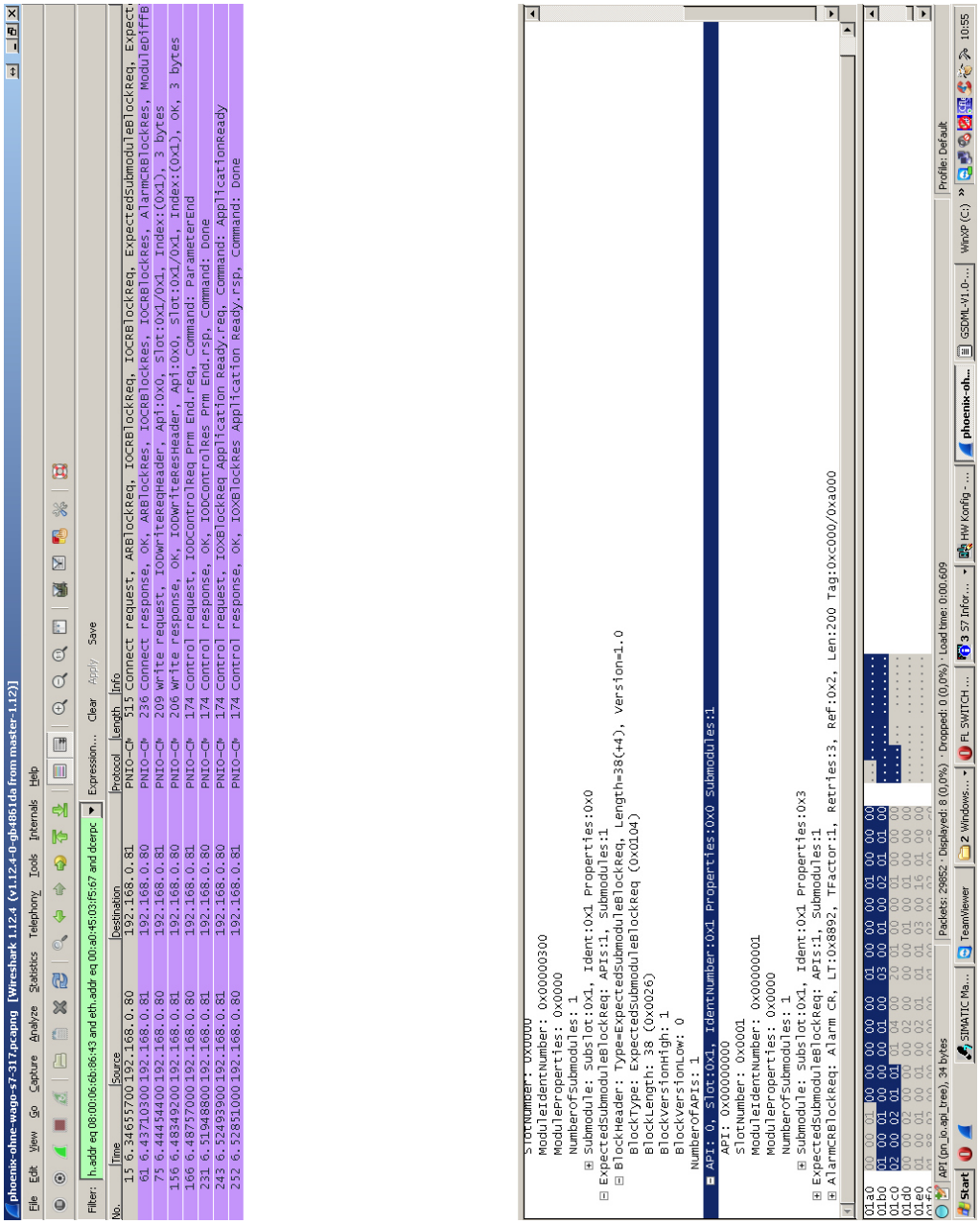

2
EQ
=
count1
counter
AND
&
Start
Stop
ADD
EN
ENO
+
count2
value
count2
INT_TO_BYTE
EN
ENO
count2
Counter2
MOVE
EN
ENO
0
count1
3
EQ
=
count2
counter
AND
&
Start
Stop
ADD
EN
ENO
+
count3
value
count3
INT_TO_BYTE
EN
ENO
count3
Counter3
MOVE
EN
ENO
0
count2


4
EQ
count3
counter
AND
Start
Stop
ADD
EN
ENO
count4
value
count4
INT_TO_BYTE
EN
ENO
count4
Counter4
MOVE
EN
ENO
0
count3
5
EQ
count4
counter
AND
Start
Stop
ADD
EN
ENO
count5
value
count5
INT_TO_BYTE
EN
ENO
count5
Counter5
MOVE
EN
ENO
0
count4

6

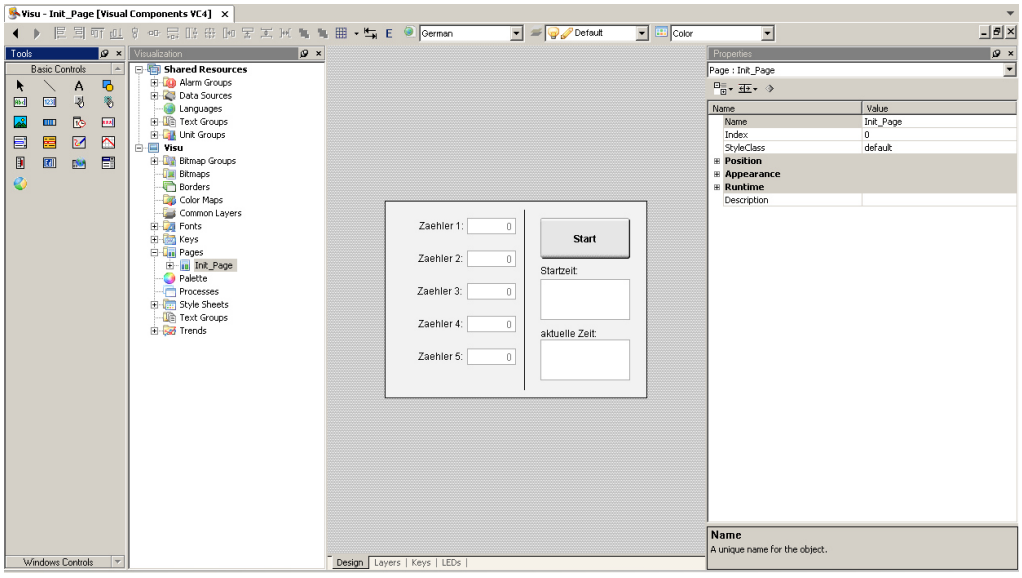


Anhang B: Wireshark- Mitschnitt für Anlauf von S7-317

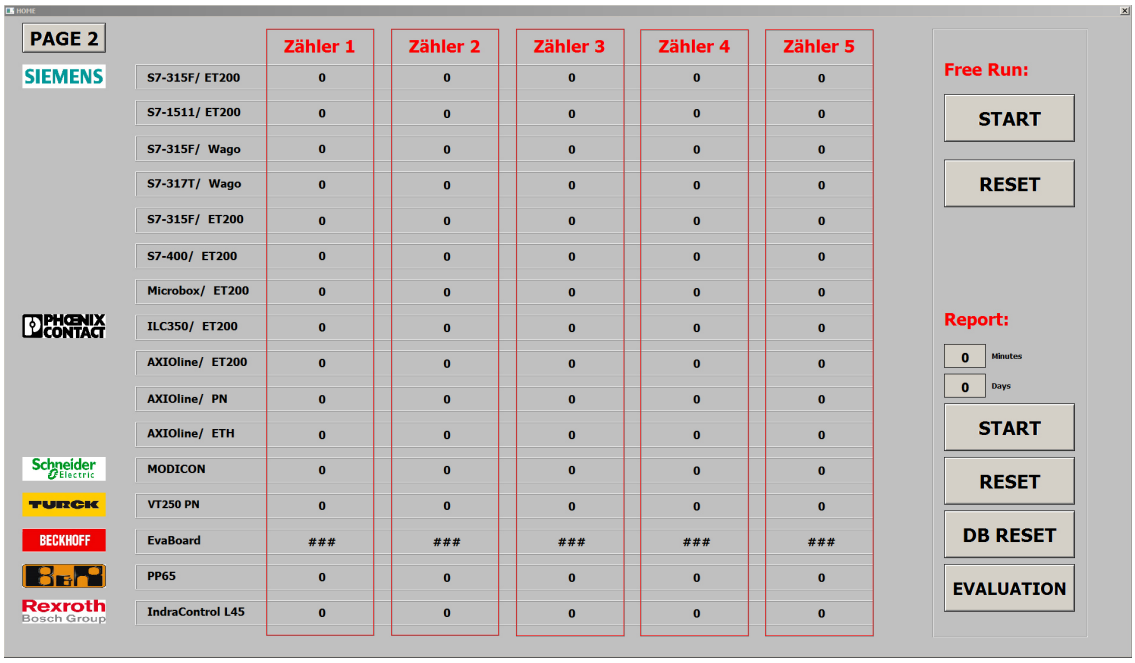


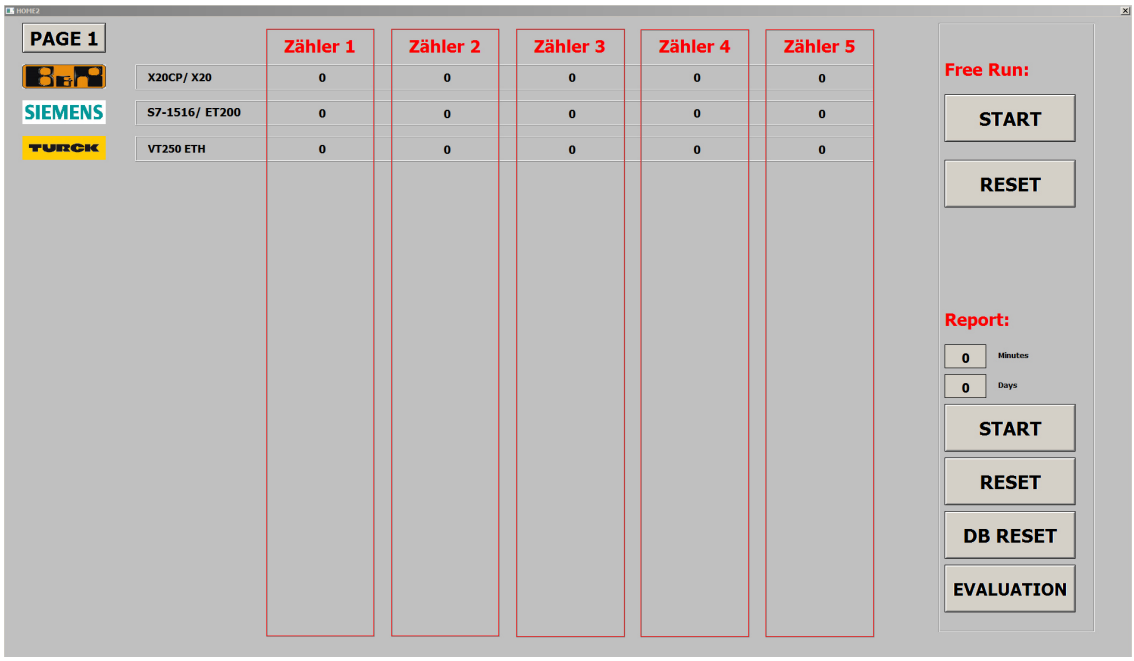
Anhang C: OVIS Visualisierung für VT250

Anhang D: B&R Visualisierung PowerPanel 65



Anhang E: Visualisierungen des Monitoringsystems





AUSWERTUNG

		Zähler 2	Zähler 3	Zähler 4	Zähler 5
SIEMENS	S7-315F/ ET200	0 sec	0 min	0 h	0 d
	S7-1511/ ET200	0 sec	0 min	0 h	0 d
	S7-315F/ Wago	0 sec	0 min	0 h	0 d
	S7-317T/ Wago	0 sec	0 min	0 h	0 d
	S7-315F/ ET200	0 sec	0 min	0 h	0 d
	S7-400/ ET200	0 sec	0 min	0 h	0 d
	Microbox/ ET200	### sec	0 min	0 h	0 d
PHOENIX CONTACT	ILC350/ ET200	0 sec	0 min	0 h	0 d
	AXIOline/ ET200	0 sec	0 min	0 h	0 d
	AXIOline/ PN	0 sec	0 min	0 h	0 d
	AXIOline/ ETH	0 sec	0 min	0 h	0 d
Schneider Electric	MODICON	0 sec	0 min	0 h	0 d
TURCK	VT250 PN	0 sec	0 min	0 h	0 d
BECKHOFF	EvaBoard	### sec	### min	### h	### d
B&R	PP65	0 sec	0 min	0 h	0 d
Rexroth Bosch Group	IndraControl L45	0 sec	0 min	0 h	0 d

DB write

BACK

AUSWERTUNG2

		Zähler 2	Zähler 3	Zähler 4	Zähler 5
B&R	X20CP/ X20	0 sec	0 min	0 h	0 d
SIEMENS	S7-1516/ ET200	0 sec	0 min	0 h	0 d
TURCK	VT250 ETH	0 sec	0 min	0 h	0 d

DB write

BACK

Anhang F: Fotografie

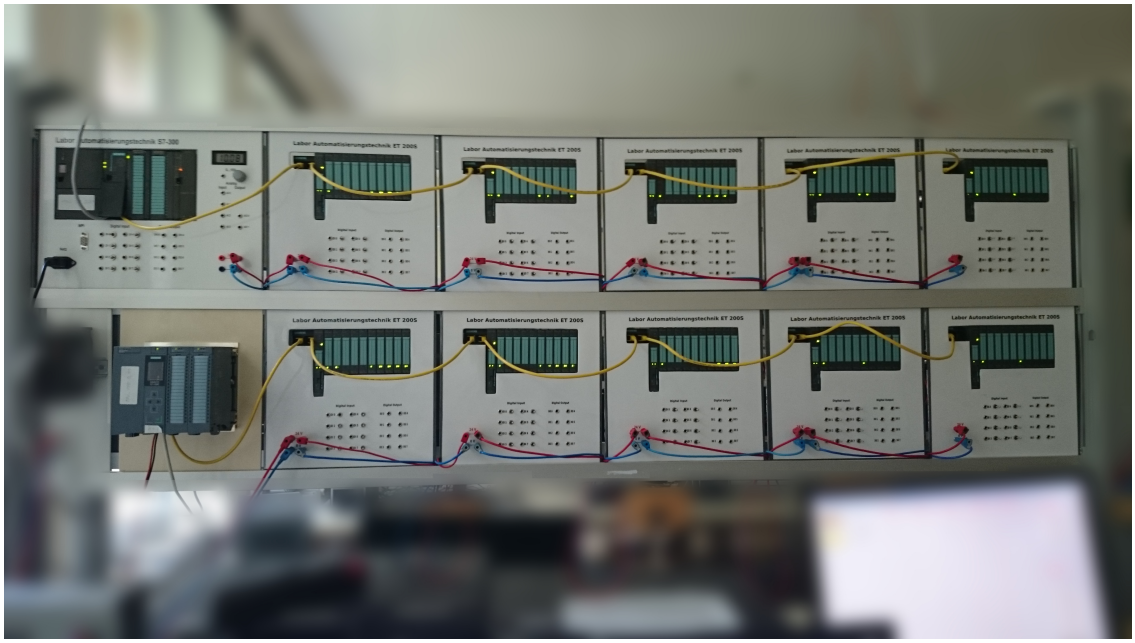


Abbildung F.1: System 1, 2 (v. O.)

Abbildung F.2: System 3

Abbildung F.3: System 4

Abbildung F.4: System 5

Abbildung F.5: System 6, 12

Abbildung F.6: System 7

Abbildung F.7: System 8

Abbildung F.8: System 9

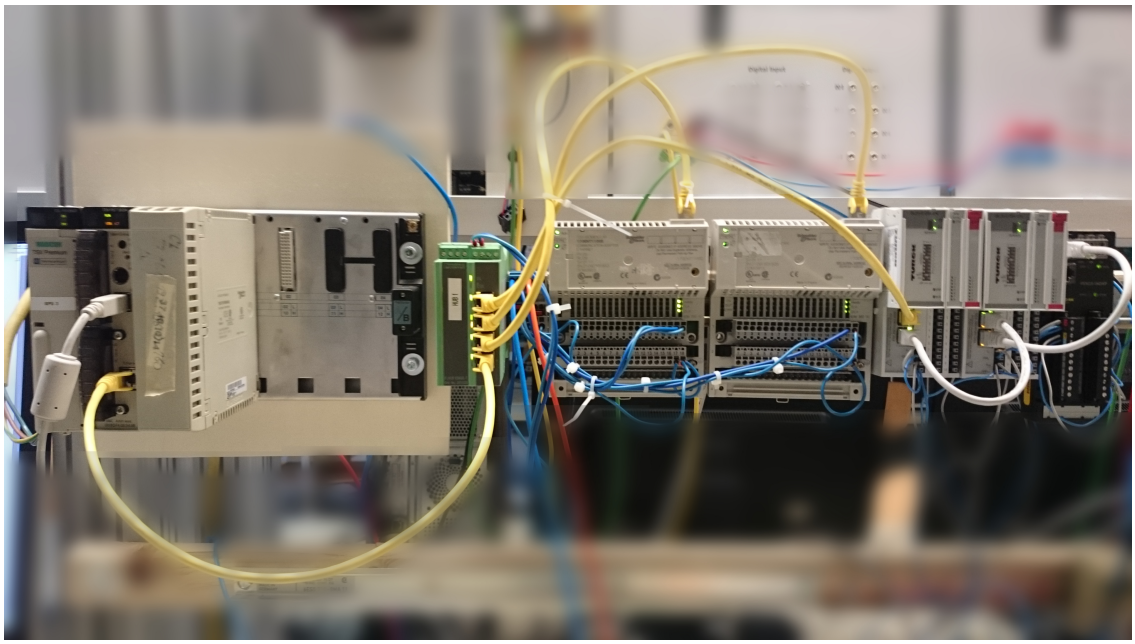


Abbildung F.9: System 10

Abbildung F.10: System 11

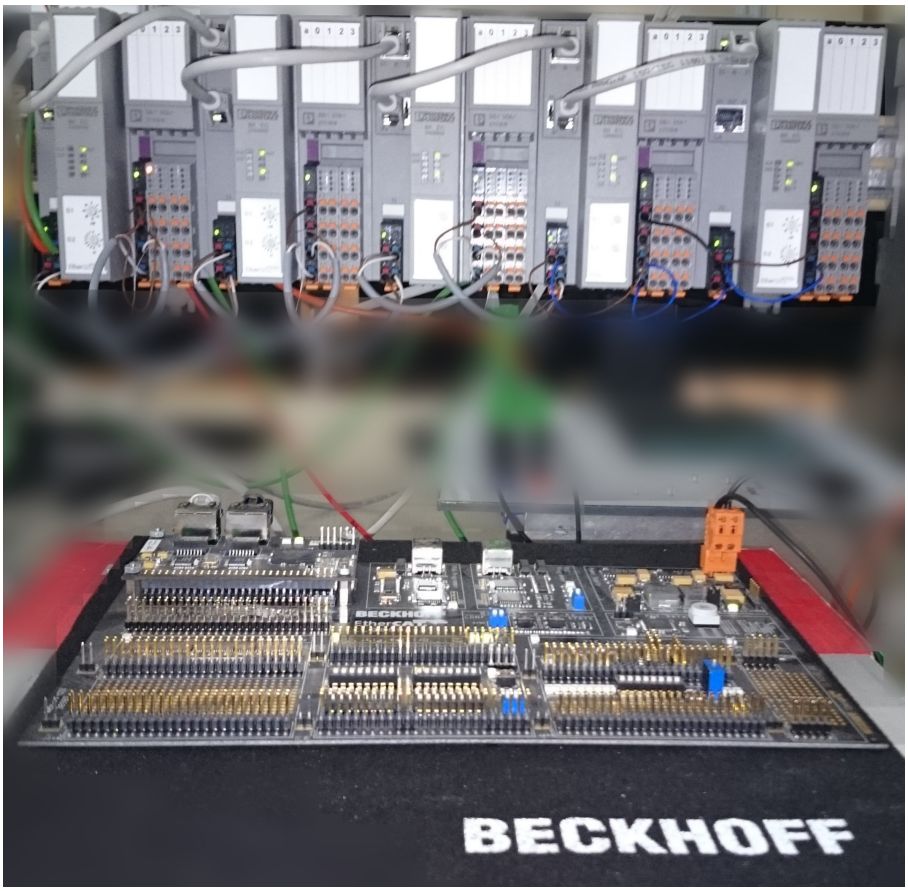


Abbildung F.11: System 13

Abbildung F.12: System 14

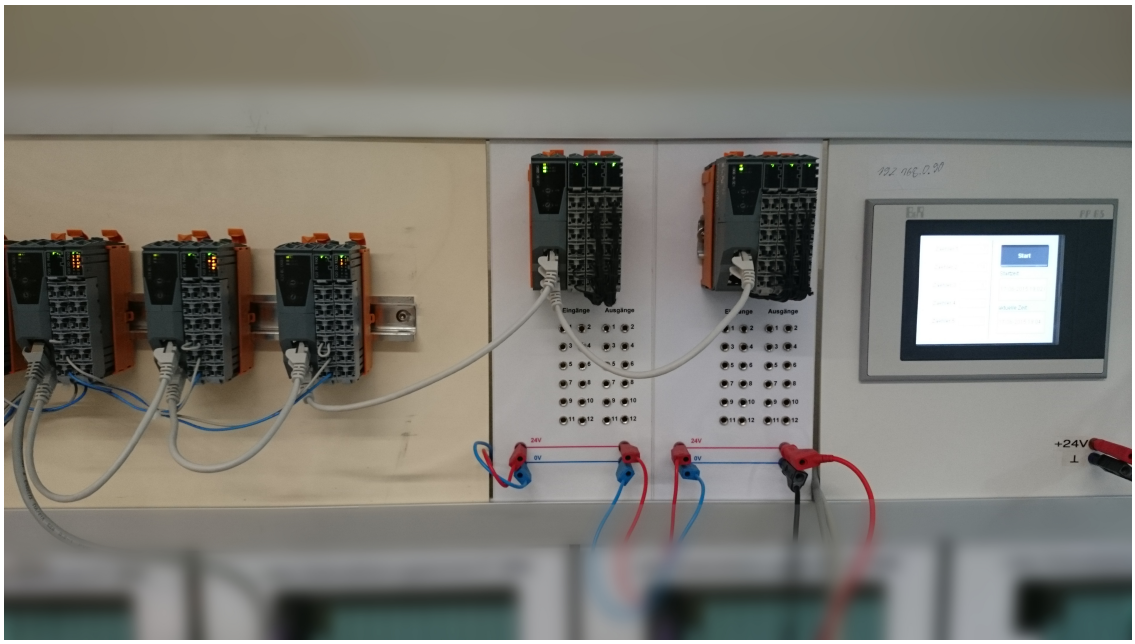


Abbildung F.13: System 15

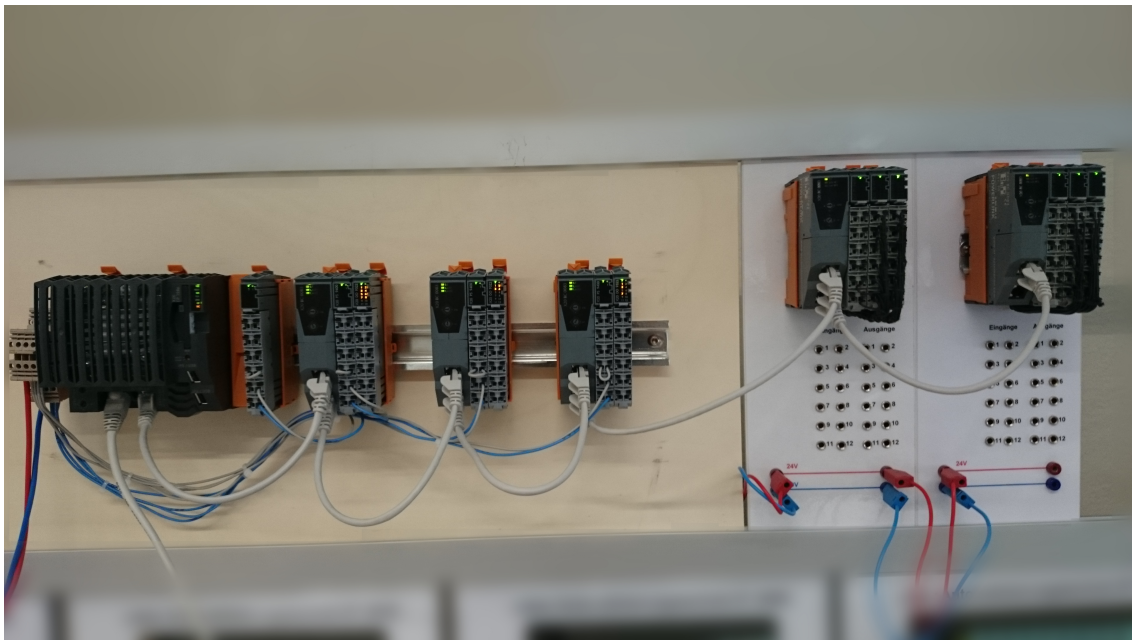


Abbildung F.14: System 16

Abbildung F.15: System 18

Abbildung F.16: System 17, 19 (v. O.)

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 23. Juni 2015