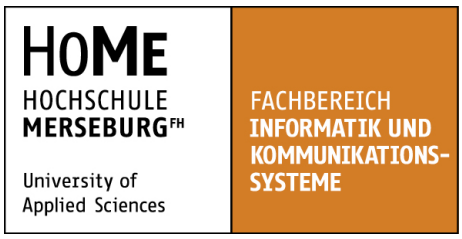


Zur Erlangung des Grades

eines

Bachelor of Engineering (B. Eng.)

von Herrn Emanuel Klann

geboren am:	13. August 1992
in:	Nordhausen
Studiengang:	Medien-, Kommunikations- und Automationssysteme
Matrikelnummer:	18721
Abgabedatum:	15. Oktober 2014

Thema:	Entwurf und Programmierung einer Automatisierungslösung für ein Bürogebäude
Betreuer:	Prof. Dr.-Ing. Peter Helm

Inhaltsverzeichnis

Abkürzungsverzeichnis

AS	Ablaufsprache
AWL	Anweisungsliste
BA	Busankoppler
BACnet	Building Automation and Control Networks
BCI	BatiBUS Club International
BU	BaseUnit
CM	Kommunikationsbaugruppe / Communication Module
CoDeSys	Controller Development System
CPU	Prozessor / Central Processing Unit
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
DB	Datenbaustein
DC	Gleichstrom / Direct Current
DIN	Deutsches Institut für Normung
DP	Dezentrale Peripherie
EHSA	European Home Systems Association
EIB	Europäischer Installationsbus
EIBA	European Installation Bus Association
EMA	Einbruchmeldeanlage
EN	Europäische Norm
FB	Funktionsbaustein
FC	Funktion
FUP	Funktionsplan
HLK	Heizung, Lüftung und Klima
HMI	Human-Machine Interface
ID	Identifikator
IEC	International Electrotechnical Commission
I/O	Input/Output

IM	Interfacemodul
IP	Internet Protocol
IRT	Isochrone Real Time
ISO	International Standardization Organisation
KNX	Konnex
KOP	Kontaktplan
LED	Leuchtdiode / Light Emitting Diode
LON	Local Operating Network
MC	Memory Card
MMS	Mensch-Maschine-Schnittstelle
OB	Organisationsbaustein
OSI	Open Systems Interconnection
PC	Personal Computer
PN	PROFINET (Process Field Ethernet)
PS	Stromversorgung / Power Supply
PVA	Photovoltaikanlage
RFID	radio-frequency identification
RT	Real Time
RTU	Remote Terminal Unit
S7	STEP 7
SCADA	Supervisory Control and Data Acquisition
SCL	Structured Control Language
SM	Signalbaugruppe / Signal Module
SPS	Speicherprogrammierbare Steuerung
ST	Strukturierter Text
TCP	Transmission Control Protocol
TIA	Totally Integrated Automation
TM	Technologiebaugruppe / Technology Module
VBA	Visual Basic for Applications
WinCC	Windows Control Center

1 Einleitung

1.1 Einführung

Die Gebäudeautomation stellt in der heutigen Zeit ein zentrales Thema der Automatisierungstechnik dar. Im Wohnungs- und Zweckbau hat der Automatisierungsgrad beachtlich zugenommen. Auch in Zukunft wird dieser weiter zunehmen, da sich Bewohner und Betreiber immer mehr Komfort, Sicherheit und Wirtschaftlichkeit ihrer Gebäude wünschen. Hierbei kann nicht nur der Nutzer schlichtweg Geld sparen, auch die Gesellschaft profitiert von den Vorteilen moderner Gebäude. Ein Hauptziel der Gebäudeautomation ist Energieeinsparung sowie ein intelligentes Energiemanagement. So kann sie gezielt an Stellen verbraucht und eingespart werden, an denen es sich als sinnvoll erweist. In Zweckgebäuden wird eine hohe Flexibilität in Bezug auf Änderungen in der Raumnutzung erwartet. Dies wird mithilfe der Gebäudesystemtechnik gewährleistet und umgesetzt. Ein modernes System wie Konnex (KNX), Local Operating Network (LON) oder Building Automation and Control Networks (BACnet) verbindet viele grundlegende Funktionen über ein Bussystem. Immer häufiger werden aber auch Standard Industriesteuerungen in diesem Aufgabenbereich eingesetzt und sogar im professionellen Maße von Unternehmen angeboten. Vorteile dieser Automatisierungslösung liegen oftmals im Preis und in der Flexibilität bei der Programmierung. Denn dort können Funktionalitäten individuell angepasst und optimiert werden. Die WAGO Kontakttechnik GmbH & Co. KG bietet verschiedene Bausteinbibliotheken im Bereich der Gebäudeautomatisierung für Ihre Steuerungen an. Natürlich ist ein wesentlicher Mehraufwand in Verkabelung und vor allem Programmierung zu leisten. Dies wird allerdings von Softwareentwicklern, deren Aufgabengebiet die tägliche Arbeit mit speicherprogrammierbaren Steuerungen (SPS) beinhaltet, oftmals gern in Kauf genommen. Abweichend von der konventionellen Gebäudesystemtechnik basierend auf herkömmlich genannten Kommunikationssystemen, ist es also auch denkbar und nicht ungewöhnlich eine Industriesteuerung in diesem Anwendungsgebiet zum Einsatz zu bringen. Die Dezentralisierung der Automatisierungsfunktionen ist vor allem bei großen Gebäuden sinnvoll und verbessert zusätzlich die Umstände des Verkabelungsauf-

wandes.

1.2 Aufgabenstellung

Beim Erweiterungsbau der Büro- und Werkstattflächen der Firma HothoData GmbH soll der Raumkomfort durch den Einsatz einer intelligenten Steuerung erhöht werden. Um gleichzeitig die Kernkompetenzen der Firma demonstrieren zu können, soll diese Hausautomatisierung mit einer SPS der Firma SIEMENS umgesetzt werden. Hierbei werden gewisse Schwerpunkte im Vornherein festgelegt. Zu realisieren ist eine Lichtansteuerung mit Raum- und Außenhelligkeitsbezug, eine kompakte Jalousiesteuerung mithilfe verschiedener Sensorik, anwesenheitbedingtes Zu- und Abschalten der Arbeitsplatz-Elektronik, eine Heizungssteuerung für die einzelnen Büroräume und abrundend eine passende und anschauliche Visualisierung mit Bedienlösung. Hierfür soll ein Programm basierend auf der internationalen Norm IEC 61131 entstehen. Im Flur des Neubaus wird ein Touchpanel verbaut, das für grundlegende Beobachtung und Bedienung genutzt werden soll. Außerdem findet an diesem zentralen Punkt der Firma eine Anmeldung der Ingenieure für die Registrierung der Anwesenheit und die erwähnte Freigabe der Arbeitsplatz-Elektronik statt. Alle benutzerspezifischen Funktionen können dort eingestellt und Informationen abgerufen werden. Auf dem Personal Computer (PC) sollen gleiche Möglichkeiten der Raumanpassung möglich sein. Hinzu kommt eine sinnvolle Einbindung der, mit dem Neubau hinzugekommenen, Photovoltaikanlage (PVA) im Programm und der Visualisierung.

1.3 Zielsetzung

Das Ziel dieser Arbeit stellt den Versuch, mithilfe unkonventioneller Automatisierungstechnik eine Gebäudeautomatisierung zu realisieren, dar. Des Weiteren ist die Zufriedenheit der Nutzer ein zentrales Thema, da die Arbeitsmoral der Ingenieure nicht durch unnötige Funktionalitäten oder allgemeine Fehlfunktionen negativ beeinflusst werden darf. Das Programm muss unter anderem auch für eine eventuelle Umstrukturierung der Raumaufteilung oder anderweitige Veränderungen geeignet sein. Es wird also Wert auf Individualität der Struktur und Bausteine gelegt. Ein Einarbeiten in die Programmlogik muss unkompliziert und zügig von statten gehen können. Damit dies gewährleistet ist, werden firmeninterne

programmiertechnische Gepflogenheiten berücksichtigt und in den Programmablauf eingepflegt. Am Ende soll die Steuerung in der Lage sein, alle Komfortfunktionen auf Wunsch auszuführen oder die übliche Handbedienung zu gewährleisten.

Unter Einbeziehung der genannten Bedingungen stehen im ersten Abschnitt dieser Arbeit theoretische Grundlagen sowie erste Vorbetrachtungen im Vordergrund. Hierbei werden benötigte Soft- und Hardware erläutert und mögliche Systemvarianten aufgeführt. Des Weiteren ist ein im Vorfeld entwickeltes Konzept dargelegt, welches als funktionelle Orientierung während der Programmerstellung dienen soll. Im Hauptteil werden Probleme und deren Lösung erläutert, sowie fertige Programmabschnitte in ihrer Wirkung detailliert erklärt. Aufgegriffen werden alle Programmteile und die Visualisierung. Außerdem wird das fertige Programm, welches die Automatisierungslösung der genannten Aufgabenstellung darstellt, beschrieben. Im letzten Abschnitt dieser Arbeit wird auf den aktuellen Projektstand hingewiesen.

2 Grundlagen

2.1 Hausautomatisierung

2.1.1 Beleuchtung

Es hat sich bewährt, bei modernen Beleuchtungssteuerungen nicht nur eine gewöhnliche Zeitschaltung in die Automatikfunktion zu integrieren, sondern auch Helligkeitssensorik zu verwenden. Meist ist eine Konstantlichtregelung für die Raumautomation unverzichtbar. Wesentlich komplexer fällt eine lichtgeführte Raumhelligkeitsanpassung aus. Diese stellt Vorgaben an die Leuchtmittel und setzt Dimmaktorik voraus. Grundsätzlich sind alle Glüh- und Halogenlampen über Phasenan- oder Phasenabschnittdimmer dimmbar. Jedoch trifft dies für Leuchtstoffröhren, Energiesparlampen und Leuchtdioden (LED) nur unter Zuhilfenahme von zusätzlichem Aufwand zu, weshalb diese meist nicht im Privatgebrauch zum Einsatz kommen. Je nach gewünschter Funktionalität und gefordertem Komfort können alle denkbaren Leuchten verbaut werden. Eine Anwendungsmöglichkeit stellt ein belegungsabhängiges Beleuchten oder Freigeben über Präsenztaster oder -melder, die den Energieverbrauch reduzieren, dar. Denkbar ist diese Anwendung beispielsweise in Durchgangsbereichen im Wohnhaus, als Beleuchtungsfreischaltung des Arbeitsplatzes nach dem Einstechen oder sogar in Verbindung mit einem Fremdsystem zur Freischaltung der Beleuchtung in Bereichen eines Gebäudes, z.B. Hotelzimmern oder Abteilungen. Durch Lichtszenen können verschiedene Stimmungen induziert werden, wodurch dem Raum zusätzlich Atmosphäre verliehen wird. Hierbei könnten Einstellungen wie „Fernsehen“ oder „Arbeiten“ definiert werden, welche gewisse Leuchten ansteuern oder gegebenenfalls dimmen. Beim Einsatz dieser Aspekte steht der Komfort sowie das Wohlbefinden des Nutzers an oberster Stelle. Weiterhin kann eine richtige und situationsbedingte Beleuchtung die Sicherheit und das Arbeiten in automatisierten Gebäuden erheblich verbessern. Zum einen ist eine Kopplung mit der Einbruchmeldeanlage (EMA) denkbar, welche Möglichkeiten zur Schaltung einer voreingestellten Lichtszene im Einbruchsfall bietet oder für die Verriegelung aller Bediengeräte sorgt. Außerdem ist mithilfe eines Auslösens von Rauchmeldern oder Notschaltern nicht ausgeschlossen, eine Notbeleuchtung

zu schalten, die auch bei Dunkelheit Notwege und Sammelplätze gut sichtbar macht. Doch nicht nur Sicherheitsaspekte sind von großer Bedeutung. Auch das Erleichtern der Arbeit durch Zusatzfunktionen in automatisierten Gebäuden gilt als erstrebenswert. Zum Beispiel könnte eine Putzlichtschaltung zur Erhöhung der Raumhelligkeit oder eine individuell anpassbare Arbeitsplatzbeleuchtung, die sich für das Lesen und Schreiben besonders hell und für Computerarbeit dunkler einstellt, realisiert werden. Oftmals sind Diffuslichtregelungen im Zusammenspiel mit Sonnenstand und Jalousie durch die Verstellung der Lamellen das Ziel einer komplettierten automatischen Beleuchtungssteuerung [MHH10].

2.1.2 Beschattung

Eine automatisierte Steuerung von Beschattungselementen im Außenbereich steht im Zusammenhang mit Lichtverhältnissen der Außenwelt, der Raumhelligkeit und Windeinwirkungen bei Windböhen o. ä. Um eine Beschädigung der Komponenten zu verhindern, hat das deutsche Institut für Normung (DIN) mit der europäischen Norm (EN) DIN EN 13659 ein Hochfahren bei anliegender Maximalgeschwindigkeit des Windes sowie eine Reaktionsverzögerungszeit festgelegt. Diese Zeiten unterscheiden sich je nach Typ (Jalousie, Rolladen, Markisen etc.), Größe, Lamellenform und Anbauart. Im Gebäudeinneren ist eher eine Verwendung als Sichtschutz oder Verdunkelung von z.B. Glasräumen denkbar. Oberstes Anliegen an die Automatisierung von Außenjalousien ist der Komfort und die Energieeinsparung, die mit der richtigen Jalousiefahrt einhergehen. Beim Aspekt des Komforts wird beispielsweise Wert auf eine Beschattung bei tiefstehender Sonne, Abdunkelung beim Fernsehen, Zeitschaltung zum Wecken des Nutzers oder optimaler Raumhelligkeit für verschiedene Raumanwendungen gelegt. Doch auch die Bedeutung des Energiesparens nimmt, unter anderem aufgrund stetig zunehmender Energiekosten, rapide zu. Grundsätzlich kann durch eine logische Jalousieführung tagsüber der Beleuchtungsaufwand optimiert werden. Dies ist nicht nur im Sinne des Energiemanagements, sondern auch des Benutzers, da Tageslicht als angenehmer empfunden wird. Allerdings dient die Sonne nicht nur dem Zweck der Beleuchtung, sondern kann auch als Wärmequelle genutzt werden. Einerseits kann im Winter das Auskühlen des Raumes mit ihrer Hilfe gemindert werden, andererseits gilt es das unkontrollierte Aufheizen im Sommer zu vermeiden. Eine optimale Jalousieführung ersetzt damit zwar keine Klimaregelung, bringt

aber Einsparungen im Energieaufwand der Klimageräte mit sich.

2.1.3 Heizungs- und Klimaanlagen

Der größte Teil der verbrauchten Energie im alltäglichen Haushalt wird in die Raumtemperierung investiert. Das individuelle Behaglichkeitsgefühl des Menschen steht dabei im Vordergrund. Temperatur, Qualität, Feuchtigkeit sowie die Bewegung der Luft mit der Temperatur der Wände im Verhältnis zur Tätigkeit sind ausschlaggebend für das Wohlbefinden des Nutzers. Die Automatisierung der Klima- und Heizkomponenten ist energiesparend, aber ohne Verzicht auf Behaglichkeit und Komfort zu realisieren. Im Bereich Heizung, Lüftung und Klima (HLK) sind verschiedene Systeme einsetzbar und individuell umzusetzen. KNX, LON und BACnet bieten Lösungen für eine solche Realisierung an und werden kontinuierlich weiterentwickelt. Grundsätzlich sieht eine Heiz- und Klimasteuerung eine Raumtemperaturregelung unter möglicher Anwesenheitseinbeziehung mithilfe der Vorgabe eines Sollwertes vor. Eine typische Regelung ist die Einzelraumregelung. Diese kann durch Auswerten der Sensorwerte von Temperatur und Anwesenheit (Bewegungsmelder, Präsenzmelder) in Verbindung mit möglichen Zeitprofilen (Tag, Nacht, Winter, Sommer,. . .) je nach Nutzung eine gewünschte Raumtemperatur vorgeben, um den Energiebedarf so niedrig wie möglich halten zu können [KHS09].

2.2 Kommunikation

Kommunikation stellt in allen Bereichen der Technik ein zentrales Thema dar. Sie ist die Schnittstelle zwischen verschiedensten Teilnehmern und verantwortlich für das Verstehen der zu übertragenden Information. Offene Kommunikationssysteme werden durch das sog. ISO/OSI-Referenzmodell (Open Systems Interconnection) definiert. Dabei wird die Kommunikation abstrakt in sieben Ebenen untergliedert, wobei die entstehenden Schichten die Art und Weise der Busphysik, den Datentransport, -zugriff und -austausch sowie auszuführende Applikationen des Anwenders beschreiben. Aufgrund dessen wird dieses Modell auch als OSI-Schichtenmodell (vgl. Tabelle 2.1) bezeichnet [SW12, Seite 8 ff.].
Die populärste Datenkommunikation innerhalb eines Rechnernetzes findet über das in der Bürowelt etablierte Ethernet-TCP/IP-Bussystem (Transmission Control Protocol / Internet

Tabelle 2.1: OSI Schichten

Schicht	Bezeichnung (DE)	Bezeichnung (EN)	Aufgabe
1	Bitübertragungsschicht	Physical Layer	Physikalische Übertragung von Signalen
2	Sicherungsschicht	Data Link Layer	Datenübertragung
3	Vermittlungsschicht	Network Layer	End-zu-End-Verbindung zwischen Rechnern
4	Transportschicht	Transport Layer	Sichere Verbindung herstellen
5	Steuerungsschicht	Session Layer	Verbindung über längere Zeiträume aufrechterhalten
6	Darstellungsschicht	Presentation Layer	Systemunabhängige Datendarstellung
7	Anwendungsschicht	Application Layer	Kommunikationsabläufe einer Anwendung

Protocol) statt. Der Trend dieses System auch auf die Produktionswelt auszudehnen ist seit langem ein Diskussionsthema der Automatisierungsbranche. Ein Ansatz mit diesem Ziel ist PROFINET (Process Field Ethernet) (PN), welches das derzeitig eingesetzte Feldbussystem PROFIBUS in einem einzigen Bussystem zu vereinen versucht. Basierend auf Ethernet, ist das weltweit genormte PN-Bussystem die standardisierte offene[1] Lösung für Industrial Ethernet.

2.2.1 Ethernet

Ethernet ist derzeit das meist eingesetzte Bussystem in lokalen Netzwerken. Es bedient die Schichten 1 und 2 des OSI-Schichtenmodells. Somit werden Übertragungsmedien (Kabeltypen, -arten), Anschlüsse (Steckverbindungen) und die damit zusammenhängenden elektrischen Eigenschaften sowie das Buszugriffsverfahren bei der Datenübertragung definiert. Darüber hinaus ist mit dem TCP/IP-Protokoll eine sinnvolle Anwendung durch das Bedienen höherer Protokollschichten möglich. Dieses befindet sich mit dem Internet-Protokoll (IP) auf der Schicht 3 oberhalb von Ethernet. Dadurch ermöglicht es ein *Routing*, also die Suche nach einem optimalen Verbindungsweg zwischen Kommunikationspartnern. Mithilfe der im Internet-Protokoll definierten IP-Adresse ist eine weltweite Kommunikation beim Zugriff auf

[1] bedeutet in diesem Zusammenhang, dass eine herstellerunabhängige Gerätekommunikation gewährleistet wird

das Internet möglich. Die Schicht 4, die Transport-Schicht, wird vom sog. Transmission Control Protocol (TCP) bedient. Dieses ist wichtig, da eine virtuelle *Punkt-zu-Punkt-Verbindung* zwischen zwei Kommunikationspartnern benötigt wird. Es dürfen einzelne Datenpakete einer Datensequenz weder verloren gehen noch manipuliert werden [SW12, Seite 270 ff.]. In der Feldebene ist der Einsatz von Ethernet-TCP/IP mitunter der fehlenden Echtzeitfähigkeit problematisch. Heutzutage ist aber trotz Defiziten der Datentelegramme die Übertragungsrate mit bis zu 10 GBit/s im Vergleich zu konventionellen Feldbussen deutlich schneller. Aufgrund solcher Aspekte und der kontinuierlichen Weiterentwicklung Ethernet-basierter Bussysteme gewinnt deren Anwendung in der Automatisierungstechnik immer mehr an Bedeutung.

2.2.2 PROFINET

PROFINET ist modular aufgebaut und kann somit vom Anwender je nach erfordertem Kommunikationsprofil in seinen Funktionalitäten individuell angepasst werden. Letztendlich besteht PROFINET aus 2 grundlegenden Konzepten. Zum Einen das sog. PROFINET IO, welches eine I/O-Datensicht auf die dezentrale Peripherie und die Echtzeitgeschwindigkeit der RT- (Real Time) und IRT-Kommunikation (Isochrone Real Time) beschreibt [SW12, Seite 283 ff.]. Zum Anderen das PROFINET CBA, das sich gut für Maschinen-Maschinen-Kommunikation eignet. Somit versucht PROFINET durch deren Kommunikationskonzepte optimale Performance je nach gebrauchter Anwendung zu gewährleisten. Diese Arbeit bezieht sich lediglich auf die Kommunikation mittels PROFINET IO, da diese Gegenstand der Automatisierungslösung ist.

PROFINET IO

> „PROFINET IO ist als Konzept eine Nachbildung von PROFIBUS DP auf Industrial Ethernet Basis und Switching-Technologie.“ [WZ14, Seite 580]

Grundsätzlich kann der Anschluss von PROFINET IO-Feldgeräten entweder sternförmig [SW12, Seite 8] über Mehrport-Switches oder linienförmig [SW12, Seite 3] mit einem direkt im Feldgerät verbauten Switch erfolgen. Somit steht PROFINET IO für das Steuerungskonzept mittels dezentraler Peripherien. I/O-Devices sind, wie beim kommunizieren basierend auf Ethernet üblich, mit MAC- und IP-Adresse, Subnetzmaske und Gerätename

eindeutig identifizierbar. Das verwendete Kommunikationsmodell des Datenverkehrs auf dem Bus ist prinzipiell dem *Master-Slave-Verfahren* gleich und wird an dieser Stelle *Provider-Consumer-Verfahren* genannt [WZ14, Seite 575 ff.].

Ein I/O-System besteht aus I/O-Controller und I/O-Devices (dezentrale Peripherien) mit oder ohne eigenen Prozessor (CPU). Der I/O-Supervisor ist ein PC, der entweder als Server oder zur Visualisierung genutzt wird, aber auch als Programmierungs- Diagnose und Wartungsgerät fungieren kann. Ein Human-Machine Interface (HMI) Panel übernimmt zusätzlich oder alleinstehend Visualisierungsaufgaben (vgl. Abbildung 2.1).

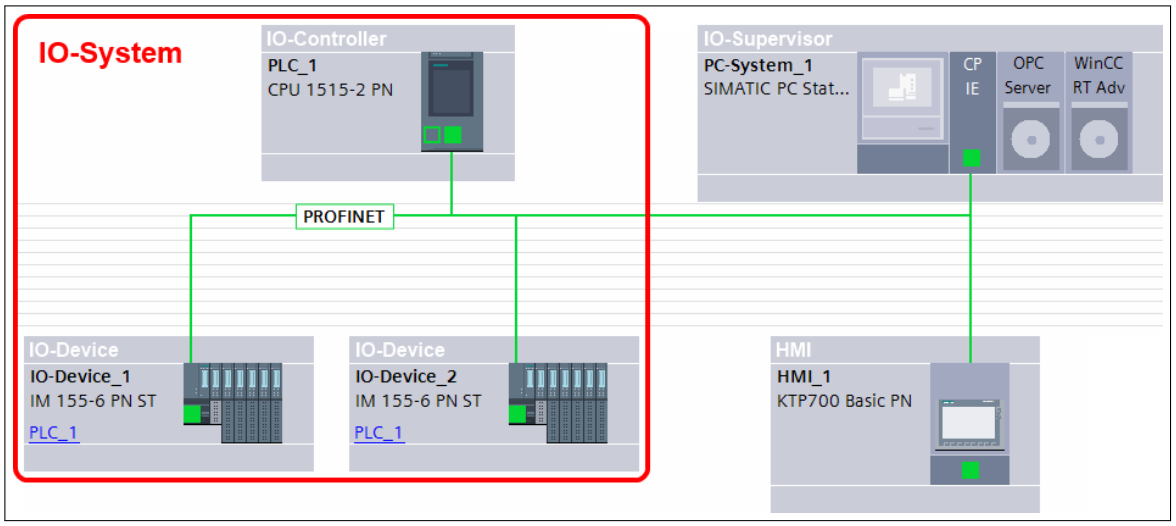


Abbildung 2.1: Aufbau eines PROFINET IO-Systems

2.2.3 Modbus

Heutzutage ist der Modbus in der Modbus-TCP-Variante ein genormtes und standardmäßig eingesetztes Bussystem in der Kommunikation der Automatisierungstechnik. Weitere Ausführungen wären Modbus-RTU (Remote Terminal Unit), Modbus-ASCII und Modbus-PLUS. Erstere kommuniziert mit der SPS direkt in binärer Form und beschreibt die Ursprungsübertragungsart von Modbus. Modbus-ASCII überträgt seine Daten mittels ASCII-Zeichen und ist somit für den Menschen direkt lesbar. Dies erfolgt aber auf Kosten der Datenrate, welche somit geringer als bei Modbus-RTU ausfällt. Beide Varianten kommunizieren über serielle Schnittstellen mit dem Buszugriffsverfahren Master-Slave. Modbus-PLUS hingegen bedient sich dem Token-Passing-Prinzip, wobei auch hier die Kommunikation seriell erfolgt.

Modbus-TCP

Modbus-TCP ist ein Industrial-Ethernet basiertes Bussystem, das nicht mit dem CSMA/CD-Buszugriffsverfahren (Carrier Sense Multiple Access / Collision Detection), sondern wie schon die vorhergehenden Modbus-Varianten mittels Master-Slave-Prinzip (Client-Server), agiert. Grundsätzlich ist dieser Bus zur schnellen Kommunikation von Automatisierungssystemen untereinander geeignet, kann aber auch als Backbone-Bus für andere prozessnahe Busse dienen. Das Telegramm ist bis auf den fehlenden Fehlererkennungscode innerhalb des Protokolls den anderen Modbus-Varianten strukturell ähnlich. Wie schon erwähnt kommuniziert das Modbus-TCP-Protokoll auf Basis der Ethernet typischen Busphysik und zusätzlichen TCP/IP-Bitmustern.

2.3 Speicherprogrammierbare Steuerung

2.3.1 Hardware

Grundlegend ähnelt die Struktur einer SPS, deren letztendliche Funktion vom Anwender individuell über ein Programm bestimmt werden kann, der eines Rechners. Sie ist modular aufgebaut und ist bereits mit wenigen Komponenten lauffähig. Der Hardwareaufbau einer solchen Steuerung besteht aus einer Stromversorgung, optional in Form einer PS-Baugruppe (Power Supply), sowie einer Zentraleinheit, die das Herzstück eines jeden modernen technischen Gerätes darstellt, der CPU (Central Processor Unit). Alle Module werden auf einem Baugruppenträger, der aus Profilschiene und Rückwandbus besteht, montiert und miteinander verbunden. Auf diesem können nach der CPU Signal- (SM), Technologie- (TM) oder Kommunikationsbaugruppen (CM) verbaut werden (vgl. Abbildung 2.2). Die SM-Baugruppen sind die Schnittstelle zu dem zu steuernden Prozess. Diese können als Ein- oder Ausgabebaugruppen fungieren und sind für Digital- oder Analogsignale mit verschiedenen Strömen oder Spannungen geeignet. TM-Baugruppen übernehmen Signalverarbeitungen unabhängig von der CPU. Hierbei findet eine Signalaufbereitung, -verarbeitung und entweder eine -bereitstellung für die CPU oder eine direkte -rückgabe an den Prozess statt. Die CM-Baugruppe ermöglicht einen Datenverkehr mit unterschiedlichen Kommunikationspartnern über Schnittstellen, welche die CPU bezüglich des Protokolls oder Funktionalität nicht zur Verfügung stellt [Ber14].
Heutzutage ist es üblich ein SPS-Steuerungssystem über einen Feldbus an eine dezentrale

Peripherie anzukoppeln. Diese besteht aus einem Interfacemodul (IM) und weiteren I/O-Modulen. Außer dieser Systeme gibt es PC-basierte Steuerungen, die durch die Möglichkeit eine komplexe Prozessdatenauswertung einfach zu hinterlegen und abzuspeichern, immer mehr an Bedeutung gewinnen. So kann die Visualisierung und Datenbankerstellung erheblich vereinfacht werden. Die sogenannte Soft-SPS (PC mit SPS-Software) bringt allerdings eine geringe Ausfallsicherheit, aufgrund von eventuellen Instabilitäten heutiger Betriebssysteme, mit sich. Dafür fällt ein solches System wesentlich billiger aus als ein konventioneller Steuerungsaufbau [WZ14].

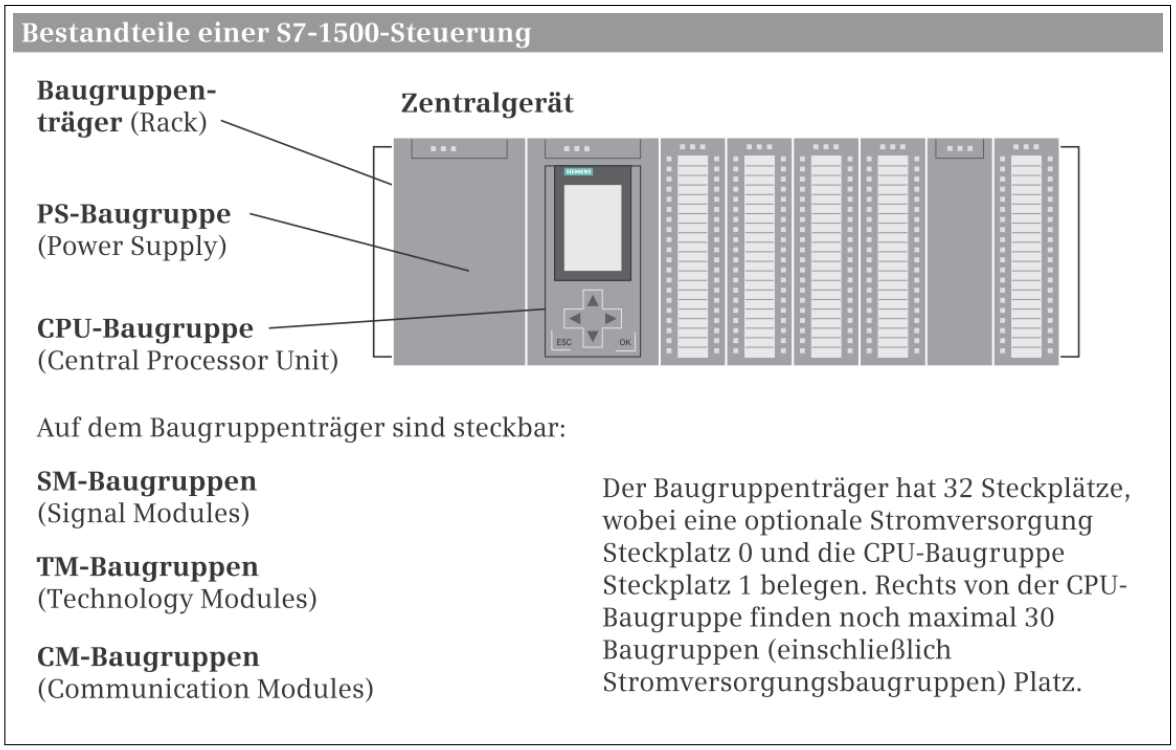


Abbildung 2.2: Bestandteile einer S7-1500 Steuerung (Quelle: [Ber14])

2.3.2 Die Norm IEC 61131-3

Das Programmieren ist ein wesentlicher Bestandteil der Automatisierungstechnik und unterscheidet sich in Aufbau und Ablauf unter den Automatisierungsfirmen. Um Missverständnisse zu vermeiden, ist ein standardisiertes und herstellerunabhängiges Programmierkonzept entstanden, welches das Ziel der Vereinfachung und Geringhaltung der Kosten des Programmieraufwandes verfolgt. Die Europanorm Deutsches Institut für Normung (DIN) Europäische Norm (EN) 61131-3 auf Basis der Weltnorm IEC 61131-3, hat den Programmierstandard

durch das Definieren grundlegender Konzepte entscheidend vereinheitlicht. Dieses Konzept besteht laut [WZ14] aus den folgenden wesentlichen Teilen. Zum Einen beschreibt die *Datentyp- und Variablenkonzeption* alle Datentypen, Konstanten und Variablen verbindlich. Die so eingeführten Datentypen sind Grundlage der Variablendeklaration für symbolische und absolute Adressierung. Zum Anderen ist mit dem *Programmorganisationskonzept* ein System, welches sog. Programmorganisationseinheiten ordnet und entsprechend untergliedert, eingeführt worden. Dieses besteht aus einem Haupt- und zwei Unterprogrammtypen. Im Wesentlichen werden hier Gedächtnisfunktionen beschrieben, die ihre Werte über einen Zyklus hinaus speichern können (Datenbausteine). Im nächsten Normabschnitt ist das *Fachsprachenkonzept* definiert. Hier werden die Programmiersprachen für die Softwareentwicklung festgelegt. Mit dem Funktionsplan (FUP) und dem Kontaktplan (KOP) existieren zwei grafisch orientierte und mit der Anweisungsliste (AWL) und dem Strukturierten Text (ST) zwei textuelle Programmiersprachen. Die Ablaufsprache (AS) beinhaltet sowohl grafische als auch textuelle Elemente. Als letztes beschreibt der Autor das *Taskkonzept*, welches die Ausführung einer Automatisierungsaufgabe beschreibt. Diese erfolgt über eine priorisierte, zeitzyklische und ereignisorientierte Programmsteuerung. Somit legt die Norm DIN EN 61131-3 alle für die herstellerunabhängige SPS-Programmierung wichtigen Konzepte in einer herstellerspezifischen Programmierumgebung, wie z.B. das Controller Development System (CoDeSys) oder STEP 7 bzw. das Totally Integrated Automation (TIA) Portal, fest.

2.3.3 Programmiersprachen

In der IEC 61131-3 sind, wie bereits erwähnt, alle Programmiersprachen, mit denen eine Speicherprogrammierbare Steuerung (SPS) programmiert werden kann, definiert. Die folgenden Beispiele sollen die Programmierweise mithilfe der einzelnen Sprachen verdeutlichen. Diese basieren auf der Darstellungsweise von STEP 7 und sind mit dem TIA Portal erstellt worden. Aufgrund dessen ist die Ansicht der folgenden Netzwerke Siemens spezifisch und unterscheidet sich von der normierten Darstellungsart.

Funktionsplan (FUP)

Der Funktionsplan ist eine der zwei grafischen Programmiersprachen. Basierend auf Logik-Gattern nutzt FUP gezielt die Vorteile der Übersichtlichkeit und Nachvollziehbarkeit kleinerer

Netzwerke und dessen leichtere Handhabung in der Programmierung und Fehleranalyse. Der Nachteil dieser Darstellungsweise eines SPS-Programms ist die immer weiter abnehmende Übersicht bei großen Netzwerken. Dies ist bei grafischen Sprachen ein allgemeines Problem. Das folgende Beispiel (vgl. Abbildung 2.3) verdeutlicht die Darstellungsweise eines FUP-Netzwerkes.

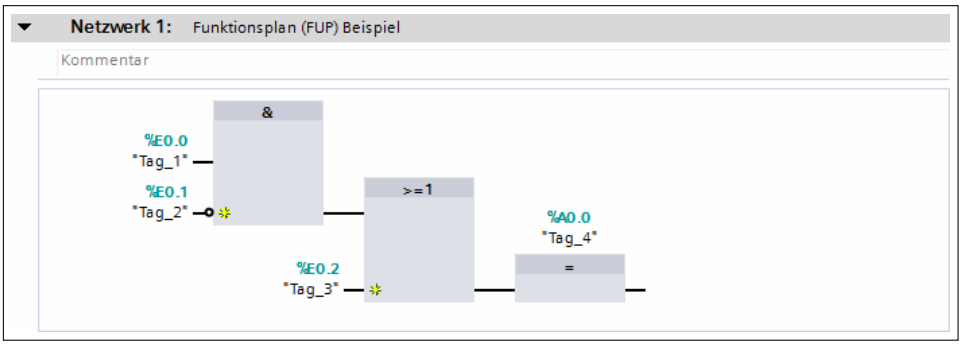


Abbildung 2.3: Beispielprogramm im Funktionsplan (FUP)

Der Ausgang **A0.0** („Tag_4“) wird **TRUE** wenn entweder **E0.0** („Tag_1“) **TRUE** und **E0.1** („Tag_2“) **FALSE** oder **E0.2** („Tag_3“) **TRUE** ist. Weiterhin sind hardware- und softwareseitige Adressen erkennbar. Diese sind in Symboltabellen hinterlegt und jederzeit anpassbar. Alle folgenden Beispielprogramme erfüllen dieselbe Funktion.

Kontaktplan (KOP)

Der zweite Vertreter grafisch orientierter Fachsprachen ist der Kontaktplan. Dieser bedient sich einer Darstellung ähnlich eines Stromlaufplans (vgl. Abbildung 2.4). Somit ist er für Personen mit fachspezifischer Expertise in diesem Gebiet ohne spezielle Programmierkenntnisse lesbar und vor allem nachvollziehbar. Als Vor- und Nachteile sind die gleichen wie bei Funktionsplan zu benennen.

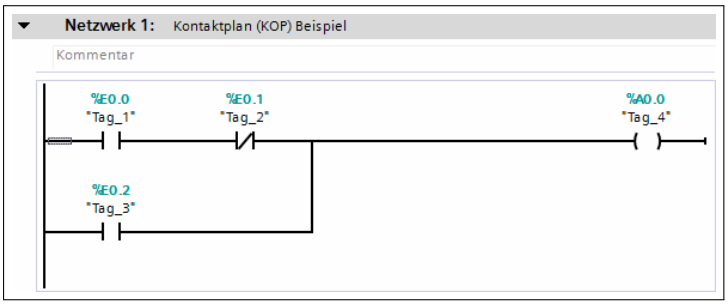


Abbildung 2.4: Beispielprogramm im Kontaktplan (KOP)

Anweisungsliste (AWL)

Die Anweisungsliste ist der erste Vertreter der insgesamt zwei textuellen Fachsprachen. Diese ist an die Assemblersprache angelehnt und ist heutzutage eher selten anzutreffen. In älteren Steuerungen ist sie noch im Einsatz, da der Programmcode früher oftmals klein gehalten werden musste um Speicher zu sparen. Auf aktuellen SPS-Systemen ist sie größtenteils vom übergeordneten Strukturierten Text abgelöst worden und daher kaum noch in Verwendung. Anhand des Beispiels (vgl. Abbildung 2.5) wird der AWL-Syntax verdeutlicht.

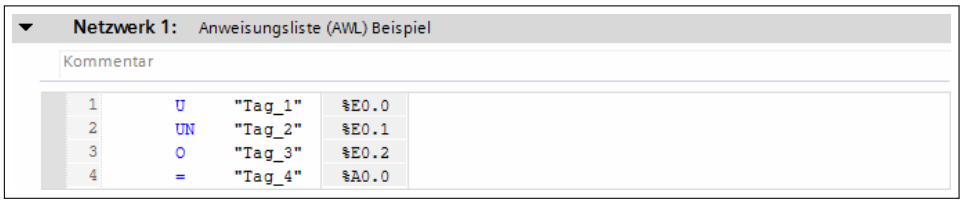


Abbildung 2.5: Beispielprogramm in Anweisungsliste (AWL)

Strukturierter Text (ST)

Der Strukturierte Text (ST), der bei Siemens S7 der Structured Control Language (SCL) entspricht, ist die aktuelle Hochsprache in der SPS Programmierung. Angelehnt an den Pascal-Syntax ist diese Sprache im direkten Vergleich mit AWL besser in ihrer Strukturierung und bietet die Möglichkeit für Softwareentwickler ein SPS-Programm ohne spezielle Vorkenntnisse in deren Programmierung nachzuvollziehen (vgl. Abbildung 2.6).

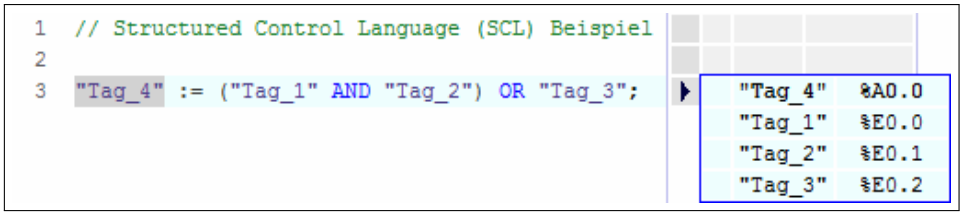


Abbildung 2.6: Beispielprogramm in Strukturiertem Text (SCL)

Ablaufsprache (AS)

Die Ablaufsprache bietet sowohl grafische als auch textuelle Elemente der Programmierung. Bei Siemens S7 wird diese Sprache als S7-GRAPH bezeichnet. Im Prinzip funktioniert diese in Form eines PETRI-Netzes. Grundlegend ist es eine Ablaufkette, die mittels Weiterschaltbedingungen (Transitionen) Steuerungsschritte schaltet. Die Transitionen werden mithilfe

von grafischen Fachsprachen definiert und Aktionen in den Schritten ausgelöst (vgl. Abbildung 2.7).

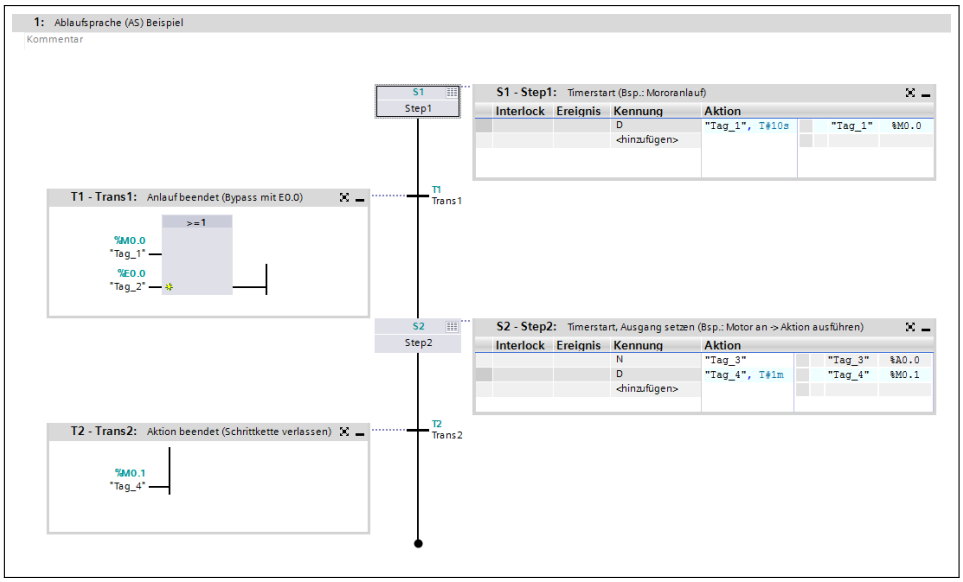


Abbildung 2.7: Beispielprogramm in Ablaufsprache (AS)

Schrittketten werden zu Beginn initialisiert. Da für S7-GRAPH ein Funktionsbaustein (FB) erstellt wird, ist dies in der Aufrufumgebung an dem FB-Eingang „INIT_SQ“ möglich. Im Beispiel ist eine Ansteuerung eines Motors realisiert. Dieser benötigt eine gewisse Zeit zum Warmlauf und kann dann seine Aktion ausführen. Im **Step1** wird der Timer zum Anlauf gestartet. Dort ist es möglich die Transition (T1) mithilfe des Bypass Signals **E0.0** („Tag_2“) zu schalten oder auf den Timerablauf **M0.0** („Tag_1“) zu warten. Nachdem T1 geschaltet hat befindet sich der Motor im Run-Zustand (**Step2**). Hierbei ist eine Bearbeitungszeit vorgesehen und der Motor wird mit dem Ausgang **A0.0** („Tag_3“) angesteuert. Dieser wird nach Ablaufen der Zeit mit dem Merker **M0.1** („Tag_4“) abgeschaltet und die Schrittkette kann nach erneutem Initialisieren von vorn beginnen.

2.3.4 Totally Integrated Automation Portal

Das TIA Portal oder auch STEP 7 Professional gilt als das zentrale Automatisierungswerkzeug für SIMATIC. Mit ihr werden aktuelle oder ältere Steuerungen konfiguriert, projektiert und programmiert. Ziel dieses Systems ist das Vereinen aller Automation Engineering Systeme in einer Entwicklungsumgebung. In der derzeitigen Version 13 unterstützt das Portal die aktuellen Betriebssysteme unter Microsoft Windows und benötigt für den Betrieb eine Autorisierung

in Form einer kostenpflichtigen Lizenz. Die Projektierung eines Siemens Controllers erfolgt in folgenden Ansichten. Die aufgabenorientierte *Portalansicht* (vgl. Abbildung 2.8) soll das Erstellen, Editieren, Verwalten und Migrieren neuer oder vorhandener Projekte vereinfachen sowie die dafür nötige Übersichtlichkeit mitbringen. Im Vordergrund stehen die wichtigsten Werkzeuge und Funktionen des TIA Portals mit einfacher und eindeutiger Navigation.

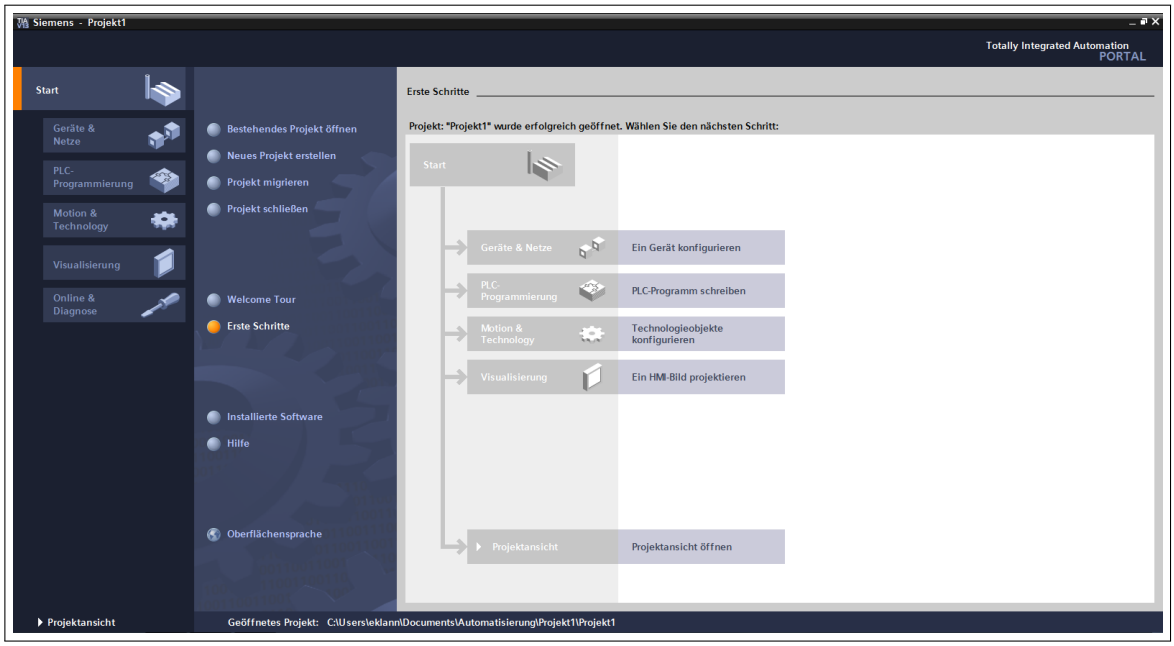


Abbildung 2.8: TIA Portal „Portalansicht“

Dem gegenüber steht die eher klassische, objektorientierte *Projektansicht* (vgl. Abbildung 2.9). Diese legt den Fokus auf Listenübersichten und dynamisierte Fensteranzeigen. Der Anwender hat also die Möglichkeit seine Programmoberfläche unkompliziert und individuell anzupassen [Ber14]. Durch das Zusammenlegen von STEP 7 und Windows Control Center (WinCC) ist eine umfangreiche und komfortable Programmieroberfläche entstanden, die viele Automatisierungsaufgaben vereint.

2.4 Visualisierung

Eine Visualisierung im Sinne der Automatisierungstechnik wird im Allgemeinen als Human-Machine Interface (HMI) oder zu deutsch Mensch-Maschine-Schnittstelle (MMS) bezeichnet. Ein HMI besteht in der Regel aus der Hardware (Ein- und Ausgabegeräte) und Software, mit der die grafische Oberfläche und der Informationsgehalt einer Visualisierung bestimmt wird.

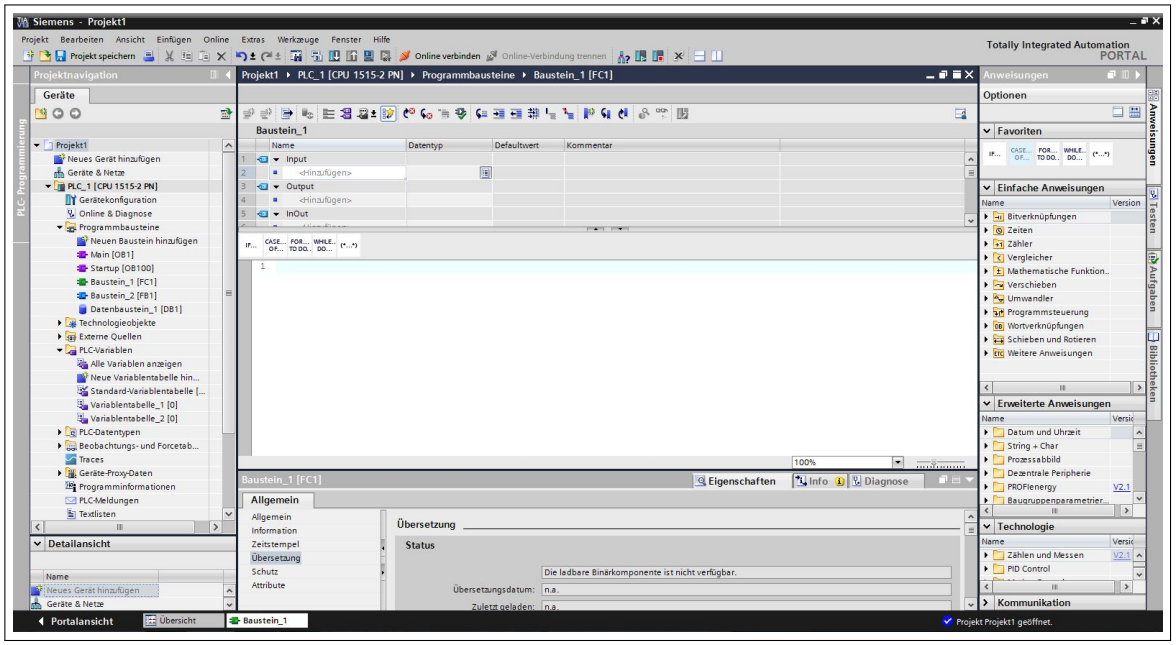


Abbildung 2.9: TIA Portal „Projektansicht“

Im Bereich der Hardware gibt es verschiedene Anbieter, die in diesem Gebiet unterschiedliche HMI-Lösungen, je nach Anwendungsspezifikationen, anbieten. Dabei sind Einsatzgebiet und Umgebungseigenschaften ausschlaggebend für die Verwendung der HMI-Hardware. Während beispielsweise in der prozessnahen Industrieumgebung eher auf ein robustes HMI-Panel gesetzt wird, kommt in einer Leitwarte ein Computer System zum Einsatz. Auf Basis dessen bekommt der Anwender je nach gefordertem Leistungsbereich eine breite Masse an HMI-Systemvarianten geboten. Im Leistungsspektrum der Firma Siemens sind unter anderem SIMATIC HMI-Panels von kleinsten Basic- bis hin zu großen Comfort- und sogar mobilen Ausführungen enthalten. Außer den Panels sind SIMATIC PC-Systeme einsetzbar, die für HMI- oder Supervisory Control and Data Acquisition (SCADA) Anwendungen ausgelegt sind. SCADA dient dem Steuern und Überwachen eines technischen Prozesses und ist das dem HMI übergeordnete System. Hier sollen beispielsweise alle Anlagenteile global einsehbar, überwachbar und steuerbar sein. Die Visualisierungssoftware oder auch Runtime, dient der Anzeige wichtiger Anlagenübersichten mit dazugehörigen Prozesszuständen. Der Anwender kann Aktualwerte oder Parameter jederzeit auslesen und somit nachvollziehen in welchem Zustand sich der Prozess befindet. Außerdem können Meldungen, Alarme, Diagnosen, Parametrierungen, usw. mithilfe von Textmeldungen und Eingabemöglichkeiten einfach vom

HMI übernommen werden, ohne in den Programmablauf eingreifen zu müssen.

SIMATIC WinCC (TIA Portal)

Diese Variante der HMI Visualisierungs- und Projektierungssoftware ist eng mit dem aktuellen TIA Portal verbunden. Der Anwender kann so alle Automatisierungsaufgaben in einer Entwicklungsumgebung vorfinden. Die Anwendungen sind einerseits in maschinennaher Programmierung und zusätzlich im Prozessvisualisierungs- oder SCADA-Umfeld abgedeckt. Des Weiteren bietet Siemens die vier Projektierungswerkzeuge (vgl. Abbildung 2.10) zur Projektierung der aktuellen SIMATIC HMI-Bediengeräte an. Mithilfe von WinCC *Basic*, *Comfort*, *Advanced* und *Professional* können Panels, Industrie-PCs oder PC-basierte Systeme zur Visualisierung genutzt werden (vgl. Abbildung 2.10).

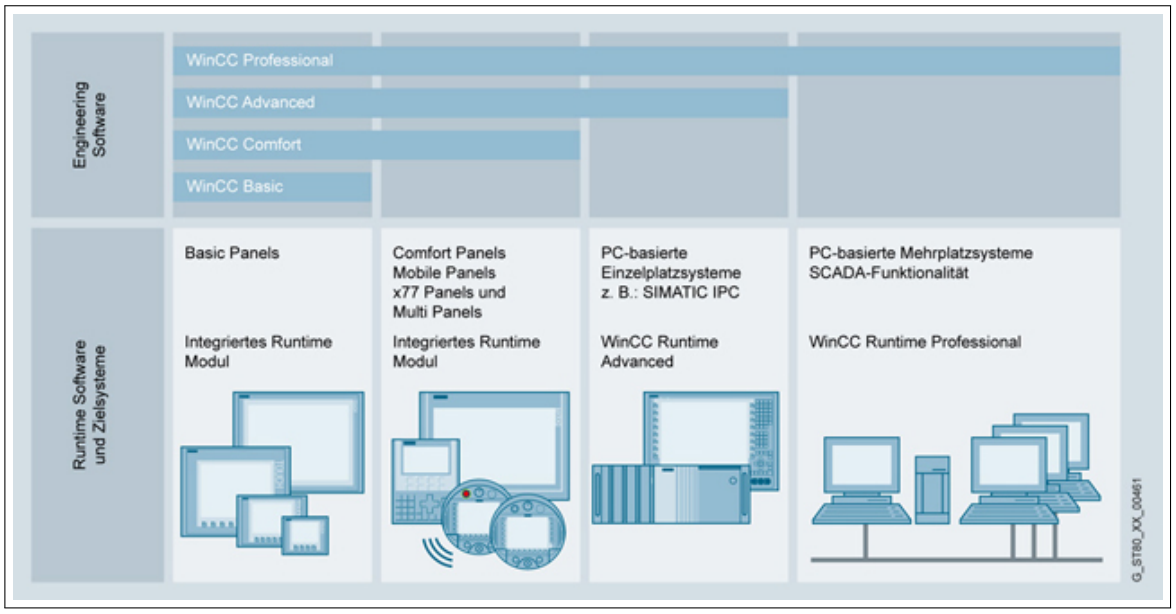


Abbildung 2.10: WinCC Lizenzen-Übersicht (Quelle: [Sie14g])

Abstufungen in den lizenzierten Varianten von WinCC sind bezüglich der Leistungsanforderung einer Visualisierung hinzunehmen. Bei Kleinanlagen mit übersichtlicher Anzahl an Prozessvariablen reicht WinCC *Basic* vollständig aus. Mit der Einschränkung, dass die *Basic* Ausführung lediglich auf SIMATIC *Basic*-Panels läuft. Die *Comfort*-Variante schließt die Projektierung von *Comfort*-Panels mit ein. Die Runtime ist bei den Panels in Form eines Moduls integriert. Im Gegensatz dazu können mit WinCC *Advanced* neben den konventionellen SIMATIC-Panels auch PC-basierte Einzelplatzsysteme, z.B. SIMATIC IPC oder Windows-

PCs, projektiert werden. Die Visualisierungsumgebung „*WinCC Runtime Advanced*“ wird dann als Programm auf dem PC installiert. Für SCADA Funktionalität wird das umfangreichste WinCC-Paket benötigt. Die „*WinCC Runtime Professional*“ bietet genau diese für den Mehrplatzbetrieb von PC-Systemen. Mit diesen Varianten deckt WinCC alle Aufgaben eines HMI-Systems der Automatisierungstechnik ab.

3 Vorbetrachtung

Die Modernisierung heutiger Gebäude kann vielseitig und individuell von statten gehen. Eine großes Feld deckt die Gebäudesystemtechnik ab. In dieser haben sich mittlerweile einige Systeme etabliert und es wird weiterhin viel Entwicklung betrieben, da die Zukunft von energiesparenden und komfortablen Gebäuden profitieren wird. Bei der Wahl eines solchen Hausautomatisierungssystems sind Vor- und Nachteile, eigene Wünsche, finanzielle Vorstellungen und der Automatisierungsumfang in Bezug auf Hardware- und Programmieraufwand besonders interessant. Im folgenden Abschnitt sind konventionelle Varianten der Gebäudesystemtechnik aufgeführt. Ziel dieser Vorbetrachtung ist das Ausschließen nicht in Frage kommender Systemausführungen und das Erläutern der Herangehensweise an die anschließende Automatisierungslösung, basierend auf einer Konzepterstellung.

3.1 Systemvarianten möglicher Automatisierungslösungen

3.1.1 Konnex (KNX)

Allgemein

In vielen der heutigen Gebäuden, die eine Gebäudesystemtechnik führen, ist KNX anzutreffen. Ein Grund dafür ist die beachtliche Anzahl an Normen, die dieses System als einzig offenen[1] Standard hervorbringt.

> „KNX ist ein industrielles Kommunikationssystem, welches in der Haus- und Gebäudesystemtechnik zur informationstechnischen Vernetzung von Geräten (Sensoren, Aktoren, Steuer- und Regelgeräte, Bedien- und Beobachtungsgeräte genutzt wird. Sein Einsatz ist auf die Elektroinstallation abgestimmt, wodurch Funktionen und automatisierte Abläufe in einem Gebäude sichergestellt werden“ [MHH10, S. 63].

KNX geht aus der European Installation Bus Association (EIBA), die den ursprünglichen europäischen Installationsbus EIB weiterentwickelt haben, hervor. Nach dem Zusammenschluss

[1] auch hier ist herstellerunabhängige Gerätekommunikation gewährleistet

mit dem BatiBUS Club International (BCI) und der European Home Systems Association (EHSA) gründete sich die heutige Konnex Association, welche den ehemaligen EIB unter dem Namen KNX vermarktet. Seitdem ist dieses Bussystem eines der meist eingesetzten Systeme in Europa [MHH10].

Kriterien

Das größte Problem von KNX ist die Finanzierbarkeit, sowie der Freiheitsgrad bei der Programmierung. Trotz der Ersparnisse bei der Verkabelung wird ein solches System bei großen Umfang der Automatisierungsaufgabe relativ teuer. Oftmals ein Grund sich gegen eine Buslösung zu entscheiden ist aber nicht nur finanzielle Aspekt, da genauso die Individualität der Einzelfunktionen durch das Vorgeben von Bibliotheksbausteinen zwangsweise eingeschränkt wird. Natürlich ist durch solche Bausteine ein unproblematischer Betrieb gewährleistet und bietet Einsteigern einen Vorteil gegenüber der freien Programmierung. Dies stellt aber für einige Anwender ein unschönes Hindernis dar. Bei diesem Projekt hat die gesamte Automatisierung aber nicht nur die Funktion des Komforts, sondern soll repräsentativ den Leistungsumfang der Firma deutlich machen. In Verbindung mit allen Einschränkungen erwies sich dies letztlich als ausschlaggebende Begründung gegen eine KNX-Lösung.

3.1.2 Local Operating Network (LON)

Allgemein

LonWorks oder kurz LON stellt im Gegensatz zu KNX deutlich höhere Anforderungen an die Leistungsfähigkeit der Prozessoren und den Programmieraufwand. LON ist überwiegend in Regelbausteinen der Gebäudeautomation und in dezentralen Komponenten der Gebäudesystemtechnik vorzufinden. Die Einsatzgebiete beschränken sich aber nicht nur darauf, denn es kann genauso gut eine Licht- oder Jalousieansteuerung übernehmen. Somit steht LON in direkter Konkurrenz zum KNX-Bussystem. Die beiden Systeme haben aber einen ähnlichen Systemansatz und ergänzen sich teilweise so gut, dass in manchen Fällen beide Protokolle mithilfe eines Busankopplers verwendet werden. Entwickelt wurde die LonWorks-Technologie von der Firma Echolon und ist auf Basis der EN 14 908 genormt [MHH10].

Kriterien

LON ist bei einer Gebäudeautomation im großen Stil grundsätzlich immer in Betracht zu ziehen. So hat diese Buslösung entscheidende Vorteile gegenüber anderen Systemen dieser Art, wie z.B. der erwähnten Leistungsfähigkeit oder der erhöhten Möglichkeiten in Dezentralisation und Kommunikation. Diese Aspekte können für manchen Anwender entscheidend in der Wahl gegen ein Welt-genormtes KNX-System oder eine Lösung basierend auf einer Steuerung sein. Trotz der genannten Vorteile wird es nicht zum Einsatz kommen, da die Handhabung eines LON-Systems nicht unproblematisch ist und einiges an Erfahrung mit dessen Programmierung voraussetzt. Des Weiteren entspricht eine solche Lösung nicht dem Leistungsspektrum der Firma und somit nicht der erwähnten Forderung der Repräsentativität.

3.1.3 Building Automation and Control Networks (BACnet)

Allgemein

BACnet ist mit der DIN EN ISO 16484-5, im Gegensatz zu LON, wie auch schon KNX, weltweit genormt. Dieses Kommunikationssystem ist üblicherweise in der Managementebene des Schichtenmodells zu finden. Grundsätzlich könnte es in allen Ebenen der Gebäudeautomation eingesetzt werden, wird aber aufgrund besonderer Stärken häufig übergeordnet in Verbindung mit einem untergeordneten KNX- oder LON-System verwendet. BACnet ist vielseitig und bietet, trotz unterschiedlicher Hard- und Software, ein gutes Kommunikationsprotokoll, das Markttransparenz und Herstellerunabhängigkeit in vielen Fällen ermöglicht [MHH10].

Kriterien

Für den Einsatz im Neubau ist BACnet nicht nötig. Dieses Protokoll ist wie erwähnt für einen übergeordneten Einsatz optimiert und würde für etwaige Anwendungen überdimensioniert sein. Das Gebäude ist nicht groß genug um eine Dezentralisierung zu realisieren, in der ein solches System in seinen Möglichkeiten sinnvoll ausgereizt werden würde.

3.1.4 Steuerung

Die HothoData GmbH hat bei ihrem Vorhaben das neue Gebäude zu automatisieren bereits konkrete Vorstellungen. Dazu gehört das System mit welcher die Hausautomatisierung reali-

siert werden soll. Wie zuvor schon erwähnt wird das System eine repräsentative Wirkung der Firma bieten. Außerdem soll von den dortigen Ingenieuren ohne großen Aufwand einfach und schnell Änderungen vorgenommen werden können. Da Steuerungen deren tägliches Handwerk sind, ist diese Forderung bei der Wahl des Systems natürlich ausschlaggebend. Des Weiteren besitzt die Firma Elektroingenieure, die Schaltschrank und Elektropläne entwickeln und ein solches Vorhaben unterstützen können. Die Werkstatt kann dann das Verkabeln und Verdrahten unkompliziert übernehmen. Somit kann alles innerhalb der Firma entwickelt und umgesetzt werden. Diese Kriterien führten zu einer nicht üblichen aber durchaus interessanten Gebäudeautomatisierung mittels SPS. Steuerungen in Gebäuden können durchaus eine lohnende Alternative für SPS-Entwickler sein. Das Haus braucht kein spezielles Bussystem und dieses muss bei Bestandsbauten nicht nachgerüstet werden. Es braucht lediglich die Hard- und Software des Steuerungssystems und weitere Schütze und Relais zum Schalten der Aktorik. Sicherungen und Hauptschalter können natürlich auch im Schaltschrank untergebracht werden. Je nach Anforderung und Vorstellungen können hier Steuerungssysteme von verschiedenen Herstellern zum Einsatz kommen. Das Unternehmen beschäftigt sich überwiegend mit Steuerungen der Firma Siemens. Somit fiel die Wahl auf eine der aktuellen S7-1500 Automatisierungssysteme. Der gesamte Aufbau wird im Folgenden aufgeführt.

3.2 Konzept

Das Konzept beinhaltet verschiedene Möglichkeiten, welche die Realisierung einer solchen Gebäudeautomatisierung mit sich bringen könnten. Nicht alle der Funktionen werden an der realen Steuerung umgesetzt, sollten aber trotzdem beachtet werden, um zukünftige Änderungen zeitnah und ohne großen Aufwand implementieren zu können.

3.2.1 Hardware

Die Hardwarekonfiguration wird mit dem TIA Selection Tool *(www.siemens.com/tia-selection-tool)* der Firma Siemens aufgebaut. Anhand dessen können die erforderlichen Hardwarebauteile nach erfolgreicher Kalkulation und Absprache mit dem Projektleiter bestellt werden. Eine Vorstellung des realen Steuerungsaufbaus (vgl. Anlage A.3, Seite A-3) und des Schaltschranks (vgl. Anlage A.5, Seite A-5) sind der Anlage zu entnehmen.

CPU 1515-2 PN + Memory Card 4 MB

Abbildung 3.1: CPU1515-2 PN

Diese CPU besitzt 2 PN-Schnittstellen. Die erste (X1) besteht aus 2 Ports. Dieser unterstützt PROFINET I/O und IRT. Nur an dieser Schnittstelle ist eine Echtzeit-Einstellung projektierbar. Sie wird zur Kommunikation zwischen dem IM und der CPU genutzt. Die zweite PN-Schnittstelle (X2) besitzt lediglich einen Port und wird zur Kommunikation mit einem übergeordneten Netz oder einer anderen Maschine verwendet. Die Schnittstelle ist nicht für einen I/O-Controller gedacht und unterstützt somit nur Basisfunktionalitäten des PN Bussystems. An diesem Port wird eine Kommunikation mit dem Firmenethernet erfolgen. Der integrierte Webserver kann die Bedienung verschiedener Funktionen über jeden PC im Firmennetz gewährleisten. Außerdem sind so verschiedene Diagnosen durchführbar, z.B kann der aktuelle Betriebszustand der CPU abgerufen, Informationen über Baugruppenzustände, aktuelle Meldungen oder Status der Topologie/PN-Geräte ausgelesen bzw. abgerufen werden. Des Weiteren unterstützt die CPU diverse Funktionalitäten wie Firmware Updates, PROFlenergy, Traces, Motion Control, integrierte Regler usw., welche aber für den geplanten Einsatz lediglich eine untergeordnete Rolle spielen. Auffallend ist das neue Farbdisplay, welches nicht nur der Optik zugutekommt, sondern auch den Diagnosemöglichkeiten direkt an der Steuerung. Somit entfallen beispielsweise Leuchtdioden für Sammelfehler oder interne Fehler und das Display übernimmt dessen Anzeigen. Schützend für die Anschlusstechnik der CPU ist eine Frontschutzklappe angebracht, die Netzwerkstecker, Betriebsartenschalter und Spannungsversorgungsanschlüsse abdeckt. Diese Klappe ist mittels einer Verriegelungslasche noch zusätzlich versiegelbar. An der Rückseite sind eine Rückwandbus Steckverbindung, Schirmkontaktflächen und Befestigungsschrauben für die Profilschienenbefestigung eingearbeitet. Außerdem besitzt die CPU einen Schacht für die SIMATIC Memory Card (MC) (6ES7954-8LC01-0AA0). Diese ist für den Ladespeicher der CPU wichtig und wird unbedingt benötigt.

Die MC gibt es in verschiedenen Ausführungen. Unter anderem mit 4 MB Speicherplatz, welche für das Projekt eingesetzt wird [Sie14f].

ET 200SP, Interfacemodul IM 155-6 PN ST inkl. Servermodul & Busadapter

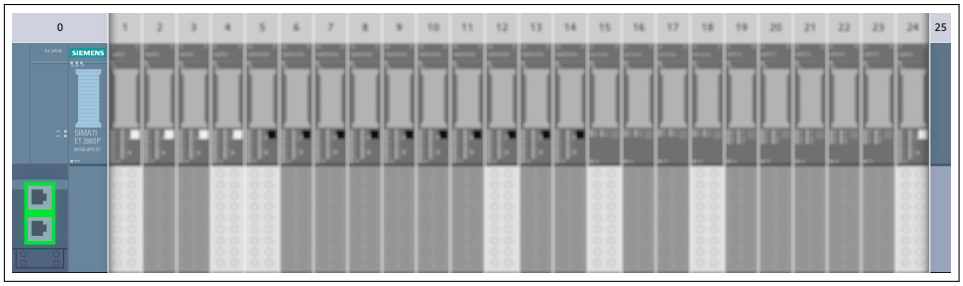


Abbildung 3.2: ET 200SP Interfacemodul (0) und Servermodul (25)

Das Interfacemodul (vgl. Anlage A.4, Seite A-4) verbindet das DP-System ET 200SP über PROFINET I/O mit der CPU (vgl. Abbildung 3.2). Dieses wird durch die Versorgungsspannung 1L+ DC 24V gespeist und über den mitgelieferten Anschlussstecker verdrahtet. An das IM können bis zu 32 Peripheriemodule mit maximal 256 Byte Input/Output (I/O) Daten angeschlossen werden. Dies kann sich bis auf einen Meter Rückwandbus ohne Interfacemodul ausdehnen. Der PROFINET IO-Anschluss erfolgt über den mitgelieferten BA 2xRJ45. Das Servermodul ist auch im Lieferumfang enthalten und hat die Aufgabe den Rückwandbus des dezentralen Peripheriesystems ET 200SP abzuschließen und gegebenenfalls eine Sammeldiagnose über eine fehlende Versorgungsspannung pro Potenzialgruppe auszugeben. Im Projekt wird das IM die Kommunikation zwischen CPU und den I/O-Peripheriemodulen übernehmen [Sie13].

ET 200SP BaseUnit

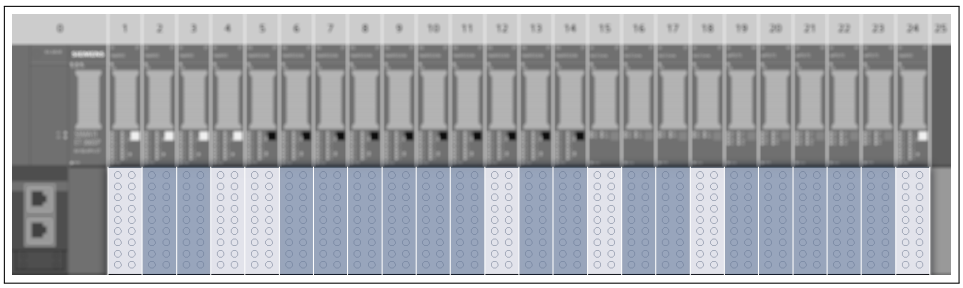


Abbildung 3.3: ET 200SP BaseUnits

Die BaseUnits existieren in verschiedenen Ausführungen. Für das Projekt werden nur BaseUnits vom Typ „A0/A1“ gebraucht (vgl. Abbildung 3.3). Es gibt von diesem Typ speisende und brückende Elemente. Für das Peripheriemodul AI 4xRTD/TC 2-,3-,4-wire HF wird des Weiteren eine zusätzliche BU (BU15-P16+A0+12D/T) benötigt, die die Klemmentemperatur erfassen kann, um die interne Temperatur zu kompensieren. Somit wird bei angeschlossenem Thermowiderstand der Messwert nicht verfälscht. Mit den speisenden BaseUnits (BU15-P16+A0+2D) werden neue Potenzialgruppen eröffnet. Die Anzahl der einsetzbaren Peripheriemodule pro Potenzialgruppe ist abhängig vom Strombedarf der betriebenen Module und der daran angeschlossenen Lasten. In diesem Projekt werden die Potenzialgruppen zur Trennung der Spannungsversorgung für einzelne Peripheriemodulgruppen verwendet. Zum Beispiel wird die Versorgungsspannung abends abgeschaltet, da aber Jalousien auch in der Nacht funktionieren müssen (Wind), werden diese mithilfe einer extra Spannungsversorgung gespeist. Die brückende BU (BU-P16+A0+2B) erweitert eine bestehende Potenzialgruppe. Ihre Klemmen haben eine Stromtragfähigkeit von max. 2A. Alle eingesetzten BaseUnits besitzen 16 Klemmen, welche immer vom angeschlossenen Peripheriemodul belegt werden [Sie14b].

ET 200SP, digitales Eingangsmodul, DI 16x24VDC ST

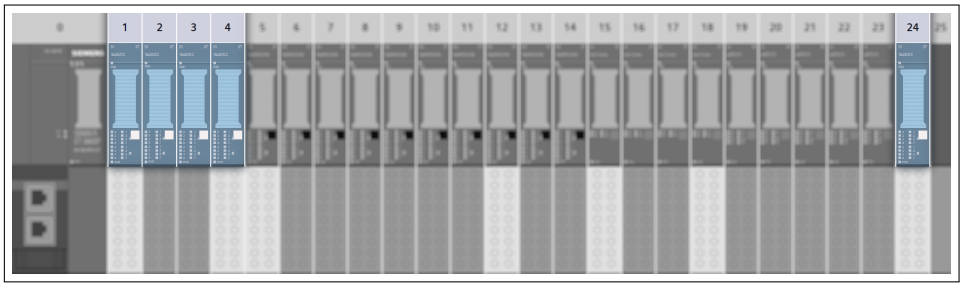


Abbildung 3.4: ET 200SP Digitaleingabemodul (1-4) (24 nachgerüstet)

Dieses Digitaleingabemodul (vgl. Abbildung 3.4) bietet 16 Eingänge und eignet sich für den Anschluss von Schaltern und 2 Drahtsensoren nach International Electrotechnical Commission (IEC) 61131-3. Des Weiteren ist kanalweise eine parametrierbare Eingangsverzögerung und modulweise eine parametrierbare Diagnose projektierbar. Dieses Modul ist nur in Kombination mit einer passenden BU einsetzbar [Sie12c].

ET 200SP, digitales Ausgangsmodul, DQ 16x24VDC/0,5A ST

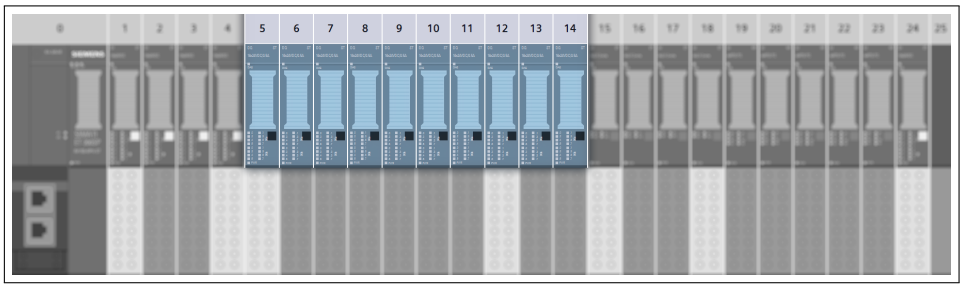


Abbildung 3.5: ET 200SP Digitalausgabemodul (5-14)

Das Digitalausgabemodul (vgl. Abbildung 3.5) besitzt 16 Ausgänge und eignet sich für Magnetventile, Gleichstromschütze und Meldeleuchten. Im Fall dieses Projektes werden vorrangig Relais angesteuert. Der Ausgangsstrom entspricht 0,5 A je Kanal und ist P-schaltend. Auch hier ist eine Diagnose parametrierbar. Wie beim Digitalausgabemodul sind gewisse Funktionen implementiert. Zum Beispiel ist eine Umparametrierung im RUN Zustand möglich. Des Weiteren ist auch dieses Modul nur in Kombination mit der passenden BaseUnit einsetzbar und setzt eine korrekte Lastgruppenbildung voraus [Sie12b].

ET 200SP, analoges Eingangsmodul, AI 4xU/I 2-wire ST

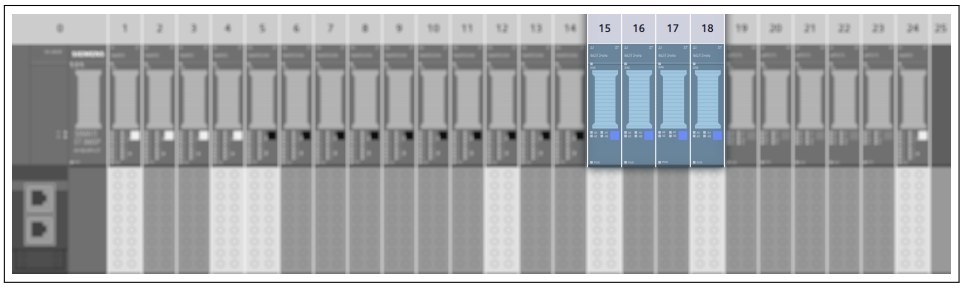


Abbildung 3.6: ET 200SP U/I-Analogeingabemodul(15-18)

Das Modul (vgl. Abbildung 3.6) besitzt 4 Eingänge für Strom- und Spannungsmessung und 2 Drahtmessumformer. Die Eingangsbereiche für die Spannungsmessung sind: ± 10 V oder ± 5 V, mit einer Auflösung von 16 Bit inklusive Vorzeichen, 1 bis 5 V oder 0 bis 10 V mit einer Auflösung von 15 Bit. Die Strommessung besitzt folgende Eingangsbereiche: 4 bis 20 mA oder 0 bis 20 mA mit einer 15 Bit Auflösung. Bei der Spannungsmessung sind die Eingänge potenzialgetrennt also, passiv zur Versorgungsspannung. Auch hier ist wieder eine Diagnose

parametrierbar. Integrierte Funktionen sind den anderen Modulen ähnlich. Eine BU ist wie bei den anderen I/O-Baugruppen erforderlich [Sie14e].

ET 200SP, analoges Eingangsmodul, AI 4xRTD/TC 2-,3-,4-wire HF

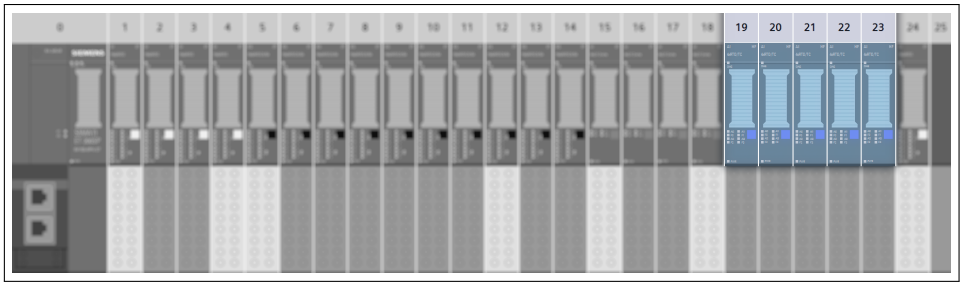


Abbildung 3.7: ET 200SP RTD/TC-Analogeingabemodul (19-23)

Die Analogeingabebaugruppe (vgl. Abbildung 3.7) hat 4 Eingänge mit einer Auflösung von 16 Bit inklusive Vorzeichen. Die Messart ist für jeden Kanal einstellbar. Dieses Modul ist für die eingesetzten Pt100 Sensoren gedacht und wird dementsprechend parametriert. Diagnose und ein Alarm bei Grenzwertüberschreitungen sind integriert und jederzeit abrufbar. Verschiedene Funktionen wie Kalibrierung zur Laufzeit oder Wertstatus bei PROFINET IO sind auch vorhanden [Sie14d].

3.2.2 Lichtsteuerung

Das folgende Konzept für eine Lichtsteuerung beinhaltet mögliche Ein- und Ausgaben im Rahmen der gewünschten Funktionalitäten. Damit ein einfaches Bedienen und eine komfortable Automatik gewährleistet ist, werden die Zusammenhänge und Funktionen tabellarisch aufgegriffen und geordnet. Auf Basis der entstehenden Tabelle (vgl. Tabelle 3.1) können Programmstrukturen erarbeitet und unter Berücksichtigung aller Ein- und Ausgänge Codebausteine programmiert werden.

3.2.3 Jalousiesteuerung

Die Konzepterstellung einer Jalousieansteuerung umfasst mehrere Bereiche der Signalverarbeitung. Es sind nicht nur Interface- oder Schaltereingaben zu beachten. Außerdem sind Wind-, Regen- (vgl. Anlage A.6, Seite A-6), Helligkeits- und ggf. Tempereatursensoriken in die Realisierung einzubeziehen. Damit diese so einfach wie möglich erfolgen kann, ist

Tabelle 3.1: Lichtsteuerungskonzept

Eingaben	Ausgaben
• Interface-Eingaben – Handbetrieb für Bereichsbeleuchtung – Licht ein oder aus und ggf. Dimmfunktion – Arbeitsplatzbeleuchtung getrennt von Bereichsbeleuchtung (separierter Handbetrieb) – Automatik ohne Jalousiesteuerung (abhängig von innen Lux, außen Lux) – Autobetrieb mit Jalousiesteuerung – Definieren verschiedener Helligkeiten (Arbeiten, Besprechung, Präsentation, usw.) – Alarmmodus (z.B. Einschalten) • Hardware-Schalter – Licht ein und aus per Tastendruck – Unterscheidung der Betätigungslänge für Zusatzfunktionen (Dimmen, Zentralschalter...) • Helligkeitssensor im Außenbereich – Analogeingang: 0 - 10 V, 4 - 20 mA – Messbereich: 0 - mind. 60000 Lux – ggf. Außenlichtsteuerung mit Bewegungssensorik – Helligkeitsautomatik mit Jalousie • Helligkeitssensor im Innenbereich – Analogeingang: 0 - 10 V, 4 - 20 mA – Messbereich: 0 - mind. 1000 Lux – Momentane Helligkeit für Automatikbetrieb – Lichtspiele (Dimmbare Leuchten) – Helligkeitsautomatik ohne Jalousie	• Leuchten – Ein- oder Ausschalten • Dimmaktor – Analogausgang: 0 - 10 V, 4 - 20 mA, programmseitig 0 - 100 % – auch digitale Ansteuerung möglich • Interface-Anzeigen – Leuchtenzustand – aktuelle Betriebsart – Alarmfunktion der Beleuchtung aktiv/inaktiv – momentane Helligkeit in Raum oder Außenbereich (möglicherweise über Grenzwerte Bezeichnungen definieren, wie beispielsweise „Direkte Sonneneinstrahlung“) – Lichtspiele (bei dimmbaren Leuchten)

eine tabellarische Übersicht (vgl. Tabelle 3.2) entstanden, die alle denkbaren Funktionen mit entsprechenden Eingaben in Verbindung bringen soll.

3.2.4 Heizungs- und Klimasteuerung

Das Heizen wird in diesem Projekt mithilfe von digitalen Heizkreisventilen von statten gehen. Die Innenraumklimageräte werden über Fernbedienungen ein- oder ausgeschaltet. Die Steuerung kann lediglich die Außengeräte der Klimaanlage unter Berücksichtigung verschiedener Aspekte freigeben (vgl. Tabelle 3.3).

3.2.5 Programmierumgebung

Neben der Hardware ist die Software ein fundamentaler Bestandteil eines Automatisierungssystems. Die Entwicklungsumgebung von Siemens zum Programmieren und Projektieren der aktuellen Siemens Hardware ist das TIA Portal Version 13 (V13) (vgl. Abschnitt 2.3.4, Seite 15). Weiterhin ist eine Lizenz dieser Version bereits innerhalb der Firma vorhanden und muss nicht zusätzlich gekauft werden. Das TIA Portal gehört zum alltäglichen Handwerk der Softwareabteilung. Die Visualisierung wird mithilfe der *WinCC Runtime Advanced* von Siemens für einen Windows-PC realisiert. Dieser ist einfach und unproblematisch über das TIA Portal zu Projektieren und anschließend zu Programmieren. Auf Basis dieses Systems soll am Haupteingang des Neubaus ein Touchscreen installiert werden, der die Aufgabe einer Gesamtübersicht des Gebäudes und weitere Komfortfunktionen übernimmt. Dabei soll natürlich die repräsentative Wirkung auf den Betrachter durch ein modernes Visualisierungsdesign gegeben sein.

Tabelle 3.2: Jalousiesteuerungskonzept

Eingaben	Ausgaben
• Interface-Eingaben – Betriebsart – Auf- oder Abfahren (automatisches Umschalten auf Handbetrieb) – impulsartiges Lamellen verstellen – ggf. maximaler Windbelastungswert änderbar (Zugriffsrechte benötigt) – Referenzfahrt zur Aktualisierung der Maximalfahrzeiten • Hardware-Schalter – Auf, Abfahren mit resultierender Lamellenverstellung – Unterscheidung der Betätigungslänge für Haltefunktion beim Fahren der Jalousie • Helligkeitssensoren im Innen- und Außenbereich – Analogeingang und Messbereich (vgl. Tabelle 3.1) – Beschattung bei direkter Sonneneinstrahlung – Nachtbetrieb (Jalousien schließen) – Automatikbetrieb • Wind- Temperatur und Regensensor – Wind- und Regensensor digital – Temperatur 4 - 20 mA Analogeingang – Schutz vor Beschädigung (Wind, Frostschutz) – Bedienverriegelung bei anstehendem Maximalwind – Reaktionshysterese	• Jalousieaktoren – Motor Auf- und Abfahren (mit Signalhaltung bei langer Tasterbetätigung) – Lamellenverstellung (per Impuls bei HMI-Eingabe und direkt mit Taster) • Interface-Anzeigen – aktuelle Jalousieposition skaliert in einen %-Wert (mithilfe der Fahrzeit) – aktuelle Betriebsart – mittels Grafikliste ggf. Jalousielage und Lamellenstand verdeutlichen

Tabelle 3.3: Heizungs- und Klimakonzept

Eingaben	Ausgaben
• Interface-Eingaben – Betriebsart – Heizen über Vorgabe eines Sollwertes der Temperatur – mögliche Abhängigkeiten der Freigebe des Außengerätes veränderbar (Zugriffsrechte benötigt) • Temperatursensor (Thermowiderstand) im Innen- und Außenbereich – Verriegelung des Kühlaggregats oder Heizkreisventils (mittels Temperaturgrenzen) bei Kälte und Wärme – Bei Frost Heizkreisventil öffnen (Zwangsheizen der Heizung zulassen) – Reaktionshysterese bei über- oder unterschreiten des Sollwertes	• Heizkreisventil – Öffnen oder Schließen • Kühlaggregat – Freigabe bei bestimmter Temperatur unter Einbeziehung gewisser Bedingungen (z.B. PVA-Erzeugung, Energieverbrauch) • Interface-Anzeigen – aktuelle Betriebsart – aktuelle Temperaturwerte(Raum-, Außentemperatur) – Soll-, Hysteresewert

4 Automatisierungslösung

Das folgende Programm stellt die Grundlage für die Gebäudeautomatisierung des Neubaus dar. Alle Programmbausteine wurden mithilfe des vorgestellten TIA Portals programmiert, wobei ein Großteil auf den Sprachen FUP und SCL basieren. Einige wenige Teile sind aufgrund der besseren Übersicht mithilfe von AWL entstanden. Im folgenden Abschnitt werden einzelne Programmbausteine näher erläutert und deren Realisierung anhand von Programmauszügen (vgl. Anlage B) erklärt. Diese beschränken sich auf einige wenige, da eine vollständige sowie untergliederte Version der Programmdokumentation auf der beiliegenden CD-ROM (vgl. Anlage C) alle genannten Codebausteine beinhaltet.

4.1 Probleme

In der Automatisierungstechnik ergeben sich immer wieder Probleme, die von der Konzeptentwicklung bis hin zur letztendlichen Umsetzung der Hard- und Software einer technischen Anlage reichen. Somit ist deren Lösung ein wichtiger Bestandteil der Automation. Solche Probleme können verschiedenster Natur sein, wobei sich das Spektrum vom simplen Platzmangel oder nicht passenden Schrauben bis zu falsch dimensionierten Motoren ausweitet. In dieser Arbeit sollen einige Probleme, die die Hausautomatisierung des Neubaus mit sich bringt, aufgezählt und erläutert werden. Die Problembehebung konnte in den meisten Fällen durch aufwändigere Programmierung erfolgen.

4.1.1 Endlagen der Außenjalousie

Eines der größten Probleme bei der Programmentwicklung bereitet die fehlende Endlagenrückmeldung der Jalousie. Auch der Antrieb selbst gibt keine analoge Positionsinformation aus. Dieser ist aber mit eigenen Endlagen ausgestattet, sodass ein dauerhaftes Signal keine Funktionsbeeinträchtigung darstellt. Doch ohne die Signale gestaltet sich eine Positionsberechnung für deren Anzeige sowie die Realisierung eines automatischen Signalabfalls der Motoransteuerung beim Erreichen einer Endlage problematisch. Eine entsprechende Problemlösung könnte ein Nachrüsten von Näherungssensoren an den jeweiligen Enden der

Jalousie bieten. Damit würde die Fahrzeit die Position der Jalousie kontinuierlich ermittelt und die Ansteuerung der Aktorik mit dem Erreichen der Endlage verriegelt. Eine Konfiguration der maximalen Fahrzeiten ist bei dieser Lösung relativ einfach, da das Programm mittels Referenzfahrten diese selbst definieren kann. In diesem Projekt wurde darauf verzichtet und die zweite kompliziertere Variante entwickelt. Auch hier ist die entscheidende Information die Ansteuerungszeit der jeweiligen Drehrichtung des Antriebes. Aber ohne Endlagen ist ein automatisches Abfallen dieses Signals ohne Zusatzaufwand nicht möglich. Deshalb ist hierbei die Definition der maximalen Fahrzeit, mit deren Hilfe das Ansteuern verriegelt werden kann, umso wichtiger. Nicht lösbar ist die abnehmende Genauigkeit der Positionsanzeige über einen gewissen Zeitraum. Es wird eine manuelle Referenzfahrt, bei welcher der Nutzer die Jalousie in der jeweiligen Endlage selbst stoppen muss, existieren. Diese Lösung wird aber immer mit Ungenauigkeit behaftet sein, weshalb dieser Vorgang in regelmäßigen Zeitabständen wiederholt werden sollte.

4.1.2 Dimmen

Ein Dimmen der Innenraumbeleuchtung ist von der Firma nicht vorgesehen wurden. Dieser Umstand schränkt eine Erstellung von Lichtszenen oder das Erreichen gewisser Helligkeiten ein. Hierbei muss grundsätzlich beachtet werden, dass ein gewisser Spielraum in der Sollwertvorgabe der Helligkeit garantiert und gegebenenfalls Verzögerung beim Schalten der Aktorik gewährleistet werden sollte. Im Vordergrund steht immer der Komfort des Menschen und damit verbunden dessen Wohlbefinden. Aufgrund der nicht vorhandenen Dimmbarkeit bringt ein Ein- und Ausschalten des Lichtes nicht immer den gewünschten Effekt auf den Komfort mit sich. Somit erfordert die in diesem Projekt zum Einsatz kommendn Beleuchtungssteuerung Hysteresen im Automatikbetrieb.

4.1.3 Anwesenheitsregistrierung

Das eingesetzte radio-frequency identification (RFID)-Lesegerät registriert den für jeden RFID-Transponder eindeutigen Identifikator (ID) und dessen Daten. Dieser ID kann in der Software weiterverwendet werden (vgl. Abschnitt 4.6.2, Seite 41). Von der Firma ist eine Identifizierung der Person und deren Registrierung der Anwesenheit im System zum späteren Abrufen gewünscht. Als erste Idee zur Umsetzung dieser Funktion bot sich die Meldeanzeige

vom WinCC. Vorteil dieser Lösung ist eine einfache Einbindung in die Visualisierung. Das Problem hierbei ist die dynamisierte Nutzerregistrierung. Da alle RFID-Tags in das System eingepflegt werden müssen, soll das Hinzufügen neuer unidentifizierter IDs gewährleistet sein, ohne diese direkt im Programm hinterlegen zu müssen. Die Meldeanzeige ist derzeit nicht in der Lage variablenabhängige Texte fehlersicher auszugeben. Deswegen kann die Anzeige des aktuellen Namens des Nutzers mit diesem Chip nicht mittels Meldeanzeige realisiert werden. Die zum Einsatz kommende zweite Möglichkeit wird das Hinterlegen der anwesenden Nutzer in einer Datenbank zur Folge haben. Hierbei kann basierend auf dieser mithilfe verschiedener Skripte eine Anwesenheitsanzeige realisiert werden.

4.1.4 Programmfehler

Während der Programmierung sind sporadisch Programmfehler aufgetreten, weshalb der OB121 implementiert wurde (vgl. Abschnitt 4.3, Seite 36). Derzeit ist dafür noch keine Lösung gefunden wurden. Der Fehler müsste weiterhin beobachtet und im OB121 behoben werden. Eine weiterer Ansatz wäre das Umschreiben der fehlerverursachenden Funktion.

4.2 Hardware-Projektierung

Im TIA Portal wird der reale Steuerungsaufbau hinterlegt. Dort werden alle Baugruppen parametriert und Diagnosefunktionen, Schnittstellen, Adressen, Potenzialgruppen, Messart der I/O-Module, usw. eingestellt. Sobald die gesamte Hardware richtig im Projekt eingearbeitet ist, kann das Programmieren der SPS beginnen. In der sog. Netzsicht (vgl. Abbildung 4.1) werden alle vorhandenen Stationen und deren Vernetzung innerhalb des Projekts angezeigt. Diese sind mittels PROFIBUS (grüne Linien) untereinander verbunden, wodurch zwei Subnetze entstehen. Die Steuerung wird mit der dezentralen Peripherie ET 200SP über die Schnittstelle X1 verbunden, während sie an ihrem Port X2 mit dem Firmenethernet, in dem sich der Visualisierungs-PC befindet, kommuniziert (vgl. Abschnitt 3.2.1, Seite 24). In der zweiten Ansicht, der Topologiesicht, wird die Vernetzung der Stationen am Ethernet-Bussystem aufgezeigt. Hierbei werden die realen Port-Verbindungen deutlich. Eine Deaktivierung der Topologiesicht der CPU kann je nach Hardwareaufbau erfolgen. Diese wird notwendig, wenn die Verbindung zum HMI projektiert werden soll, da sich im Firmenethernet viele Teilnehmer

befinden und somit keine direkte Port-Verschaltung stattfinden kann.

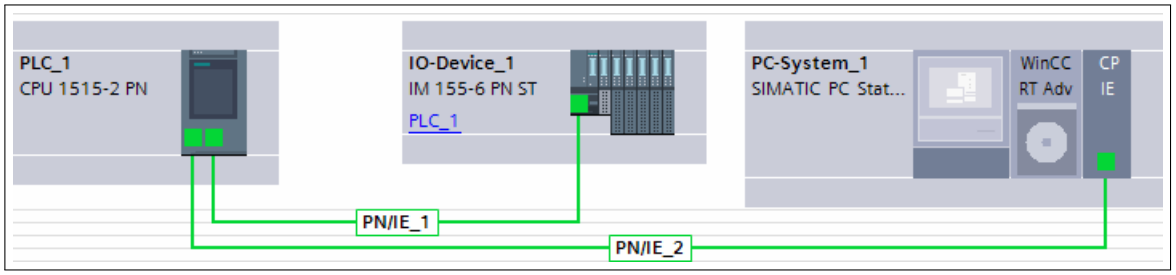


Abbildung 4.1: Netzsicht des Hardwareaufbaus

4.3 Organisationsbausteine

Ein Organisationsbaustein (OB) bildet die Grundlage der Abarbeitung eines SPS-Programms. Dieser arbeitet zyklisch, d.h. ein Programm wird Schritt für Schritt in einer bestimmten Zeit durchlaufen, Eingänge ausgelesen, verarbeitet und Ausgänge geschrieben. Die Zykluszeit hängt von den Eigenschaften eines Organisationsbausteins ab. Hierbei sind außerdem Prioritäten und Alarm-, Fehler- sowie Anlaufoptionen definiert. In Bezug auf Sicherheit und Schnelligkeit wurden in diesem Projekt weniger hohe Anforderungen an das Programm gestellt. Deshalb sind nur der OB1 für das Anwenderprogramm und der OB100 für den Anlauf der CPU sowie der OB121 für das Erkennen eines Programmfehlers im Einsatz.

4.3.1 Main [OB1]

Um die Anforderung, einen CPU-Ausfall erkennen zu können, zu erfüllen, wird am Anfang der Bausteinabarbeitung ein 2 Hz Takt auf einen Ausgang geschrieben. Dieses Signal würde bei einem Ausfall je nach Parametrierung des Hardwareausgangs entweder den zuletzt durchlaufenen Zustand halten oder abfallen. Ein Watchdog-Relais erkennt das Ausharren des Signals auf einem Wert und hat die Aufgabe, eine Notbeleuchtung zu schalten. Des Weiteren können nicht gebrauchte Module über ein sogenanntes „Haupt_ein“-Signal in der Nacht abgeschaltet werden. Dieses steht in Verbindung mit der EMA und ist bei scharfer Alarmanlage deaktiviert. Als nächstes werden die Hardwareeingänge ausgelesen und in Datenbausteinen hinterlegt (vgl. Abschnitt 4.5.2, Seite 41). Im Anschluss werden Alarme abgehandelt, damit diese Global im Anwenderprogramm zur Verfügung stehen. Nun folgt die Abarbeitung des eigentlichen Anwenderprogramms. Die Anordnung innerhalb des OB1

ist an die Ordnerstruktur in der Projektnavigation angelehnt. So ist beispielsweise die Jalousiesteuerung vor dem Heizungs- und Klimaprogrammabschnitt und unmittelbar nach der Lichtsteuerung vorzufinden (vgl. Abbildung 4.2, Seite 37). Zum Schluss werden alle Programmausgaben aus einem Datenbaustein direkt auf die jeweiligen Hardwareadressen übertragen (vgl. Anlage B.1, Seite B-1).

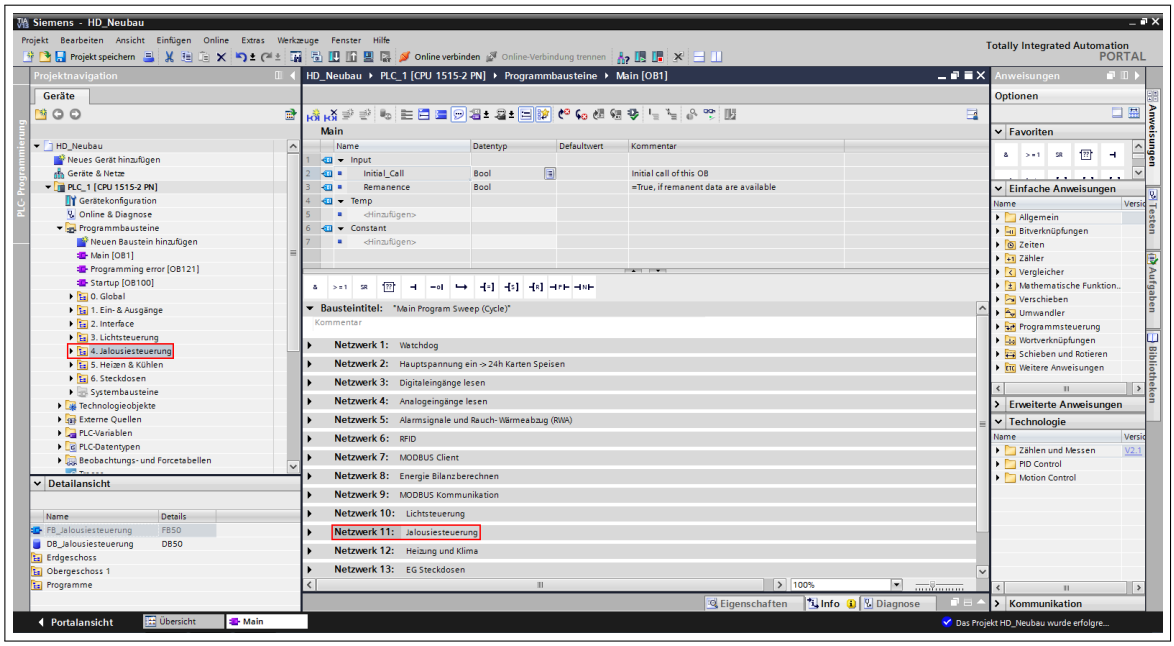


Abbildung 4.2: Struktur des OB1

4.3.2 Startup [OB100]

Der OB100 stellt einen definierten Ausgangszustand nach einem CPU-Ausfall her. In diesem Projekt wird beispielsweise die Modbus-Kommunikation wiederhergestellt. Derzeit ist noch keine weitere Funktion implementiert. Zuerst müssten an dieser Stelle gewollte Ausgangsstellungen für alle Funktionen definiert werden. Dabei wäre ein Auffahren aller Jalousien oder ein Einschalten aller Steckdosen, je nach geforderter Funktionalität, denkbar. Bei Einberechnung der Möglichkeit einer Fehlstellung der Jalousie oder durch Halten der Ausgangssignale, sowie Remanenz bei Datenbausteinvariablen, sind nicht unbedingt alle Angaben im OB100 notwendig. Nach einem CPU-Stopp ist es theoretisch möglich, den Betrieb aufzunehmen ohne den Ausfall bemerkt zu haben. Dies ist mit der Einschränkung verbunden, dass während des CPU-Stopps auf Eingaben keine Reaktion mehr erfolgt. Trotz solcher Möglichkeiten ist

der OB100 für den erwähnten Kommunikationsaufbau sowie für eventuelle Aktivitäten im Nachhinein vorgesehen.

4.4 Global

4.4.1 Datenbausteine

Globale Datenbausteine dienen dem Programmierer zur Sammlung und Verwaltung verschiedener Prozessdaten. Alles was im Programm global zur Verfügung stehen soll, kann zu besserer Übersicht und Kommunikationszwecken in Datenbausteinen hinterlegt werden. In diesem Projekt stehen insbesondere Nutzereingaben und Nutzerdaten im Vordergrund, da Beleuchtung, Jalousie, Arbeitsplätze, usw. auf diese reagieren sollen. Somit werden beispielsweise Interface-Eingaben in einem speziell dafür vorgesehenen Datenbaustein „DB_HMI [DB3]“ und Nutzerdaten in einem Nutzer-DB „DB_Nutzer [DB3]“ hinterlegt. Alarm- und Zustandssignale füllen einen zweiten DB. Außerdem ist ein Simulations-Datenbaustein während der Programmerstellung zum Einsatz gekommen, der als spätere Diagnosemöglichkeit wahrgenommen werden kann. Die Ansteuerungen für Heizung, Jalousie und Beleuchtung benötigen Instanz-Datenbausteine, welche Unterprogramme instanziiert um Speicher zu sparen und die Übersicht zu wahren. In den Instanzen können Merker und Timer genutzt werden ohne auf den CPU-Adressbereich zuzugreifen. S7 IEC-Timer basieren ohnehin auf Datenbausteinen, wodurch dieser Umstand effektiv genutzt werden kann.

4.4.2 Funktionen und Funktionsbausteine

Die Funktion „Außenhelligkeit [FC1]“ wird überall dort, wo eine Außenhelligkeit beschrieben werden muss, eingesetzt. Dieser Baustein steht somit global zur Verfügung und gibt eine Beschreibung sowie eine verwertbare Integer-Zahl der aktuellen Außenhelligkeit aus.
Für die Alarmauswertung und die Reaktion des Rauch- Wärmeabzugs darauf, beschreibt der Funktionsbaustein (FB) „FB_RWA_Alarme [FB44]“. In diesem werden durch Timer Verzögerungen eines Signalabfalls realisiert, um beispielsweise ein verfrühtes Freischalten der Jalousie bei Wind zu verhindern oder ein Frostgefahr-Signal zu generieren.
Mit dem „FB_Speicherfunktion [FB96]“ können Prozesswerte in einem Array gespeichert werden. Mithilfe des Befehls „MOVE_BLK_VARIANT“ ist es möglich, einen Wert mit dem

Datentyp Variant in Abhängigkeit eines Indexes, bei einer positiven Flanke eines Trigger-Bits in die jeweilige Array-Position zu schreiben. Danach wird der Index erhöht und das Array kann erneut beschrieben werden. Dieser FB wird zum Großteil bei der Positionsberechnung der Jalousie benötigt. Die Speicherung erfolgt wie bei einem Umlaufspeicher, d.h. nach dem Ankommen am Array-Ende wird der erste Wert überschrieben und von dort aus weitergeführt (vgl. Anlage B.2, Seite B-6).

4.4.3 Anwenderdatentypen

Nutzer

Der *Nutzer*-Anwenderdatentyp beinhaltet alle notwendigen Informationen zur Beschreibung eines Nutzers innerhalb des Programms. Dieser ist aufgebaut wie eine Struktur und enthält Datentypen wie *Bool* für Anrede, *Int* für die Arbeitsplatznummer, *Lint* für die Transponder-ID oder *Strings* für Vor- und Nachname sowie einen Abteilungsnamen. Dieser Anwenderdatentyp ist im Datenbaustein (DB) „DB_Nutzer [DB3]“ vertreten.

Licht

Als Schnittstelle zwischen HMI und Programm fungiert der *Licht*-Anwenderdatentyp. Hier wird ein virtueller Schalter zur Lichtschaltung, Automatik- und Handbetrieb-Umschaltung sowie ein Helligkeitssollwert und eine Zeit zum Ausschalten einer Leuchte bei Nichtanwesenheit des Benutzers definiert. Zu guter letzt wird einer Variable im „DB_HMI [DB3]“ dieser Datentyp, je nach Etage und Bereich sowie Arbeitsplatz, zugeordnet.

Jalousie

Der *Jalousie*-Anwenderdatentyp beschreibt Interface-Eingaben für die Jalousiesteuerung genauso. Dabei werden beispielsweise Jalousiefahrten, -position, -fahrzeiten usw. im Datentyp verankert. Auch dieser wird anschließend im „DB_HMI [DB3]“ für jede Jalousie eines Arbeitsbereiches einer Variable zugeordnet.

Kühlaggregat

Der Anwenderdatentyp *Kühlaggregat* dient zur Veränderung der Freischaltbedingungen eines Kühlaggregates über das Interface. Die Einbindung erfolgt gleichermaßen im selben

Datenbaustein wie bei zuvorigen Datentypen.

4.5 Ein- und Ausgänge

4.5.1 Adressierung

Häufig ist das Verarbeiten der Datenpunkte der erste Schritt in der Programmierung. Damit der Programmierer weiß welche Adressen für welchen Zweck gedacht sind, ist eine symbolische Adressierung mit logischen und vor allem eindeutigen Namen unabdingbar [Kaf14]. Alle Eingangs- und Ausgangsadressen können relativ einfach mithilfe des Namens im Programm eingepflegt und verarbeitet werden. In diesem Projekt wurde dieser Prozess anhand einer vorher angelegten Excel Tabelle zur Übersicht aller benötigten Datenpunkte unterstützt (vgl. Anlage C: Haus-Automatisierung (I/O-Tabelle)). Die I/O-Belegung der SPS-Baugruppen ist logisch nach Etagen oder Bereichen des Gebäudes (vgl. Anlage A.1, Seite A-1 und A.2, Seite A-2) und Anwendung angelegt, um dadurch spätere Suchen nach Adressbereichen zu vereinfachen. So sind z.B. Beleuchtung und Jalousie nacheinander Etagenweise angeordnet. Nachdem die Hardwareadressen ausgelegt werden konnten, wird die erwähnte symbolische Adressierung durchgeführt. Die Grundidee bei diesem Projekt ist das Zusammenlegen der Bereiche, in dem die einzelnen Datenpunkte tätig sein werden. Hierbei werden verschiedene Variablentabellen, wie z.B. Beleuchtung, Jalousie, Steckdosen, Heizen und Kühlen, usw., erstellt (vgl. Anlage C: Programmdokumentation). Eine solche symbolische Adresse ist in Etage, Bereich und Gerät (vgl. Abbildung 4.3) unterteilt.

Beleuchtung

		Name	Datentyp	Adresse	Kommentar
13		=G31+AB01-S01	Bool	%E2.0	Schalter - Arbeitsbereich

Abbildung 4.3: Adresse eines Bereichsschalters für die Beleuchtung

Hierbei bedeutet „=G31“ Gebäude 3 Obergeschoss 1, „+AB01“ Arbeitsbereich 1 und „-S01“ Schalter 1 in diesem Bereich. Mithilfe von Kommentaren an den einzelnen Variablen ist die Anwendung ersichtlich. Schalter für die Beleuchtung werden vor den Jalousie-Tastern immer zuerst nummeriert. So ist an vielen Stellen sofort klar, um welchen Schalter mit welcher Funktion es sich handelt.

4.5.2 Signalverarbeitung

Die Softwareabteilung der Firma arbeitet bei der Signalbehandlung innerhalb eines STEP 7 (S7)-Programms mit Datenbausteinen, in welche alle Ein- und Ausgänge über Funktionen mit einem AWL-Quelltext geschrieben und von dort aus im ganzen Programm weiterverwendet werden. Diese sind je nach Ein- oder Ausgang am Anfang oder am Ende des Organisationsbausteins angeordnet. Somit werden zu Beginn eines Bearbeitungszyklus des OB1 alle hardwareseitigen Eingänge gelesen und im entsprechenden DB hinterlegt, sowie zum Schluss Programmausgaben ausgelesen und auf die entsprechenden Hardwareausgänge geschrieben. Unterschieden werden normale und ganztägige (24h) Ein- und Ausgänge. Für diesen Unterscheidungsvorgang ist jeweils ein DB vorgesehen und entsprechend bezeichnet.

4.6 Interface

4.6.1 Modbus-Kommunikation

Mittels fertigen Siemens Bausteinen ist eine Modbus-TCP (vgl. Abschnitt 2.2.3, Seite 10) Kommunikation realisiert. Diese hat den Zweck, verschiedene Informationen über Netzbezug und -einspeisung der Firma zu erhalten. Die Steuerung kann mit diesen und denen der PVA einen aktuellen Stromverbrauch berechnen. Dies ist für eine spätere Anzeige interessant. Die Realisierung der Programmbausteine wurde von der Firma übernommen und ist nicht Inhalt dieser Arbeit gewesen.

4.6.2 Nutzeranmeldung

Wie bereits erwähnt wird mithilfe eines RFID-Lesegerätes jeder Nutzer registriert und in einem Datenbaustein dessen Daten hinterlegt. Auf dem Visualisierungs-PC sorgt ein Programm für die Datenkommunikation zwischen der Steuerung und dem RFID-Lesegerät. Auf Basis dieser Information ist die SPS in der Lage, einen automatischen Bildwechsel zu realisieren. Im HMI kann als Bereichszeiger ein Steuerungsauftrag projektiert werden, der in der Steuerung als Array aus vier Variablen mit dem Datentyp *WORD* angelegt und diesem zugewiesen wird. Um nun einen Bildwechsel im HMI zu erzwingen, wird die Konstante für den Bildwechsel in die Variable *Steuerungsauftrag[0]* „51“ und die geforderte Bildnummer in die Variable *Steuerungsauftrag[1]* eingetragen (vgl. Abbildung 4.4).

```
// Bildwechselauftrag nur wenn das aktuelle Bild nicht schon aktiv ist
IF "DB_HMI".Panel.aktives_Bild <> "DB_HMI".Panel.Bild.Homescreen THEN
    "DB_HMI".Steuerungsauftrag[0] := #auftragsnummer;
    "DB_HMI".Steuerungsauftrag[1] := "DB_HMI".Panel.Bild.Homescreen;
    "DB_HMI".Panel.aktives_Bild := "DB_HMI".Panel.Bild.Homescreen;
END_IF;
```

Abbildung 4.4: Steuerungsauftrag

Die restlichen Array-Variablen werden für diese Anwendung nicht beschrieben. Grundsätzlich ist es nun möglich, jeden Nutzer automatisch auf sein individuelles Bild zu verweisen, um von dort seine gewünschten Einstellungen vorzunehmen.

FB_RFID [FB97]

In diesem FB werden alle Reaktionen auf eine Nutzeranmeldung realisiert. Dabei wird in der Funktion „FC_aktueller_Nutzer [FC1]“ über die Transponder-ID des Chips detektiert, ob die aktuelle ID im System bereits einem Nutzer zugeordnet werden kann. Wenn derjenige vorhanden ist, werden die anliegenden Nutzerdaten aus dem Datenbaustein „FB_Nutzer [DB3]“ in die Struktur „aktueller_Nutzer“ im „DB_HMI [DB4]“ geschrieben und über eine weitere Funktion „FC_AB_Screens [FC2]“ auf ein „Lobby-Bild“ weitergeleitet. Wenn der Nutzer nicht im System vermerkt ist, wird derjenige über einen Bildwechsel darauf entsprechend hingewiesen. Dabei gibt es die Möglichkeit, die ID im System zu hinterlegen. Aktionen dieser Nutzeranmeldung sind z.B. das Zuschalten des Arbeitsplatzes und das Abschalten bei Abmeldung. In dem FB ist außerdem ein automatischer Bildwechsel nach einem Timer-Ablauf, sofern keine Eingabe getätigt wird, realisiert. Dadurch kann verhindert werden, dass das Nutzerbild bestehen bleibt, sollte derjenige nicht selbst quittieren. So kann einem unberechtigten Eingreifen in die Licht-, Heizungs- oder Jalousiesteuerung vorgebeugt werden (vgl. Anlage B.3, Seite B-8).

4.7 Lichtsteuerung

Die Grundidee der Programmierung dieser Lichtsteuerung ist eine bewusste Instanziierung aller Funktionalitäten innerhalb der einzelnen Räume. Dies hat den Vorteil, dass der komplette Abschnitt innerhalb eines Instanz-Datenbausteins realisiert werden kann. Mithilfe von duplizierten Funktionsbausteinen werden Stockwerke voneinander getrennt, worin sich wiederum

die Räume wiederfinden lassen. Somit ist im letztendlichen OB1 ein einziger Funktionsbaustein aktiv, an dem globale Signale angeschlossen werden können. Die gesamte Lichtsteuerung ist über universelle Funktionsbausteine realisiert. Diese sind sowohl im Erdgeschoss als auch in den Büroräumen einsetzbar (vgl. Anlage C: Programmdokumentation).

4.7.1 Prioritätsschalten [FC7]

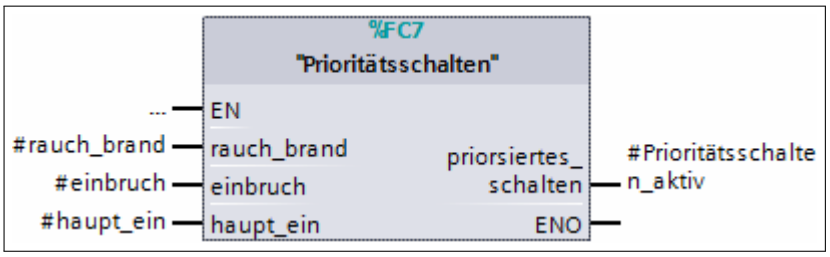


Abbildung 4.5: Baustein Prioritätsschalten [FC7]

Die Beleuchtung soll beim Anstehen eines Alarmsignals einen definierten Zustand einnehmen. So werden beispielsweise alle Leuchten nach dem Einschalten der Alarmanlage ausgeschaltet. Dieser Codebaustein ist dauerhaft aktiv und benötigt deshalb einen Merker (*#ausführen*), der die Bearbeitung freigibt. Der Ausgang *priorisiertes_schalten* zeigt einerseits, dass die Funktion aktiv ist und sorgt andererseits für das Deaktivieren der Funktionsbausteine, die für den Normalbetrieb zuständig sind. Dadurch ist es möglich, Tastereingaben bei Nacht zu verriegeln und ein nicht gewolltes Einschalten zu verhindern. Alle Leuchten werden direkt über ihren Datenbausteineintrag angesprochen und nehmen den geforderten Zustand über den Merker *#schaltart* ein.

4.7.2 Licht_Handsteuerung [FB35]

Das Herzstück der Lichtsteuerung bildet der FB35 (vgl. Abbildung 4.6). Dieser ist universell aufgebaut, sodass er alle gebrauchten Funktionen der Handsteuerung abwickeln kann. Grundsätzlich erfolgt ein Schalten des Lichtes entweder mithilfe eines Schalters oder über gewisse Zustände. Jeder Raum besitzt einen Bereichsschalter, dessen Betätigung vorher vom FB37 als lang oder kurz identifiziert wird (vgl. Abschnitt 4.7.4, Seite 45). Bei längerer Betätigung werden je nach Anwendung mehrere Leuchten geschaltet. In den Arbeitsbereichen werden so beispielsweise alle Leuchten ausgeschaltet, sofern noch eine eingeschaltet sein

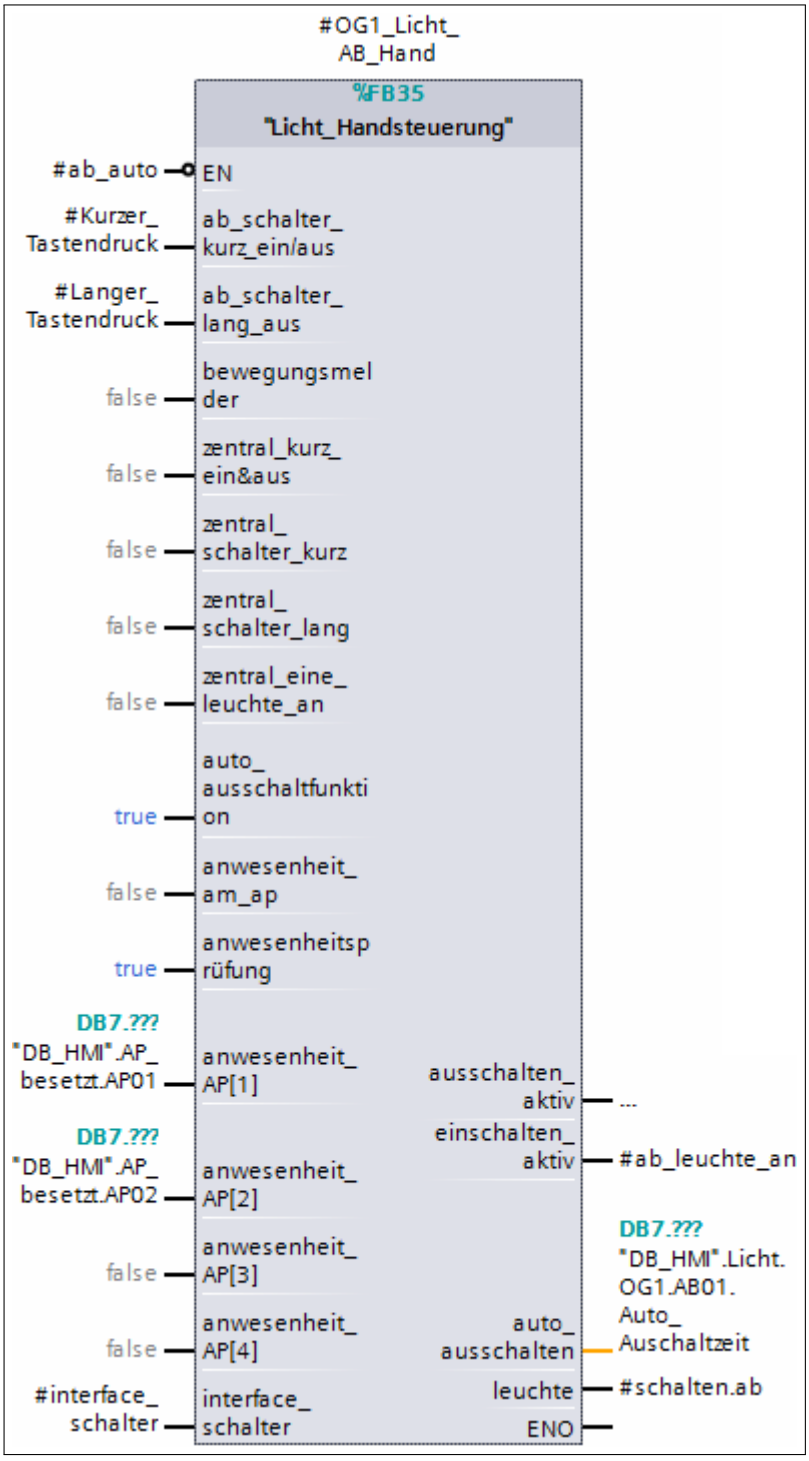


Abbildung 4.6: Baustein Licht_Handsteuerung [FB35]

sollte. Ein gesonderter Interface-Schalter und ein Bewegungsmelder-Eingang sorgen für den selben Effekt wie bei einer kurzen Tasterbetätigung des Hardwareschalters für ein Ein- und Ausschalten der Leuchte. Zusatzfunktionen für die Nutzeranwesenheitsschaltung sind mit

den jeweiligen Eingängen möglich. Wenn niemand in gewissen Arbeitsbereichen angemeldet ist, wird mithilfe eines Timers nach einer festen Zeit ein Ausschalten der Leuchte erwirkt. Außerdem sind Merker für den aktuellen Leuchtenzustand nach außen geführt. Auch hier sind Zusatzfunktionen für eine Zentralschaltung möglich.

4.7.3 Licht_Automatik [FB36]

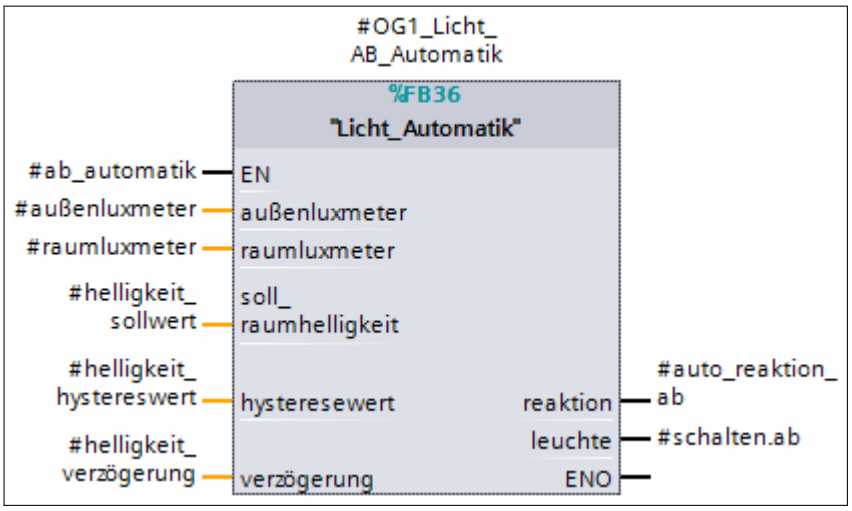


Abbildung 4.7: Baustein Licht_Automatik [FB36]

Die automatische Lichtsteuerung ist aufgrund der derzeit nicht dimmbaren Leuchten lediglich zum Einschalten bei Dunkelheit und Ausschalten, falls es zu hell ist, zuständig. Dafür wird ihr eine Raumhelligkeit sowie ein Hysteresewert und eine Schaltverzögerungszeit vorgegeben. Raum- und Außenluxmeter sorgen für die entsprechenden Istwerte. Innerhalb des Bausteins wird über die Funktion „Außenhelligkeit [FC1]“ die aktuelle Außenhelligkeit unterschieden und einem Zahlenwert zugeordnet (vgl. Abschnitt 4.4.2, Seite 38). Dieser wird in einer CASE-Anweisung abgehandelt und die entsprechenden Aktionen werden zugewiesen. Wenn kein Raumluxmeter am Baustein angeschlossen ist, wird nur in Abhängigkeit der Außenhelligkeit geschaltet.

4.7.4 Tastendruckerkennung [FB37]

Die Tastendruckerkennung spielt eine Rolle für Sonderfunktionen. Es soll z.B. möglich sein, mit dem Flurlichtschalter einmal das Flurlicht ein- und aus-, aber auch die komplette Beleuchtung im Erdgeschoss abzuschalten. Als Grundlage für das Definieren solcher Aktionen

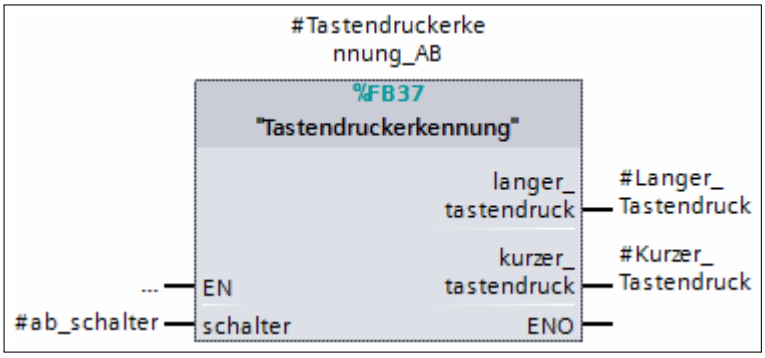


Abbildung 4.8: Baustein Tastendruckerkennung [FB37]

dient der FB37. Er unterscheidet ein Halten eines Tasters mittels Timer und verriegelt eine mehrfache Erkennung durch einen Impuls am Ausgang. Ein kurzer Tastendruck bleibt für einen Zyklus aktiv. So kann relativ schnell ein zweiter erkannt werden.

4.7.5 Zentralfunktion [FB38]

Wie bereits erwähnt ist im Erdgeschoss eine Zentralschaltung vorgesehen. Wird der Zentralschalter kurz bestätigt, schaltet sich das Durchgangslicht, bestehend aus den ersten drei Arbeitsbereichsleuchten und Flurlicht, im EG ein. Bei einem langen Tastendruck wird die Beleuchtung im gesamten EG ausgeschaltet. Die Leuchtenzustände und die Interface-Eingaben werden an den Funktionsbaustein übergeben. Der Baustein registriert, ob eine Leuchte an ist und setzt einen Ausgang zur Weiterverarbeitung. Außerdem wird der Interface-Zentralschalter im Baustein behandelt und je nach Leuchtenzustand gesetzt und rückgesetzt, um auf der Visualisierung anzuzeigen, ob ein Ausschalten von Nöten ist oder nicht.

4.8 Jalousiesteuerung

Wie schon bei der Lichtsteuerung wird auch bei der Jalousiesteuerung mittels Unterinstanzen programmiert. Dabei wird ein Funktionsbaustein die gesamte Ansteuerung der Jalousien beinhalten. Auch hier stellt die Übersichtlichkeit im Main [OB1] einen Vorteil dar. Die folgenden Programmteile werden dann für jeden Raum dupliziert und in der entsprechenden Instanz eingepflegt (vgl. Anlage C: Programmdokumentation).

4.8.1 Prioritätsfahren [FC8]

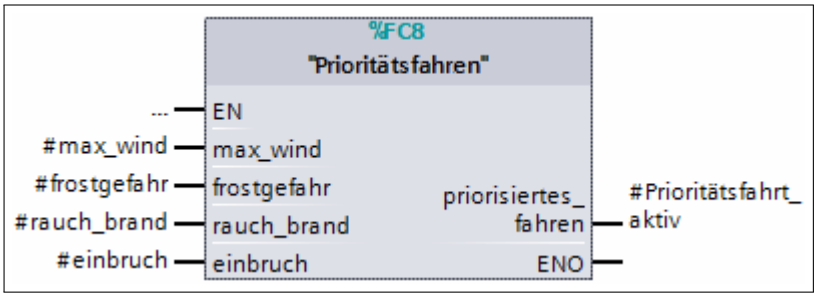


Abbildung 4.9: Baustein Prioritätsfahren [FC8]

Diese Funktion gilt als Pendant zum Prioritätsschalten innerhalb der Lichtsteuerung. Genau wie dort ist es wichtig, dass die Jalousien bei anstehendem Alarmsignal in eine definierte Position fahren. Dieser Baustein ist allen Funktionen übergeordnet, d.h. er verriegelt alle Eingaben oder Ausgaben des Normalbetriebes indem er die Instanz-Funktionsbausteine, die dafür verantwortlich sind, deaktiviert (vgl. Anlage C: Programmdokumentation). Realisiert wird ein Hochfahren beim Überschreiten des maximal zulässigen Windes für die Jalousien bei Frostgefahr (Außentemperatur unter 5°C), bei einer Brand oder Rauchwarnung von der EMA und bei Einbruchalarm. Durch den Ausgang *priorisiertes_fahren* wird die angesprochene Verriegelung der Eingaben bewirkt.

4.8.2 Nachtbetrieb [FB62]

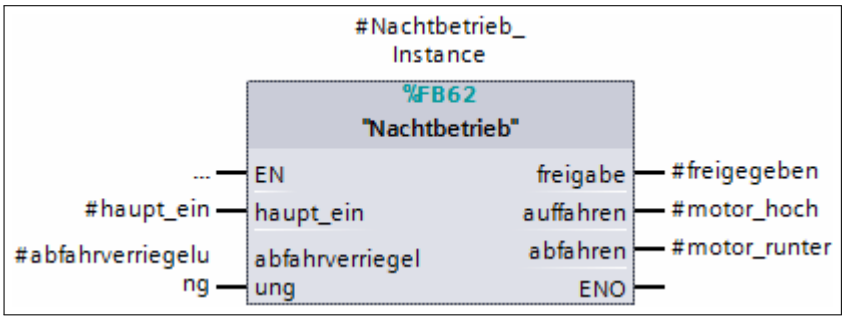


Abbildung 4.10: Baustein Nachtbetrieb [FB62]

In der Nacht ist es erforderlich, die Jalousien soweit zu verriegeln, dass sie nicht mehr manuell fahrbar sind. Damit dies geschieht, wird durch diesen Funktionsbaustein der Normalbetrieb überhaupt erst freigeben. Dadurch ist die Hand- und Automatiksteuerung erst tagsüber funktionstüchtig (vgl. Anlage C: Programmdokumentation). Das „Haupt_ein“-Signal sorgt für

genau diesen Zustand. Damit die Jalousie nicht weiter angesteuert wird wenn sie unten ist, wird mithilfe des Eingangs „abfahrverriegelung“eine unnötige Ansteuerung verhindert.

4.8.3 Jalousie_Handsteuerung [FB63]

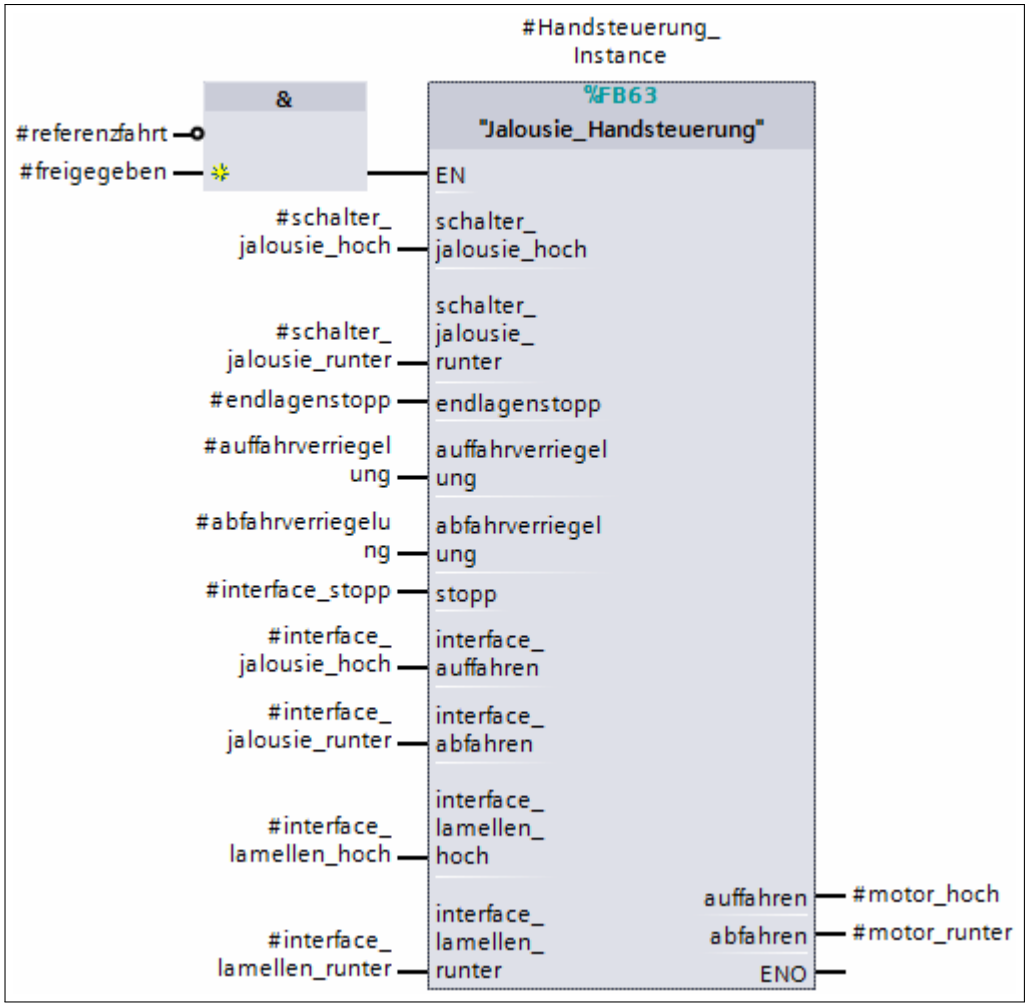


Abbildung 4.11: Baustein Jalousie_Handsteuerung [FB63]

Einen zentralen Punkt der Jalousiesteuerung stellt eine korrekte und komfortable Handsteuerung dar. Die Herausforderung besteht hierbei in der Verbindung zwischen den realen Schaltern und den Interface-Eingaben. Es soll beispielsweise eine Handeingabe auf der Anzeige sichtbar werden. Mithilfe verschiedener Merker für das Ansteuern wird der Bezug hergestellt. Zum Stoppen dient innerhalb dieses Funktionsbausteins ein Stopp-Merker. An diesen wird die Interface-Stopp-Eingabe gelegt. So kann einerseits bei einem kurzen Betätigen des Jalousie-Tasters oder andererseits mit dem Interface ein Stopp der Jalousie hervorgeru-

fen werden. Damit der Merker nicht dauerhaft beschalten ist, wird mit Flanken gearbeitet. Diese garantieren eine Bedienung vom Interface aus. Ähnlich ist der Ansatz beim Fahren der Jalousie. Im HMI wird ein Fahren, welches den Ausgang hält und ein Verstellen der Lamellen, das impulsweise arbeitet, unterschieden. Für diese beiden Arten der Ansteuerung sind Merker definiert, auf die das Interface zugreift und die von Schaltern je nach Betätigungslänge gesetzt werden können. Wenn die Jalousie in einer Endlage ankommt, kann sie mit einem gewollten Endlagenstopp zum Anhalten werden. Dabei werden lediglich Merker rückgesetzt und stattdessen der Stopp-Merker durch die entstehende negative Flanke gesetzt. Die Endlagenmeldungen kommen vom FB65.

4.8.4 Jalousie_Automatik [FB64]

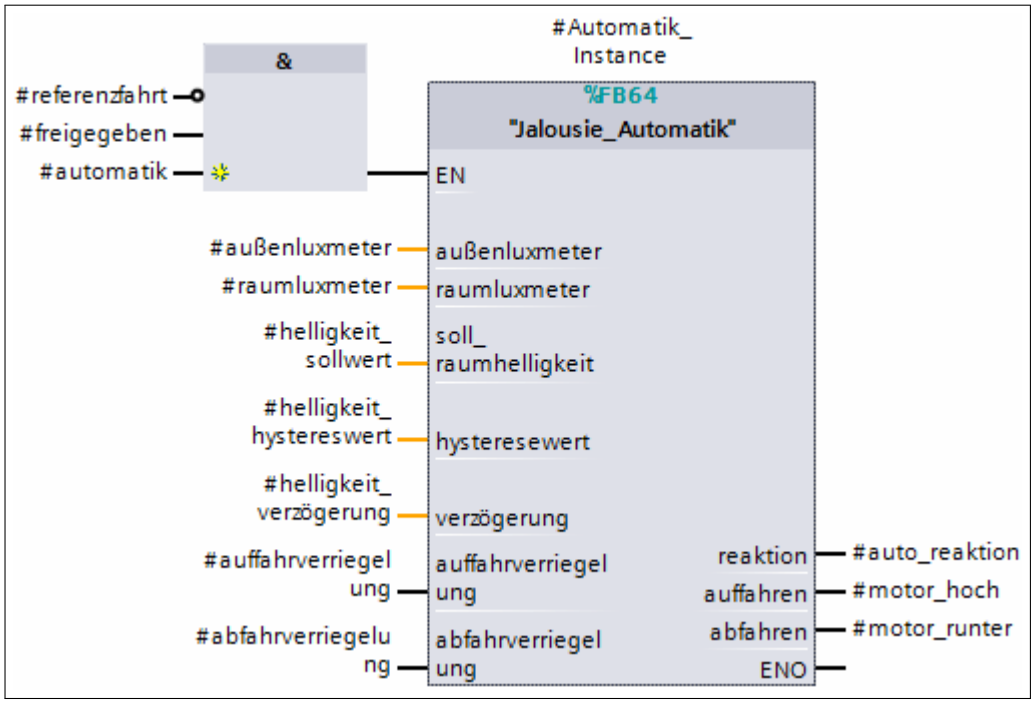


Abbildung 4.12: Baustein Jalousie_Automatik [FB64]

Auch in der Automatikansteuerung der Jalousie wird mit Raum- und Außenhelligkeit gearbeitet. Es besteht aber auch hier das Problem, dass nicht in allen Räumen Innenraumluxmeter verbaut sind. Somit muss die Automatik auf zwei Arten funktionieren. Entweder auf Basis einer Raumhelligkeitsregelung mit untergeordneter Außenhelligkeit oder ausschließlich außenhelligkeitsabhängig. Dem Baustein werden außerdem die üblichen Parameter der Hel-

ligkeit und die Auf- und Abfahrverriegelungssignale übergeben. Auf Grundlage dieser wird der Ausgang zur Ansteuerung der Jalousieaktoren beschaltet. Die alleinstehende Außenhelligkeitsregelung sieht lediglich eine Beschattung bei direkter Sonneneinstrahlung vor. Dabei wird beim Ankommen der Jalousie in der unteren Endlage ein Auf-Impuls erzeugt, der für eine beschattete Öffnung der Jalousie sorgen soll. Die Raumhelligkeitsregelung dient sowohl dem Beschatten als auch der Helligkeitserhaltung im Raum. Über- oder unterschreitet der Wert eine Grenze, wird der statische Merker zum Auf- oder Abfahren gesetzt. Alle Schaltvorhaben sind am Ausgang „reaktion“abzulesen. Die Bausteinausgaben, die für ein reales Fahren der Jalousie sorgen, werden zeitverzögert durchgeschaltet.

4.8.5 Positionsberechnung [FB65]

Dieser Funktionsbaustein ist notwendig, um die aktuelle Position der Jalousie zu erkennen (vgl. Abschnitt 4.1.1, Seite 33). Wie bereits erwähnt wird dies auf Basis der Fahrzeiten erfolgen. Als erstes wird ein Index erzeugt, der jeweils den letzten gespeicherten Wert indiziert. Die aktuellen Fahrzeiten werden mittels Timer erfasst und nach einem Stoppsignal (Trigger) gespeichert (vgl. Abschnitt 4.4.2, Seite 38). Für die Anzeige wird jeweils der letzte Zeitwert in einem DB hinterlegt. Anschließend werden die Zeiten normiert, mit dem letzten normierten Positionswert verrechnet und in einem Array gespeichert. Der FB65 gibt außerdem mit einer Verzögerung, die ein komplettes Auf- oder Zufahren der Jalousie garantieren soll, eine Verriegelung der Ansteuerung aus. Mithilfe einer IF-Anweisung wird der normierte Positionswert auf seinem Maximum gehalten um Verfälschungen der Position zu verhindern. Zur Anzeige kommt der skalierte Positionswert, der am Bausteinende auf den entsprechenden Ausgang geschrieben wird. Zuletzt wird auch ein Anzeigenrücksetzen realisiert, das bei Berechnungsfehlern die Positionsberechnung auf den Ausgangszustand zurückführt (vgl. Anlage B.4, Seite B-14).

4.8.6 Referenzfahrt [FB66]

Die Referenzfahrt hat den Hintergrund, Maximalfahrzeiten in beide Richtungen zu aktualisieren. Bei großen Jalousien sind die Fahrzeiten richtungsabhängig unterschiedlich. Außerdem können diese aufgrund von Verschleißerscheinungen der Motoren variieren. Der FB66 richtet die Jalousie, indem er sie auffährt, bis das Ankommen am oberen Ende vom Benutzer

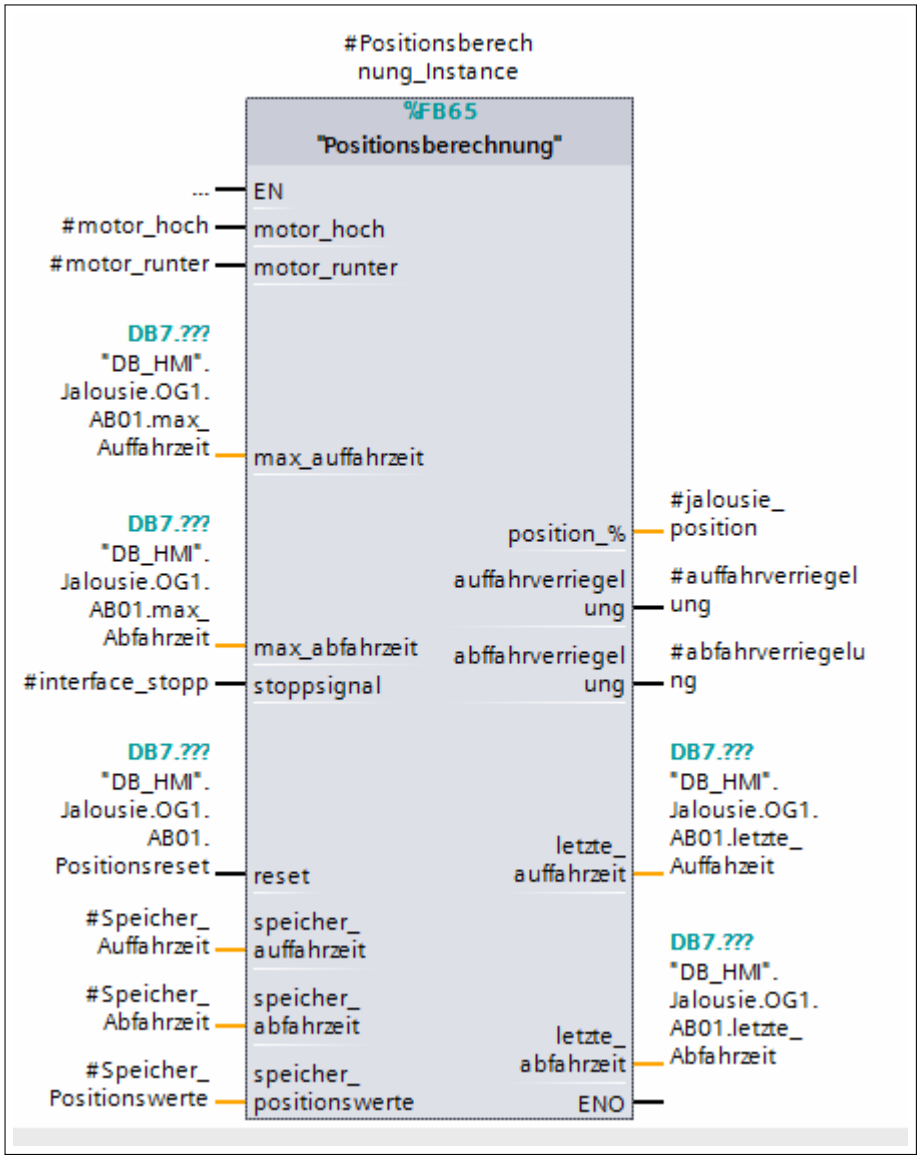


Abbildung 4.13: Baustein Positionsberechnung [FB65]

bestätigt wird. Danach wird ein Abfahren bewirkt und der Nutzer bestätigt im Moment des Ankommens der Jalousie am unteren Ende. Diese Zeit wird gespeichert und am Ende übernommen. Nach dem selben Prinzip wird auch die Hochfahrzeit bestimmt. Zum Schluss der Bearbeitung kann der Nutzer die Werte übernehmen, neu aufnehmen oder verwerfen (vgl Abbildung 4.14). Der aktuelle Bearbeitungsschritt wird in einer Integer-Variablen *#Bild* hinterlegt. Im HMI ist dadurch mithilfe von Sichtbarkeiten ein Menü für die Referenzfahrt entstanden (vgl. Anlage B.5, Seite B-19).

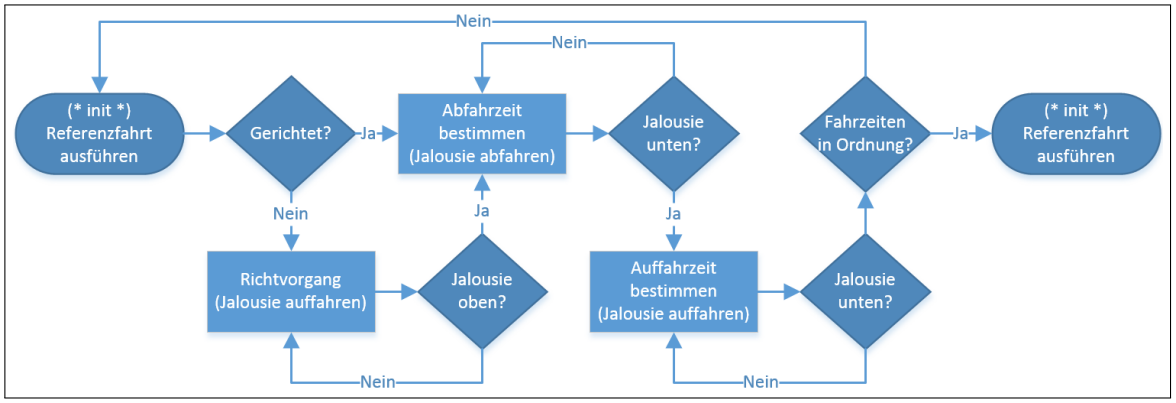


Abbildung 4.14: Prinzip der Referenzfahrt

4.9 Heizen und Kühlen

Die Bausteine für die Heizung oder Kühlaggregate werden in einem dafür vorgesehenen FB angeordnet (vgl. Anlage C: Programmdokumentation). Dieser kann dann im Main [OB1] aufgerufen und abgearbeitet werden. Unterinstanzen sind nicht notwendig, da keinerlei Gedächtnis zur Abarbeitung der Funktionen benötigt wird. Alle gebrauchten Werte werden aus einem globalen DB entnommen.

4.9.1 Heizungssteuerung [FC9]

Der Heizungssteuerung werden die Raumtemperatur mit Grenzen und Hysterese sowie Bits zur Veränderung des Sollwertes vorgegeben. Beim reinitialisieren der Datenbausteine werden alle Variablen mit ihren Startwerten initialisiert, weswegen am Anfang des Bausteins der Sollwert festgelegt wird. Auf diesen kann der Nutzer entweder basierend auf den dafür vorgesehenen Bits oder direkt Einfluss nehmen. Damit die Abstände immer schrittweise gleich bleiben, wird der Wert entweder gerundet oder die Kommastelle auf jeweils ein halbes Grad gesetzt. Wird der maximal mögliche Sollwert der Raumtemperatur über- oder unterschritten, erfolgt ein Überschreiben auf das Maximum mittels IF-Anweisung. Des Weiteren sind je nach angegebenem Differenzwert Reaktionsgrenzen des Heizkreisventils definiert. Dieser ist standardmäßig gleich 2°C, es sei denn, keine Grenzen sollen verwendet werden. Für diesen Fall wird er auf 0°C gesetzt. Natürlich kann das Interface genauso beeinflusst werden. Die eigentliche Ansteuerung des Heizkreisventils ist auch nur bei aktivem Haupt_ein Signal möglich. Letztendlich kommt es zum Heizen, wenn der Grenzwert unterschritten wird.

4.9.2 EG_Kühlaggregat [FC10]

Für das Erdgeschoss sind derzeit keine Temperatursensoren angeschlossen und im Programm somit nicht verarbeitet. Für eine eventuell spätere Nachrüstung sind aber vorgesehene Eingänge bereits vorhanden und mit den geplanten Sensoren verschaltet. Die Freischaltung des Kühlaggregates erfolgt bei unscharfer Alarmanlage basierend auf der Außentemperatur. Befindet sich diese über einer gewissen Temperatur, kann im EG gekühlt werden. Als Besonderheit gilt das Einbeziehen der PVA-Erzeugung. So würde diese das Kühlen verhindern, wenn sie zu gering ausfällt. Es ist möglich, die Funktion über das HMI zu deaktivieren.

4.9.3 OG1_Kühlaggregate [FC11]

Im Grunde funktioniert die Freischaltung der Kühlaggregate genauso wie beim FC10 mit dem Unterschied, dass hier größtenteils Innenraumtemperaturen vorliegen. Somit ist die Einbeziehung der Außentemperatur nur noch optional. An einem Kühlaggregat sind mehrere Raumgeräte angeschlossen, wodurch der wärmste Raum für eine Freischaltung des Aggregates sorgt. Dies wird innerhalb des Programms basierend auf einem Vergleich aller Raumwerte mit dem Sollwert der Raumtemperatur realisiert. Die Verknüpfung dieser Vergleiche mit einem ODER stellt sicher, dass der wärmste Raum berücksichtigt und somit ein Kühlen ermöglicht wird. Aufgrund der Individualität dieses Bausteins kann die dritte Raumtemperatur nur einbezogen werden, wenn es sich um das Außengerät handelt, welches nicht wie üblich zwei sondern drei Innengeräte versorgt.

4.10 Steckdosen

Im Neubau sind die Steckdosen nicht dauerhaft einfach aktiv. Je nach Anwesenheit, Funktion oder Stockwerk variieren die Bedingungen in ihrer Speisung. Die dafür verantwortlichen Funktionen sind direkt im Main [OB1] verankert. Ein Testen der Steckdosen ist durch das Deaktivieren der folgenden Funktionen möglich.

4.10.1 EG_Steckdosen [FC12]

Das Erdgeschoss besitzt unterschiedliche Verwendungsorte von Steckdosen. Es soll zum Beispiel ein Laden von Geräten über Nacht möglich sein. Dieser Umstand setzt eine Unterschei-

dung der Steckdosen voraus. Die Warmwasserpatrone ist abhängig von der Stromerzeugung der PVA und wird somit erst geschaltet, wenn ein Stromüberschuss zu verzeichnen ist. Alle AB-Steckdosen sind über Nacht abgeschaltet, um unnötige Stand-By-Verbraucher aus dem Netz zu nehmen.

4.10.2 OG1_Steckdosen [FC13]

Diese Funktion besitzt die gleiche Aufgabe wie die oben genannte. Der einzige Unterschied besteht darin, dass die Benutzeranwesenheit eine Freischaltung der Arbeitsplatz-Steckdosen zur Folge hat. Die Erkennung der Arbeitsplatzanwesenheit erfolgt in einem dafür vorgesehenen Baustein FC_AP_Anwesenheit.

4.11 Visualisierung

4.11.1 Aufbau und Navigation

Vom Grundbild „Home“ ist die Navigation durch die öffentlichen Bilder möglich. Hierbei können Energiedaten und später auch Anwesenheiten eingesehen werden. Eine nachträgliche Erweiterung der Funktionen und des Bildumfangs ist wünschenswert. In der oberen Navigationsleiste sind vier Symbole zu sehen, links davon eine Temperaturanzeige und in der rechten oberen Ecke ein Auswahlmenü. In dieses können weitere Funktionen eingepflegt werden. Der Hauptaugenmerk liegt derzeit auf den Gesamtübersichten der Stockwerke. Diese werden später zu den Administrator-Bildern gehören, da von dort aus alle Eingaben zur Steuerung gemacht werden können. Es ist eine öffentliche Variante dieser Übersichten geplant, doch die Implementierung steht noch aus. Die wesentlich komplexeren Bildstrukturen benötigt die Nutzeranmeldung und -regisitrierung. Diese werden über Steuerungsaufträge der SPS aufgerufen (vgl. Abschnitt 4.6.2, Seite 41). In dieser Struktur sind alle Arbeitsbereichs- und Nutzerverwaltungsbilder hinterlegt. Alle Gegenstände der Visualisierung können der beiliegenden CD-ROM entnommen werden (vgl. Anlage C: Programmdokumentation (vollständig) oder Anlage C: Visualisierungs-PC).

Nutzernavigation

Nachdem ein Nutzer sich an dem RFID-Lesegerät angemeldet hat, wird dieser entweder auf das Bild „Lobby“ oder „Nutzer_Fehler“ weitergeleitet. Wenn die ID des Transponders nicht vorhanden ist, kann der Benutzer sich mithilfe des Bildes „neuer_Nutzer“ hinzufügen oder zum Startbild zurückkehren. Bei vorhandener ID wird der hinterlegte Nutzer mit seinem Namen begrüßt. Von hier aus kann er auf seinen Arbeitsplatz navigieren und Einstellungen vornehmen. Des Weiteren kann er sich abmelden, indem er weitere Bilder wie den „Abmeldescreen“ und den „A-Bestätigungsscreen“ , der den positiven Bescheid einer erfolgreichen Abmeldung präsentieren soll, durchläuft (vgl. Abbildung 4.15). Auch sind Erweiterungen in allen Bereichen möglich und sollten nach der Inbetriebnahme des Systems individuell angepasst werden.

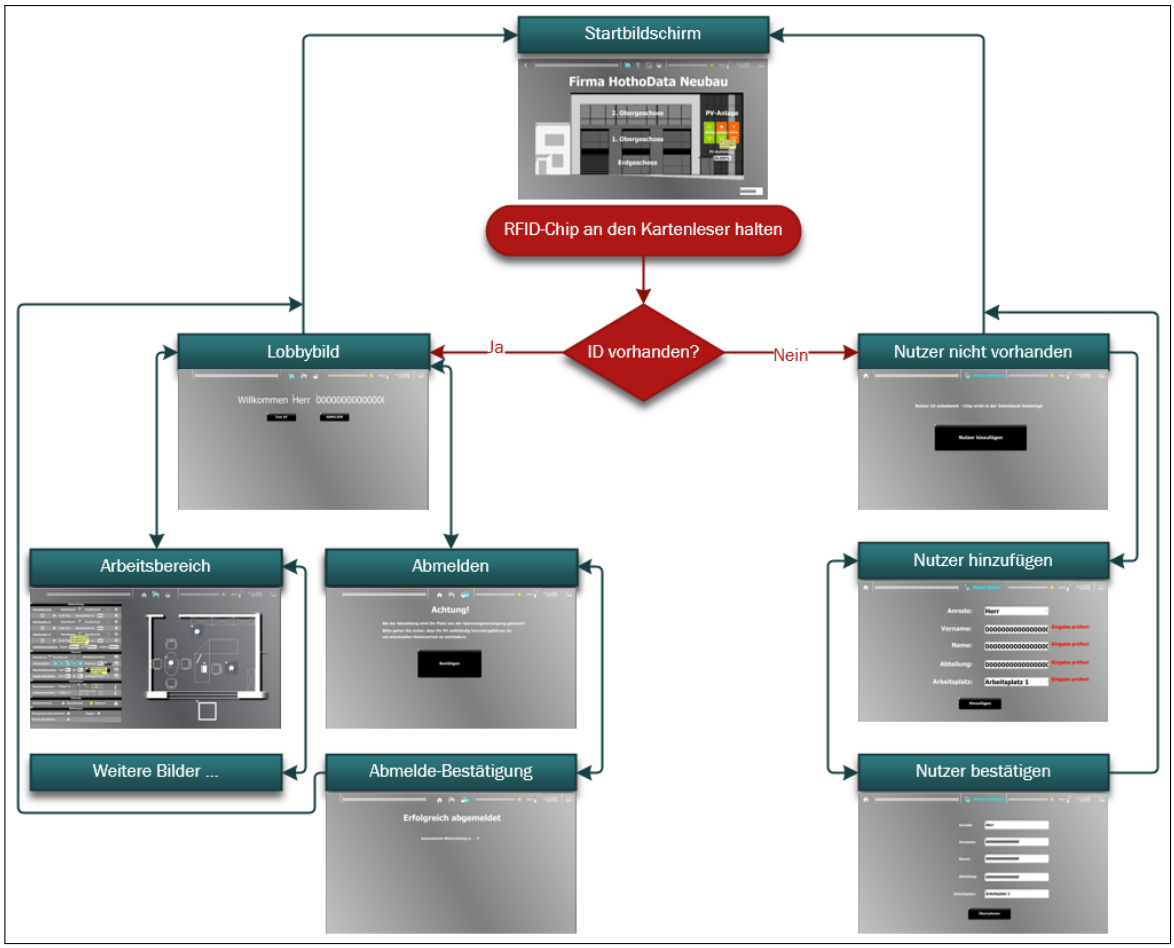


Abbildung 4.15: Nutzernavigation

4.11.2 Skripte

In WinCC werden Skripte auf Basis der Programmiersprache Visual Basic for Applications (VBA) geschrieben. Diese Sprache ist eine von Microsoft entwickelte, ereignisorientierte Codesprache für Makros. Die Sprache wird nicht zum programmieren eigenständiger Programme genutzt. Alle in dem Projekt vorkommenden Skripte dienen dem Ausführen gewisser Routinen. Die zugehörigen Quelltexte können den Anlagen entnommen werden (vgl. Anlage C: Programmdokumentation).

„screen_info“

Dieses Skript dient dem Zweck, den aktuellen Titel des Bildes in eine String-Variable zu schreiben. Außerdem wird die Sichtbarkeit eines Objekts verändert. Es wird im Aufgabenplaner nach jedem Bildwechsel ausgeführt.

„animation“

In der Visualisierung kommen zwei animierte Objekte mit hinterlegten Grafiklisten vor. Die Animation wird mithilfe des CPU-Taktes und diesem Skript erzeugt. Es sind insgesamt fünf Grafiken vorhanden, die die Animation hervorrufen sollen. In Abhängigkeit einer Integer-Variablen werden sie weitergeblättert. Damit die Animation von vorn beginnen kann, wird unter Zuhilfenahme dieses Skriptes die Variable zurückgesetzt.

„appear_info“

Informationen zu erstellten Funktionalitäten sind wichtig, um den Benutzer auf gewisse Dinge aufmerksam zu machen. Damit dies aber auch noch optisch hochwertig von statten gehen kann, ist es die Aufgabe des Skriptes einerseits das Bit für die Informationsanzeige zurückzusetzen und andererseits das Erscheinen eines Informationsdialoges zu verschönern. Nachdem ein Infobutton auf der Visualisierung aktiviert wurde, gleitet die Infobox aus dem Button heraus nach oben. Das Gleiten wird mithilfe des Skriptes „wait“ realisiert.

„wait“

Diese Funktion ist als untergeordneter Programmcode zu verstehen. Dieser wird an mehreren Stellen innerhalb der Skripte benötigt und kann somit einheitlich dort verwendet werden. Der

Inhalt dieses Skriptes beschränkt sich auf einen Timer, der die Skriptabarbeitung verlangsamen soll. So wird ermöglicht, dass eine Variable, die das Verschieben eines Objektes bewirkt, langsam erhöht werden kann und dadurch der Gleiteffekt entsteht. Dem Skript wird eine Zeit als Parameter vorgegeben, wobei die Timerschnelligkeit nach oben hin begrenzt ist.

„slide_up“ und „slide_down“

Auch diese Skripte nutzen den Gleiteffekt. Auf Basis dieser wird ein Auswahlmenü realisiert, welches mehrere Objekte gleiten lässt.

„show_time“

Das letzte Skript verarbeitet den Datentyp TIME. Dieser wird in Millisekunden an das HMI übergeben und ist somit für den Benutzer eher unkomfortabel zu lesen. Dieses Skript berechnet Sekunden und Minuten aus den gegebenen Millisekunden. Diese werden dann mit einem Doppelpunkt von einander getrennt und in eine String-Variable übernommen. Wenn ein Zeitablauf dargestellt werden soll, muss die maximale Zeit als Parameter übergeben werden.

5 Schlussbemerkung

5.1 Zusammenfassung

Die Gebäudeautomation spielt eine bedeutende Rolle für die Moderne. Der immer wichtiger werdende Energieaspekt motiviert Gesellschaft und Technik, diesen Bereich kontinuierlich weiterzuentwickeln. In diesem Zusammenhang gibt es Systeme, deren Ziel eine immer effizientere Gebäudeautomatisierung darstellt. Das Anliegen dieser Arbeit ist das Programmieren und Dokumentieren einer Automatisierungslösung eines solchen Vorhabens. Für eine solche Umsetzung erwiesen sich im Vornherein einige Schritte als notwendig. Zuerst musste das System unter Berücksichtigung verschiedener Kriterien gewählt werden. Anhand dessen wurde sich für eine Lösung mittels Speicherprogrammierbarer Steuerung entschieden. Anschließend erfolgte die Planung des Steuerungsaufbaus, der für die zu realisierenden Aufgaben benötigt wird. Dazu mussten zukünftige Ein- und Ausgaben analysiert, strukturiert und entsprechenden Baugruppen zugeordnet werden. Je nach Automatisierungsumfang musste eine möglichst günstige und passende Steuerung gewählt werden. Um dies zu erreichen, wurde der Aufbau geplant und in Absprache mit dem Projektverantwortlichen kalkuliert. Im Anschluss konnten alle Baugruppen bestellt und in den Schaltschrank verbaut werden. Neben der Hardware galt es die Software zu programmieren, welches den größten Anteil dieser Arbeit in Anspruch nahm. Dafür entstand vorerst ein Konzept, das während der Programmierung als funktioneller Leitfaden diente. Außerdem wurden hardwareseitige Ein- und Ausgänge auf Basis von symbolischen Adressen in das Softwareprojekt eingearbeitet, um die Programmierung zu erleichtern. Letztendlich konnte unter Zuhilfenahme der Steuerungskonzepte bezüglich der Vorgaben an die Automatisierungslösung ein Programm entstehen, das einen Großteil der gewollten Anforderungen erfüllt. Unter den realisierten Funktionen befanden sich beispielsweise eine automatisierte Beleuchtungssteuerung mit Betätigungslängenunterscheidung, eine Jalousiesteuerung, eine Heizungs- und Klimasteuerung, ein belegungsabhängiges Arbeitsplatzfreischalten, eine Anmeldung und eine zugehörige Visualisierung. Für das Umsetzen der Visualisierung entstanden verschiedene Skripte, die diese funktionell oder optisch aufwerteten.

Nicht alle Funktionalitäten konnten aufgrund von zeitlichen und technischen Engpässen ausreichend getestet oder implementiert werden, weshalb im Nachhinein noch Beobachtungen und Verbesserungen zu tätigen sind.

5.2 Ausblick

In Anlehnung an das Beschriebene sind Verbesserungen bezüglich der Hard- und Software wünschenswert. Das angesprochene Problem der Endlagenrückmeldungen der Jalousie könnte vermieden werden, indem beispielsweise Sensoren nachgerüstet werden, die gleichzeitig die Referenzfahrt hinfällig machen könnten sowie eine kontinuierliche Positionsaktualisierung zur Folge hätten. Die fehlende Dimmbarkeit der Leuchten ist ein großes Manko der automatisierten Beleuchtungssteuerung im Neubau. Da sich die Firma aufgrund verschiedener Umstände gegen eine Nachrüstung benötigter Aktorik entschied, ist dafür vorerst keine Lösung in Sicht. Die Anwesenheitsregistrierung erfordert eine ausführliche Testphase, um die vollständige Funktionalität testen zu können. Diese sollte vor dem schlussendlichen Einzug stattgefunden haben, um ungewollte Verzögerungen zu vermeiden. Gleichzeitig müsste die Programmabarbeitung beobachtet und kontinuierlich angepasst und verbessert werden. Die Programmerstellung wurde durch außerplanmäßige Änderungen immer wieder unterbrochen und verlangsamt. Des Weiteren war der Neubau erst spät für Testungen vorbereitet und somit konnten viele Funktionen erst in der Schlussphase der praktischen Phase dieser Arbeit erarbeitet und eingepflegt werden. Im Großen und Ganzen besteht bei der Programmierung weiterhin Anpassungs- und Änderungsbedarf. Viele der umgesetzten Funktionen können möglicherweise noch individueller gestaltet und je nach Bedarf verbessert werden. Für die Zukunft wäre eine Erweiterung gewisser Funktionen im Programm und der Visualisierung anzustreben. Einige Umsetzungsvorschläge sind noch offen und benötigen eine Änderung des Quellcodes. Abschließend ist noch einiges an Arbeit zu leisten, damit das automatisierte Gebäude letztendlich den Komfort- und Qualitätsansprüchen einer modernen Hausautomatisierung entspricht.

Literaturverzeichnis

[Ber12a] BERGER, Hans: *Automatisieren mit SIMATIC - Controller, Software, Programmierung, Datenkommunikation, Bedienen und Beobachten*. 5. Aufl. Erlangen : Publicis Kommunikationsag, 2012. – ISBN 978–3–8957–8677–8

[Ber12b] BERGER, Hans: *Automatisieren mit STEP 7 in KOP und FUP - Speicherprogrammierbare Steuerungen SIMATIC S7-300/400*. 6. überarb. u. erw. Auflage. Erlangen : Publicis Kommunikationsag, 2012. – ISBN 978–3–8957–8411–8

[Ber14] BERGER, Hans: *Automatisieren mit SIMATIC S7-1500 - Projektieren, Programmieren und Testen mit STEP 7 Professional*. Erlangen : Publicis Kommunikationsag, 2014. – ISBN 978–3–89578–403–3

[Kaf14] KAFTAN, Jürgen: *SPS-Aufbaukurs mit SIMATIC S7*. 3. Aufl. Würzburg : Vogel Business Media, 2014. – ISBN 978–3–8343–3255–4

[KHS09] KRIESEL, Werner ; HELM, Peter ; SOKOLLIK, Frank: *KNX/EIB für die Gebäudesystemtechnik in Wohn- und Zweckbau -*. 5. völlig neu bearbeitete und erweiterte Auflage 2009. Heidelberg; München; Landsberg; Frechen; Hamburg : Hüthig, 2009. – ISBN 978–3–7785–4054–1

[Kre14] KRETZER, Peter: *TIA Portal V13 Tipps und Tricks Automatisierungsforum Mai 2014*. Mai 2014. – [Stand 04. September 2014]

[MHH10] MERZ, Hermann ; HANSEMANN, Thomas ; HÜBNER, Christof: *Gebäudeautomation - Kommunikationssysteme mit EIB/KNX, LON und BACnet*. 2., neu bearbeitete Auflage 2010. München; Wien : Fachbuchverl. Leipzig im Carl-Hanser-Verlag, 2010. – ISBN 978–3–4464–2152–3

[Sch12] SCHMITT, Karl: *SPS-Programmierung mit ST - nach IEC 61 131 mit CoDeSys und mit Hinweisen zu STEP 7 V11 ; [Tools, Tests und Lösungen auf CD-ROM]*. 1. Aufl. Würzburg : Vogel, 2012. – ISBN 978–3–8343–3251–6

[Sie12a] SIEMENS AG: *Broschüre: SIMATIC STEP 7 im Totally Integrated Automation Portal Intuitives und effizientes Engineering – vom Microcontroller bis zum PC-basierten Controller*. Version: November 2012. http://www.automation.siemens.com/salesmaterial-as/brochure/de/brochure_tia_portal_de.pdf. – [Online; Stand 04. September 2014]

[Sie12b] SIEMENS AG: *Gerätehandbuch: SIMATIC ET 200SP Digitalausgabemodul DQ 16x24VDC/0,5A ST (6ES7132-6BH00-0BA0)*. Version: Dezember 2012. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=48741164939&GuiLanguage=de&query=A5E03574418-02&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E03574418-02 [Online; Stand 04. September 2014]

[Sie12c] SIEMENS AG: *Gerätehandbuch: SIMATIC ET 200SP Digitaleingabemodul DI 16x24VDC ST (6ES7131-6BH00-0BA0)*. Version: Dezember 2012. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=48740927883&GuiLanguage=de&query=A5E03573440-02&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E03573440-02 [Online; Stand 04. September 2014]

[Sie13] SIEMENS AG: *Gerätehandbuch: SIMATIC ET 200SP Interfacemodul IM 155-6 PN ST (6ES7155-6AU00-0BN0)*. Version: April 2013. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=52945769739&GuiLanguage=de&query=A5E03576903-02&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E03576903-02 [Online; Stand 04. September 2014]

[Sie14a] SIEMENS AG: *Broschüre: SIMATIC WinCC Das skalierbare und offene SCADA System für maximale Anlagentransparenz und Produktivität*. Version: 2014. https://c4b.gss.siemens.com/resources/images/articles/e20001-a820-p810.pdf. – [Online; Stand 16. September 2014]

[Sie14b] SIEMENS AG: *Gerätehandbuch: BaseUnits (6ES7193-6BP...)*. Version: Februar

2014. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=62910475403&GuiLanguage=de&query=A5E03727048-04&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E03727048-04 [Online; Stand 04. September 2014]

[Sie14c] SIEMENS AG: *Gerätehandbuch: SIMATIC ET 200SP Analogausgabemodul AQ 4xU/I ST (6ES7135-6HD00-0BA1).* Version: Juli 2014. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=69647681931&GuiLanguage=de&query=A5E03573364-AC&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E03573364-AC [Online; Stand 23. September 2014]

[Sie14d] SIEMENS AG: *Gerätehandbuch: SIMATIC ET 200SP Analogeingabemodul AI 4xRTD/TC 2-/3-/4-wire HF (6ES7134-6JD00-0CA1).* Version: Februar 2014. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=63143677835&GuiLanguage=de&query=A5E03573288-AD&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E03573288-AD [Online; Stand 04. September 2014]

[Sie14e] SIEMENS AG: *Gerätehandbuch: SIMATIC ET 200SP Analogeingabemodul AI 4xU/I 2-wire ST (6ES7134-6HD00-0BA1).* Version: Juli 2014. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=69646117259&GuiLanguage=de&query=A5E03572831-AC&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E03572831-AC [Online; Stand 04. September 2014]

[Sie14f] SIEMENS AG: *Gerätehandbuch: SIMATIC S7-1500 CPU 1515-2 PN (6ES7515-2AM00-0AB0).* Version: Februar 2014. http://www.automation.siemens.com/mdm/default.aspx?Language=de&ShowMsg=false&DocVersionId=63947404299&GuiLanguage=de&query=A5E32332363-AA&cssearchengine=NEW. – Dokument-Identifikationsnummer: A5E32332363-AA [Online; Stand 04. September 2014]

[Sie14g] SIEMENS AG: *HMI Software*. Version: 2014. https://mall.industry.siemens.com/collaterals/files/3/JPG/G_ST80_DE_00461j.JPG. – [Online; Zugriff 23. September 2014]

[Sie14h] SIEMENS AG: *Industry Mall: Katalog- und Bestellsystem für Automatisierungs- und Antriebstechnik*. Version: 2014. https://mall.industry.siemens.com. – [Online; Stand 16. September 2014]

[SW12] SCHNELL, Gerhard ; WIEDEMANN, Bernhard: *Bussysteme in der Automatisierungs- und Prozesstechnik: Grundlagen, Systeme und Anwendungen der industriellen Kommunikation*. 8., aktualisierte u. erw. Aufl. Wiesbaden : Springer-Vieweg, 2012. – ISBN 978–3–83480–901–2

[WZ14] WELLENREUTHER, Günter ; ZASTROW, Dieter: *Automatisieren mit SPS - Theorie und Praxis - Programmierung: DIN EN 61131-3, STEP7, CoDeSys, Entwurfsverfahren, Bausteinbibliotheken. Applikationen: Steuerungen, Regelungen, Antriebe, Safety. Kommunikation: AS-i-Bus, PROFIBUS, Ethernet-TCP/IP, PROFINET, Web-Technologien, OPC, WLAN*. 5. korr. u. erw. Aufl. 2011. Wiesbaden : Vieweg+Teubner Verlag, 2014. – ISBN 978–3–8348–1504–0

Abbildungsverzeichnis

Tabellenverzeichnis

Anlagenverzeichnis

Selbstständigkeitserklärung

Hiermit versichere ich, Emanuel Klann,
dass ich diese vorliegende Arbeit mit dem Thema:

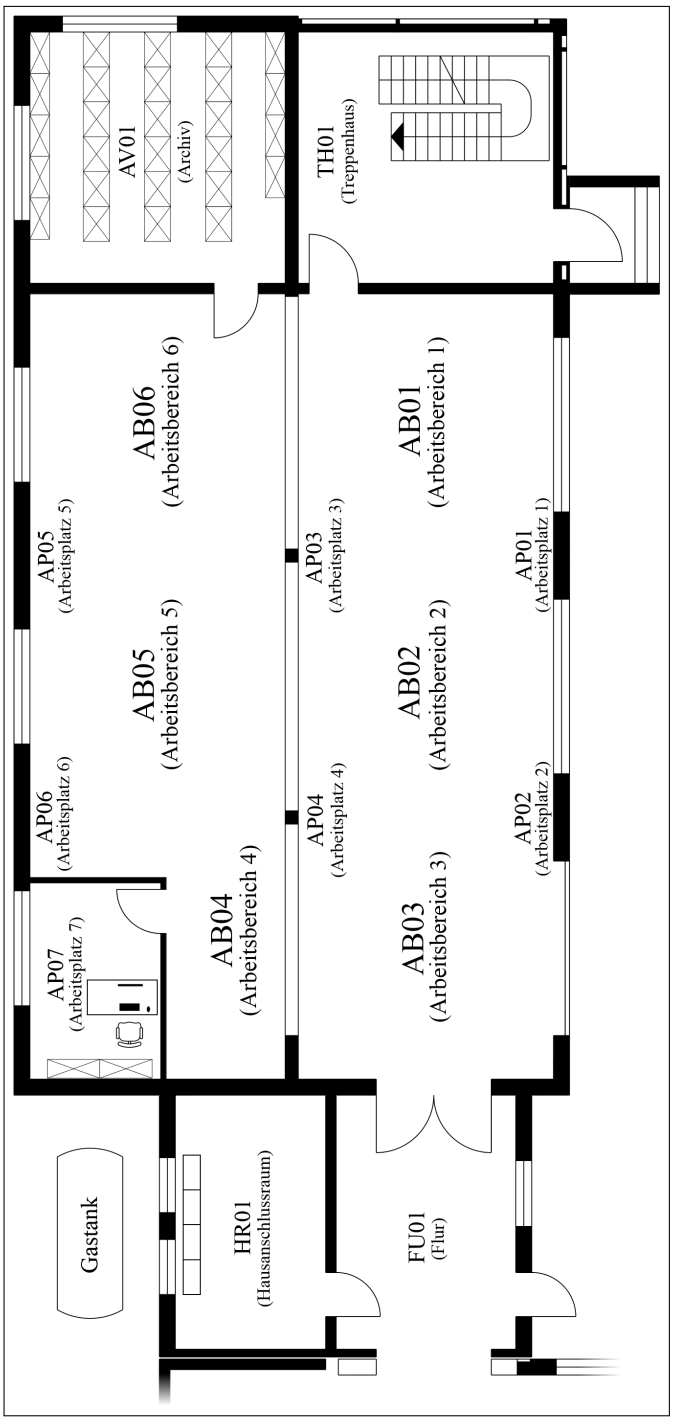
Entwurf und Programmierung einer Automatisierungslösung für ein Bürogebäude

selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

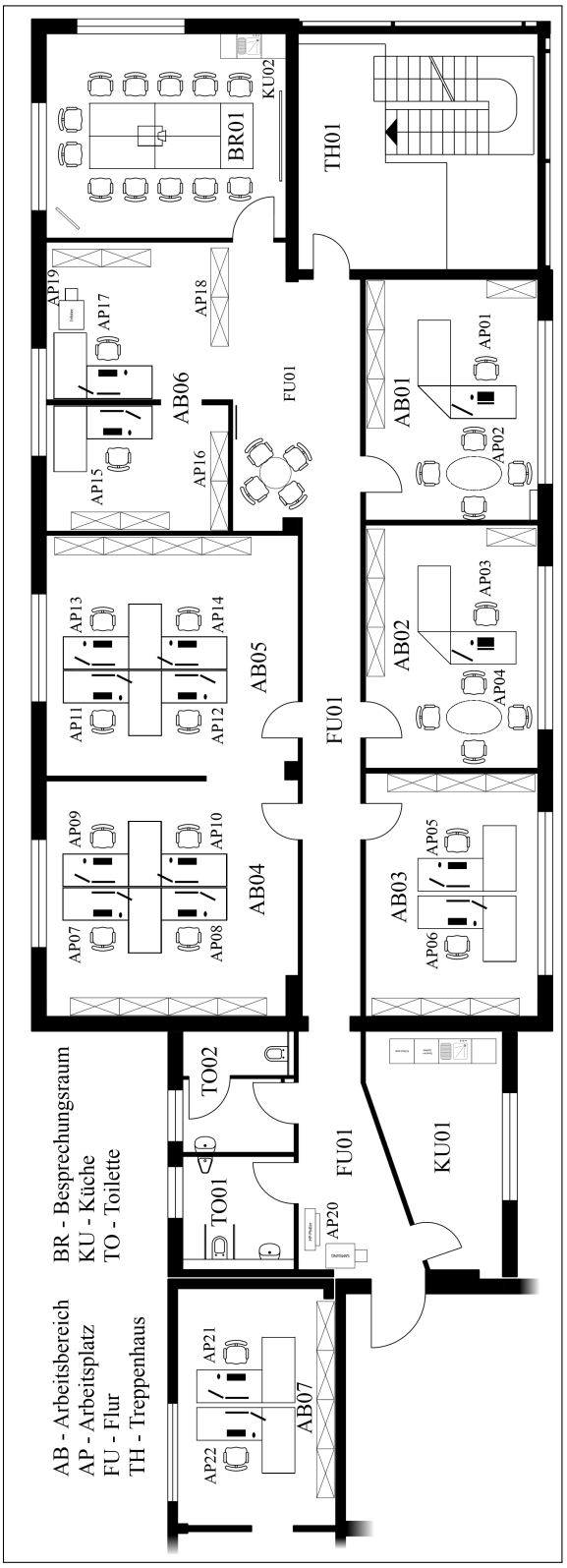
Ort, Datum	Unterschrift

Anlage A: Der Neubau

A.1 Grundriss des Erdgeschosses



A.2 Grundriss des ersten Obergeschoss



A.3 ET 200SP Hardwareaufbau

A.4 ET 200SP Interfacemodul

A.5 Schaltschrank

A.6 Wind- und Regenrelais mit Messeinrichtung

Anlage B: Programmbausteine

B.1 Main [OB1]

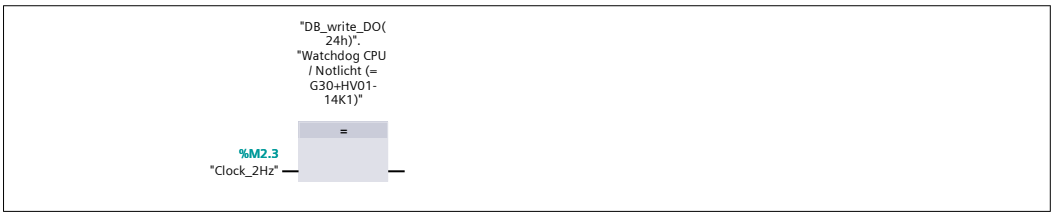
Totally Integrated Automation Portal		

Main [OB1]

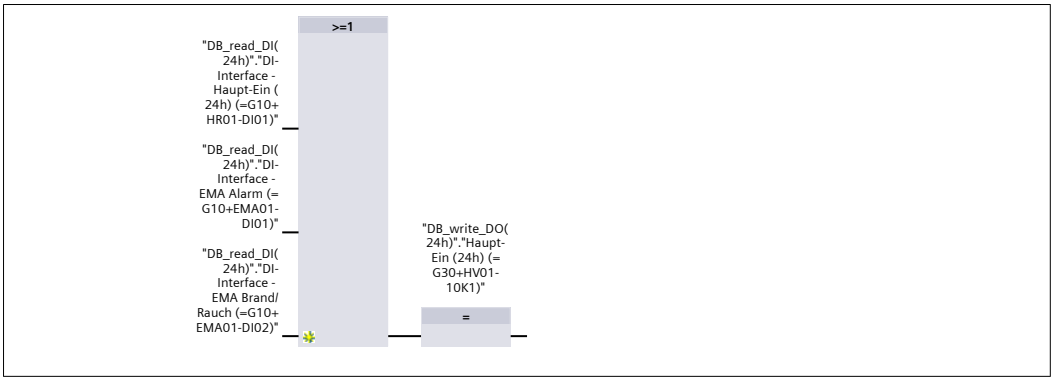
Main Eigenschaften					
Allgemein					
Name	Main	**Nummer**	1	**Typ**	OB
Sprache	FUP	**Nummerierung**	automatisch		
Information					
Titel	"Main Program Sweep (Cycle)"	**Autor**		**Kommentar**	
Familie		**Version**	0.1	**Anwenderdefinierte ID**	

Name	Datentyp	Defaultwert
Input		
Initial_Call	Bool	
Remanence	Bool	
Temp		
Constant		

Netzwerk 1: Watchdog



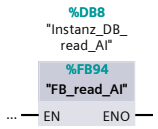
Netzwerk 2: Hauptspannung ein -> 24h Karten Speisen



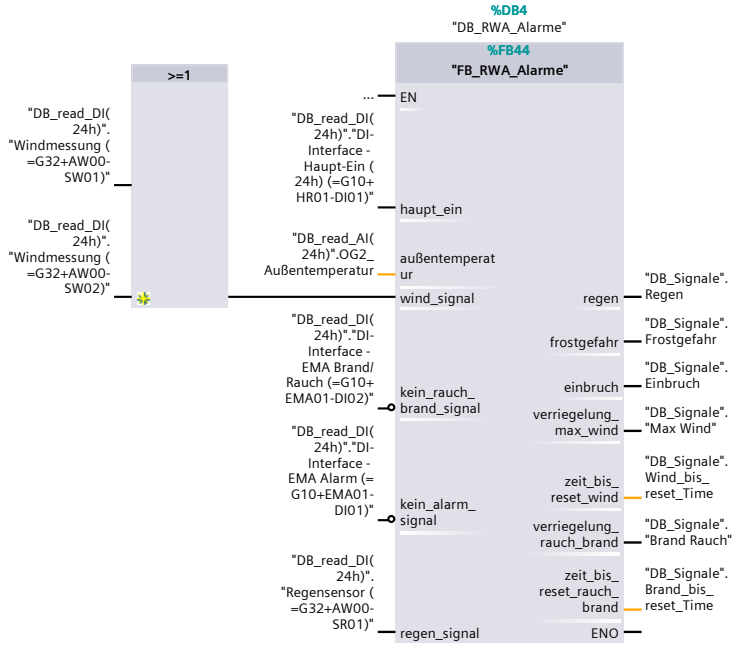
Netzwerk 3: Digitaleingänge lesen

Totally Integrated Automation Portal		

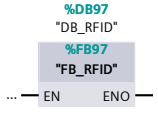
Netzwerk 4: Analogeingänge lesen



Netzwerk 5: Alarmsignale und Rauch- Wärmeabzug (RWA)

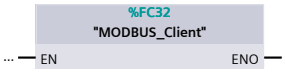


Netzwerk 6: RFID



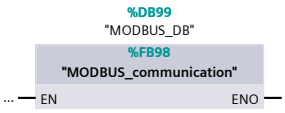
Netzwerk 7: MODBUS Client

Totally Integrated Automation Portal		

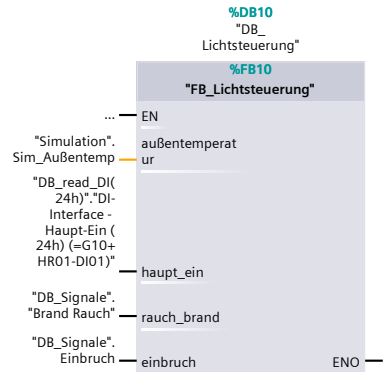


Netzwerk 8: Energie Bilanz berechnen

Netzwerk 9: MODBUS Kommunikation

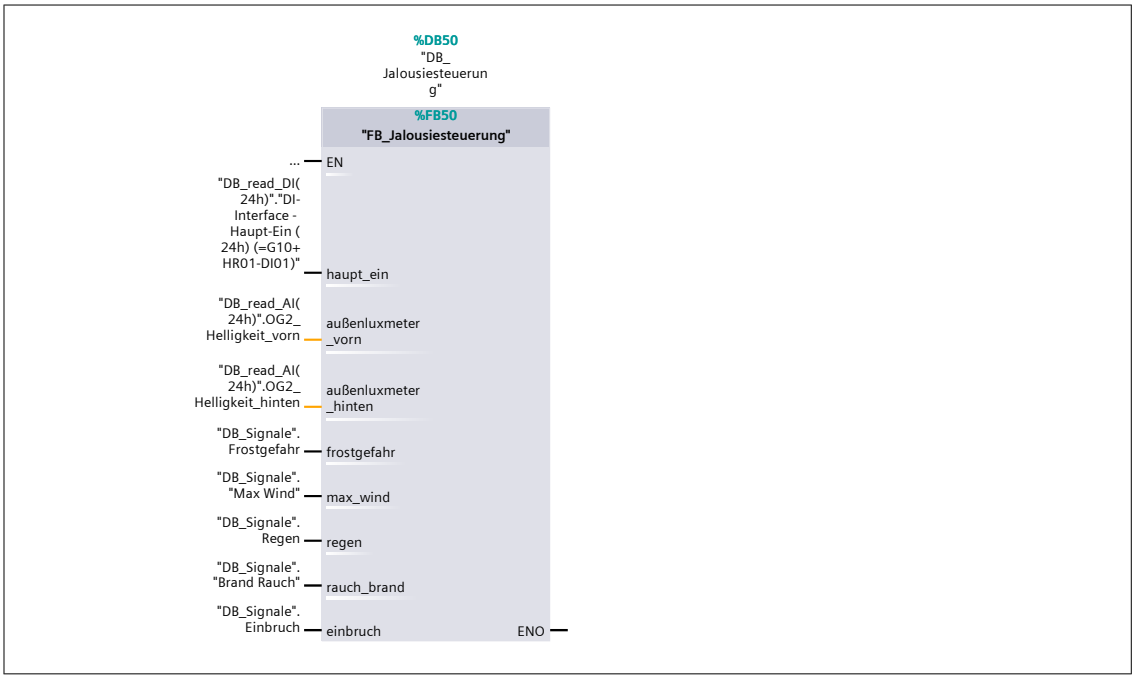


Netzwerk 10: Lichtsteuerung

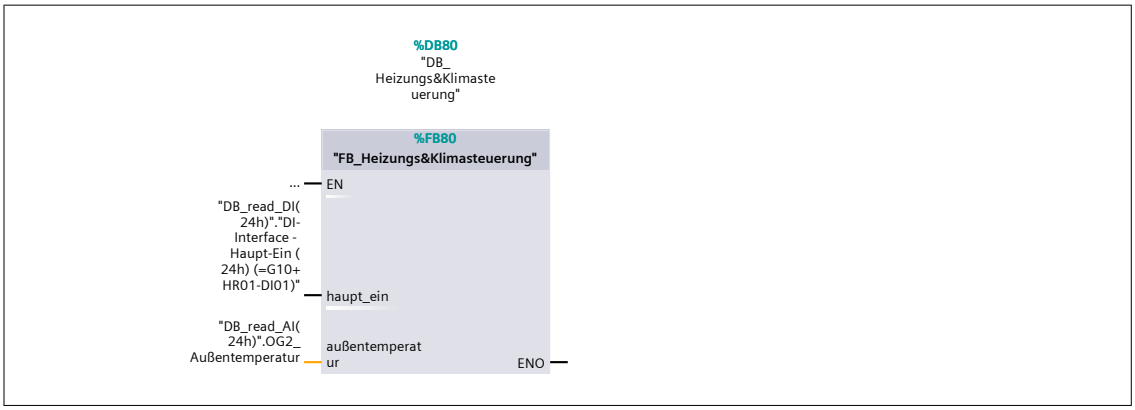


Netzwerk 11: Jalousiesteuerung

Totally Integrated Automation Portal		

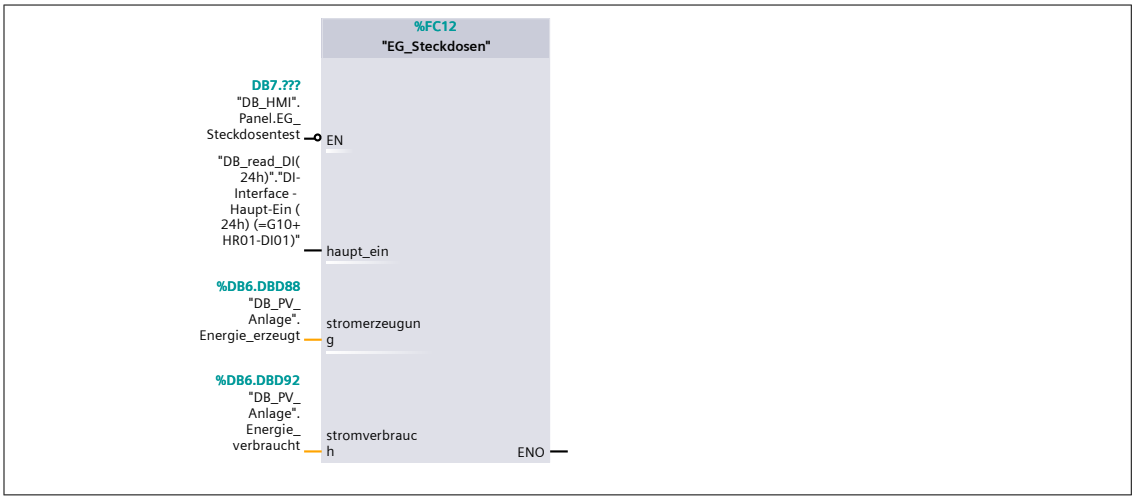


Netzwerk 12: Heizung und Klima

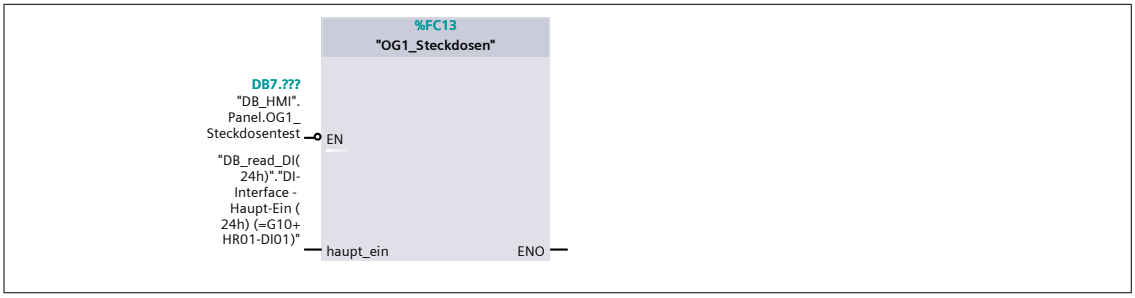


Netzwerk 13: EG Steckdosen

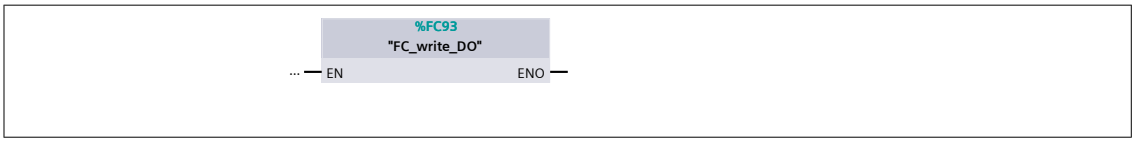
Totally Integrated Automation Portal		



Netzwerk 14: OG1 Steckdosen



Netzwerk 15: Digitalausgänge schreiben



B.2 Speicherfunktion [FB96]

Totally Integrated Automation Portal		

Speicherfunktion [FB96]

Speicherfunktion Eigenschaften					
Allgemein					
Name	Speicherfunktion	**Nummer**	96	**Typ**	FB
Sprache	SCL	**Nummerierung**	manuell		
Information					
Titel	Speicherfunktion	**Autor**	Klann	**Kommentar**	Für diese Funktion wird ein zu speichernder Wert [wert], einen Speicherort in Form eines Arrays [speicherort_array]... ! Achtung: Datentyp der Elemente des Arrays muss gleich dem des Wertes sein! ...und ein Speicherereignis [trigger] zugewiesen. Der Baustein schreibt die Werte an den angegebenen Ort. Die Länge das Arrays kann beliebig sein [0...n]. Ausgaben sind [datentypfehler] und [kopierfehler]. Hinweise auf eventuelle Falscheingaben.
Familie		**Version**	0.1	**Anwenderdefinierte ID**	

Name	Datentyp	Defaultwert	Remanenz
▼ Input			
wert	Variant		
trigger	Bool	false	Nicht remanent
speicherort_array	Variant		
▼ Output			
aktueller_index	Int	0	Nicht remanent
datentypfehler	Bool	false	Nicht remanent
kopierfehler	Int	0	Nicht remanent
InOut			
▼ Static			
index	Int	0	Nicht remanent
trigger_alt	Bool	false	Nicht remanent
▼ Temp			
länge_array	Int		
Constant			

```
0001 (*****************************************************************************
0002 ***************************** Speicherfunktion ******************************
0003 *****************************************************************************)
0004
0005 // Datentyp von Array und vom zu speichernden Wert muss übereinstimmen
0006 (* Natürlich muss das Array auch wirklich ein Array sein ansonsten wird
0007  Datentypfehler ausgegeben *)
0008 IF TypeOf(#wert) = TypeOfElements(#speicherort_array) AND IS_ARRAY(#speicher-
     ort_array) THEN
```


Totally Integrated Automation Portal		

```
0009      #datentypfehler := false;
0010  ELSE
0011      #datentypfehler := true;
0012  END_IF;
0013
0014  // Erkennen der Arraylänge
0015  #länge_array := UDINT_TO_INT(CountOfElements(#speicherort_array) - 1);
0016
0017  (* bei positiver Triggerflanke übergebener #wert in das Array schreiben und
0018   index hochzählen *)
0019  IF NOT #trigger_alt AND #trigger THEN
0020      #kopierfehler := MOVE_BLK_VARIANT(SRC := #wert, COUNT := 1, SRC_INDEX :=
      0, DEST_INDEX := #index, DEST => #speicherort_array);
0021
0022      IF #index < #länge_array THEN
0023          #index := #index + 1;
0024      ELSE
0025          #index := 0;
0026      END_IF;
0027
0028  END_IF;
0029
0030  #trigger_alt := #trigger;
0031  #aktueller_index := #index;
```

B.3 FB_RFID [FB97]

Totally Integrated Automation Portal		

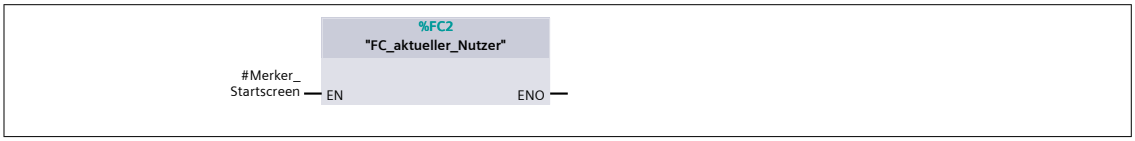
FB_RFID [FB97]

FB_RFID Eigenschaften					
Allgemein					
Name	FB_RFID	**Nummer**	97	**Typ**	FB
Sprache	FUP	**Nummerierung**	manuell		
Information					
Titel	FB_RFID	**Autor**		**Kommentar**	
Familie		**Version**	0.1	**Anwenderdefinierte ID**	

Name	Datentyp	Defaultwert	Remanenz
Input			
Output			
InOut			
▼ Static			
Merker_Startscreen	Bool	false	Nicht remanent
Merker_Lobbyscreen	Bool	false	Nicht remanent
Merker_Fehlerscreen	Bool	false	Nicht remanent
Merker_AB_Screen	Bool	false	Nicht remanent
Merker_Abmeldescreen	Bool	false	Nicht remanent
Merker_A-Bestaetigungsscreen	Bool	false	Nicht remanent
Chip_da	Bool	false	Nicht remanent
▼ Auto_Home_Timer	IEC_TIMER		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
▼ Ausschalverzögerung	IEC_TIMER		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
▼ Auto_Start_Timer	IEC_TIMER		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
Zeit_bis_SC	Time	T#0ms	Nicht remanent
Zeit_bis_SC_DInt	DInt	0	Nicht remanent
Zeit_bis_SC_Anzeige	DInt	0	Nicht remanent
Temp			
Constant			

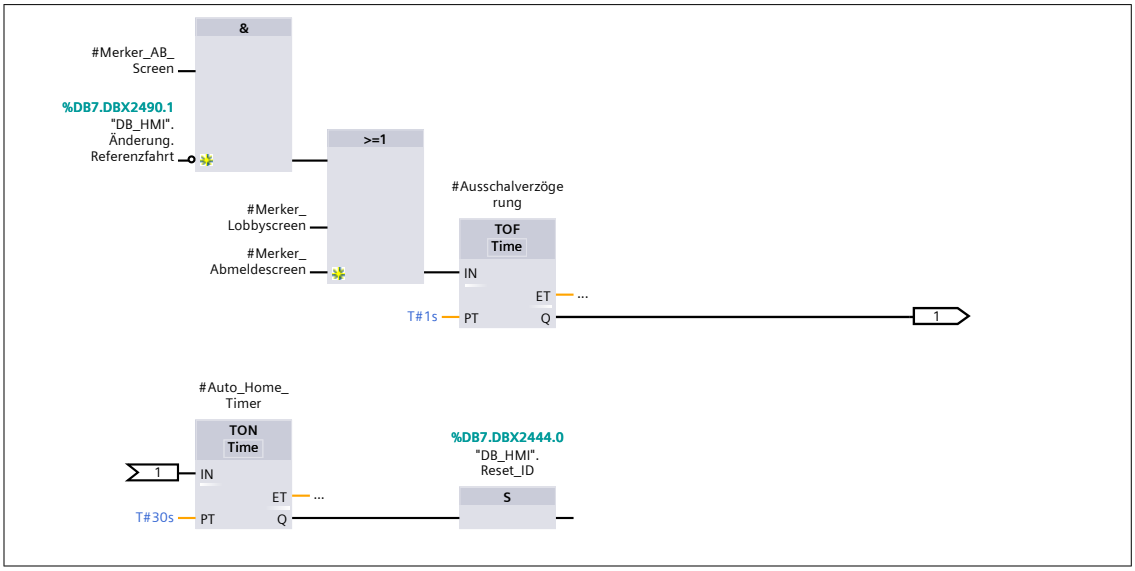
Totally Integrated Automation Portal		

Netzwerk 1: erkennen des aktuellen Nutzers

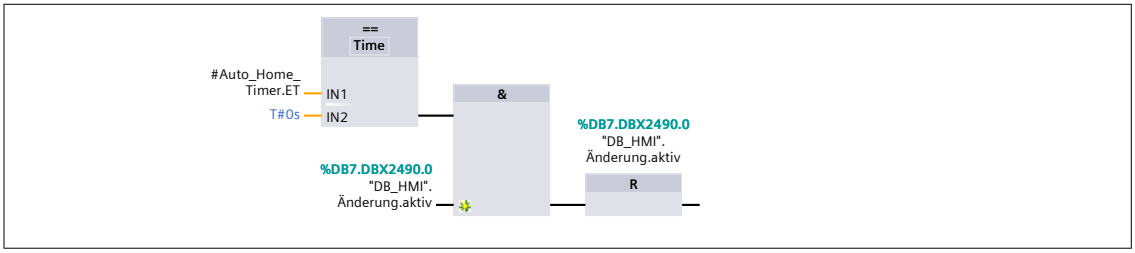


Netzwerk 2: Automatisch das Grundbild aufrufen

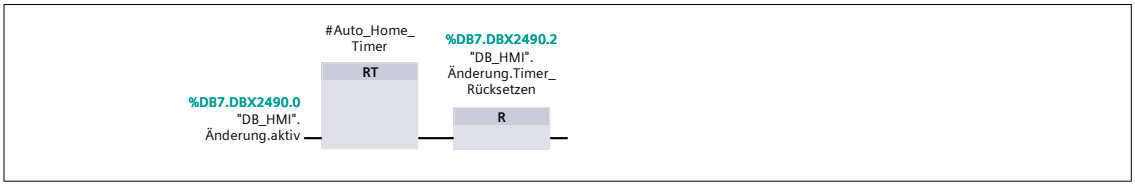
Netzwerk 2: Automatisch das Grundbild aufrufen



Netzwerk 3: für das registrieren einer Nutzer Eingabe -> setzt Auto Timer zurück

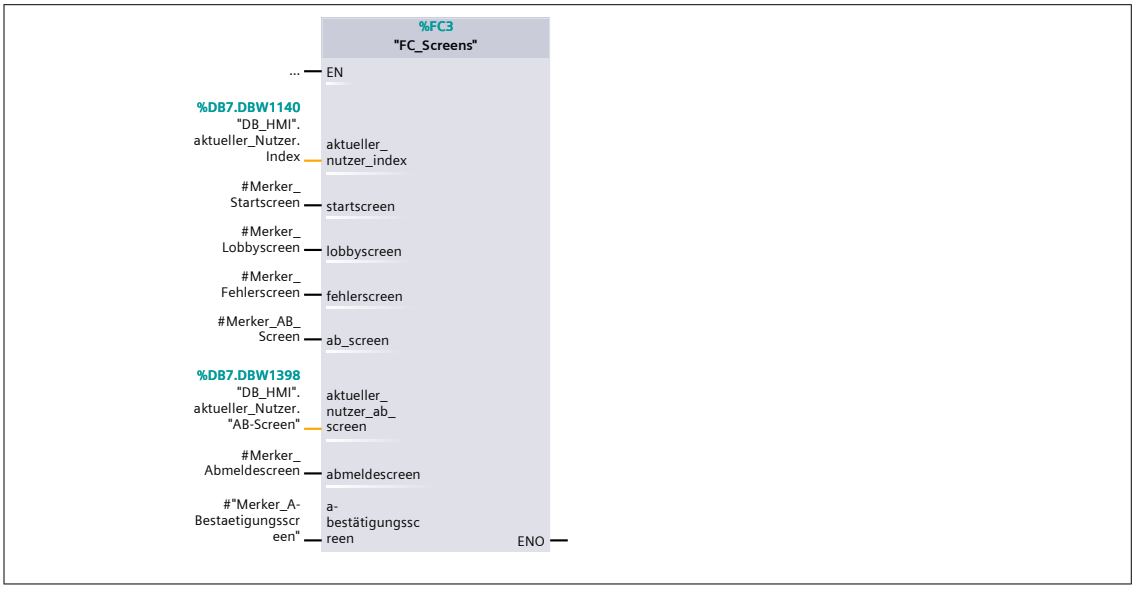


Netzwerk 4: für das registrieren einer Nutzer Eingabe -> setzt Auto Timer zurück

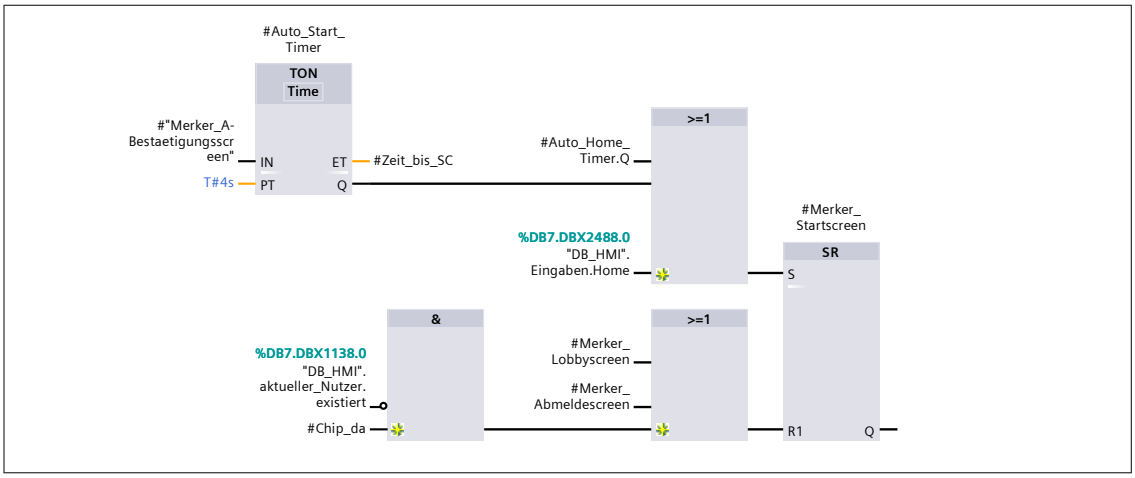


Totally Integrated Automation Portal		

Netzwerk 5: Screen-Auswahl Funktion

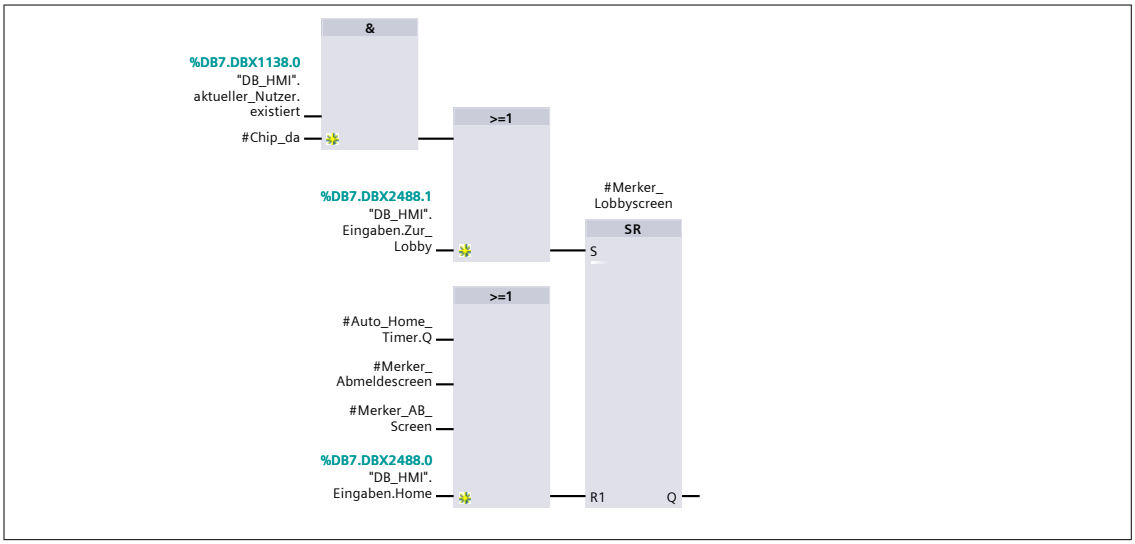


Netzwerk 6: Startscreen-Merker

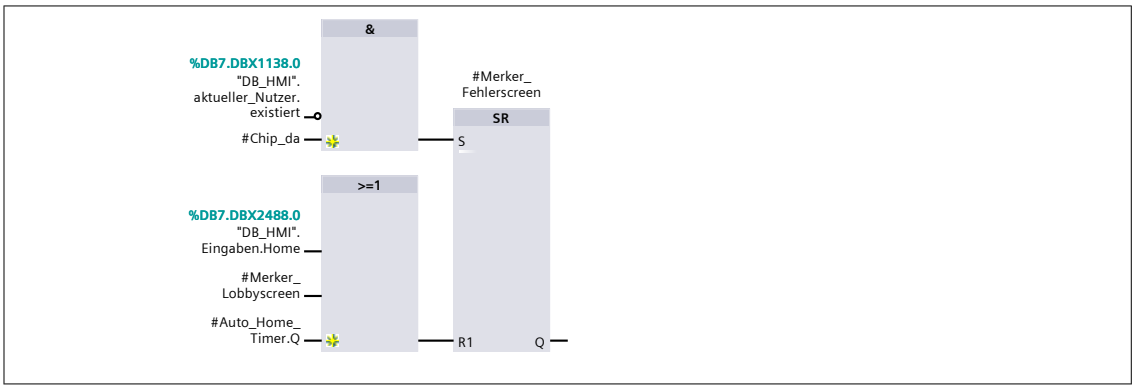


Netzwerk 7: Lobbyscreen-Merker

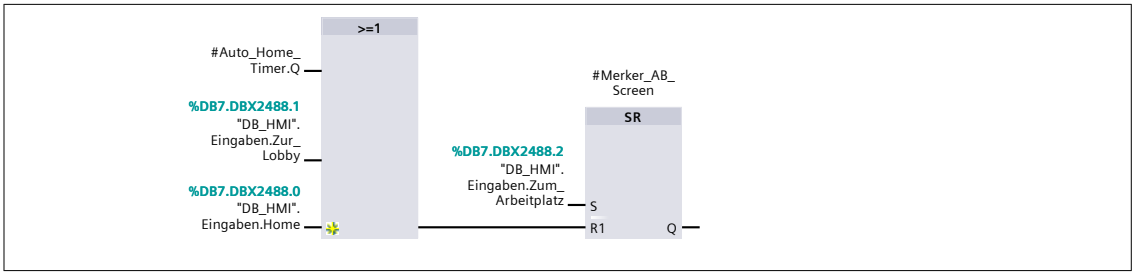
Totally Integrated Automation Portal		



Netzwerk 8: Fehlerscreen-Merker

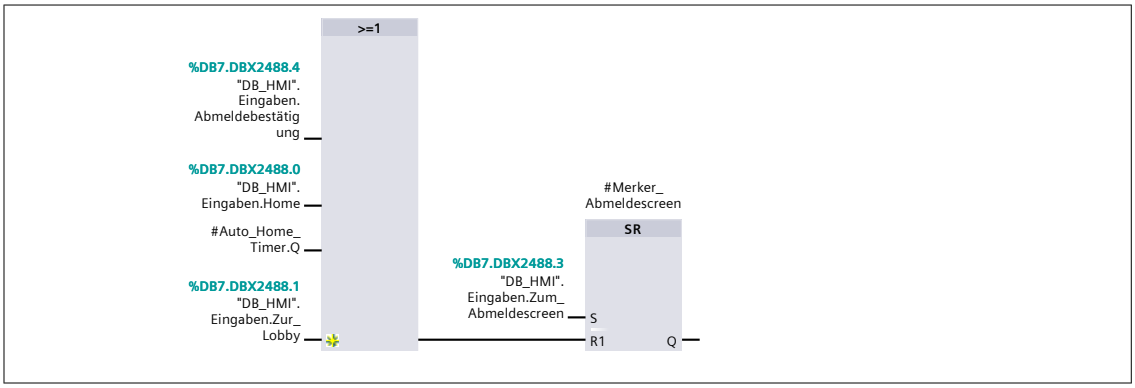


Netzwerk 9: Arbeitsbereichsscreen-Merker

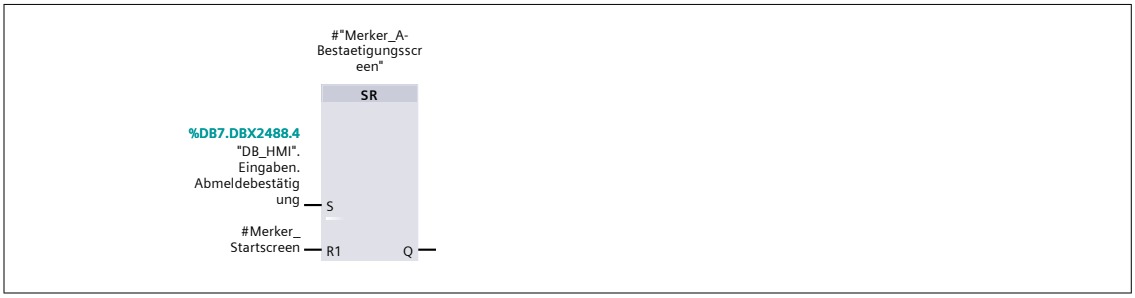


Netzwerk 10: Abmeldescreen-Merker

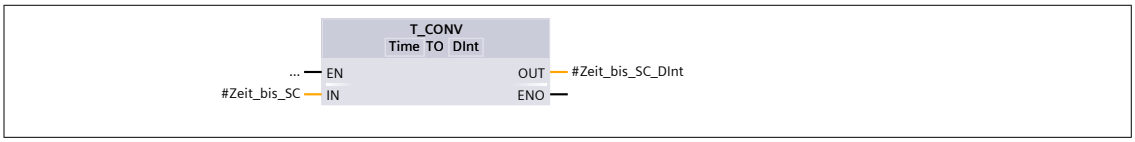
Totally Integrated Automation Portal		



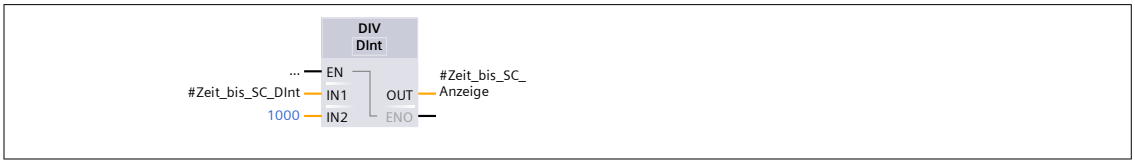
Netzwerk 11: Abmelde-Bestätigungsscreen-Merker



Netzwerk 12: Behandlung der Abmelde-Bestätigungsscreen-Zeit (1/3)

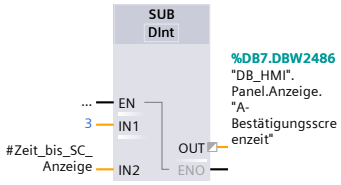


Netzwerk 13: Behandlung der Abmelde-Bestätigungsscreen-Zeit (2/3)



Netzwerk 14: Behandlung der Abmelde-Bestätigungsscreen-Zeit (3/3)

Totally Integrated Automation Portal		



B.4 Positionsberechnung [FB65]

Totally Integrated Automation Portal		

Positionsberechnung [FB65]

Positionsberechnung Eigenschaften					
Allgemein					
Name	Positionsberechnung	**Nummer**	65	**Typ**	FB
Sprache	SCL	**Nummerierung**	automatisch		
Information					
Titel		**Autor**		**Kommentar**	
Familie		**Version**	0.1	**Anwenderdefinierte ID**	

Name	Datentyp	Defaultwert	Remanenz
▼ Input			
motor_hoch	Bool	false	Nicht remanent
motor_runter	Bool	false	Nicht remanent
max_auffahrzeit	Time	T#0ms	Nicht remanent
max_abfahrzeit	Time	T#0ms	Nicht remanent
stoppsignal	Bool	false	Nicht remanent
▼ Output			
position_%	Int	0	Nicht remanent
auffahrverriegelung	Bool	false	Nicht remanent
abffahrverriegelung	Bool	false	Nicht remanent
letzte_auffahrzeit	Time	T#0ms	Nicht remanent
letzte_abfahrzeit	Time	T#0ms	Nicht remanent
▼ InOut			
reset	Bool	false	Nicht remanent
▼ speicher_auffahrzeit	Array[0..10] of Time		
speicher_auffahrzeit[0]	Time		
speicher_auffahrzeit[1]	Time		
speicher_auffahrzeit[2]	Time		
speicher_auffahrzeit[3]	Time		
speicher_auffahrzeit[4]	Time		
speicher_auffahrzeit[5]	Time		
speicher_auffahrzeit[6]	Time		
speicher_auffahrzeit[7]	Time		
speicher_auffahrzeit[8]	Time		
speicher_auffahrzeit[9]	Time		
speicher_auffahrzeit[10]	Time		
▼ speicher_abfahrzeit	Array[0..10] of Time		
speicher_abfahrzeit[0]	Time		
speicher_abfahrzeit[1]	Time		
speicher_abfahrzeit[2]	Time		
speicher_abfahrzeit[3]	Time		
speicher_abfahrzeit[4]	Time		
speicher_abfahrzeit[5]	Time		
speicher_abfahrzeit[6]	Time		
speicher_abfahrzeit[7]	Time		
speicher_abfahrzeit[8]	Time		
speicher_abfahrzeit[9]	Time		
speicher_abfahrzeit[10]	Time		

Totally Integrated Automation Portal		

Name	Datentyp	Defaultwert	Remanenz
speicher_positionswerte	Array[0..10] of Real		
speicher_positionswerte[0]	Real		
speicher_positionswerte[1]	Real		
speicher_positionswerte[2]	Real		
speicher_positionswerte[3]	Real		
speicher_positionswerte[4]	Real		
speicher_positionswerte[5]	Real		
speicher_positionswerte[6]	Real		
speicher_positionswerte[7]	Real		
speicher_positionswerte[8]	Real		
speicher_positionswerte[9]	Real		
speicher_positionswerte[10]	Real		
Static			
i	Int	0	Nicht remanent
j	Int	0	Nicht remanent
i_minus_1	Int	0	Nicht remanent
obere_endlage	Bool	false	Nicht remanent
untere_endlage	Bool	false	Nicht remanent
Fahrrichtung	Bool	false	Nicht remanent
oben_verzögerung	TON_TIME		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
unten_verzögerung	TON_TIME		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
auffahrtimer	TON_TIME		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
abfahrtimer	TON_TIME		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
ausschalt	TON_TIME		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent

Totally Integrated Automation Portal		

Name	Datentyp	Defaultwert	Remanenz
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
auffahrzeit	Time	T#0ms	Nicht remanent
abfahrzeit	Time	T#0ms	Nicht remanent
▼ Bearbeitung	Struct		Nicht remanent
normiert_auf	Real	0.0	Nicht remanent
normiert_ab	Real	0.0	Nicht remanent
skaliert_auf	Int	0	Nicht remanent
skaliert_ab	Int	0	Nicht remanent
normierte_Position	Real	0.0	Nicht remanent
Position	Int	0	Nicht remanent
▼ auffahrzeiten_speichern	"Speicherfunktion"		
▼ Input			
wert	Variant		
trigger	Bool	false	Nicht remanent
speicherort_array	Variant		
▼ Output			
aktueller_index	Int	0	Nicht remanent
datentypfehler	Bool	false	Nicht remanent
kopierfehler	Int	0	Nicht remanent
InOut			
▼ Static			
index	Int	0	Nicht remanent
trigger_alt	Bool	false	Nicht remanent
▼ abfahrzeiten_speichern	"Speicherfunktion"		
▼ Input			
wert	Variant		
trigger	Bool	false	Nicht remanent
speicherort_array	Variant		
▼ Output			
aktueller_index	Int	0	Nicht remanent
datentypfehler	Bool	false	Nicht remanent
kopierfehler	Int	0	Nicht remanent
InOut			
▼ Static			
index	Int	0	Nicht remanent
trigger_alt	Bool	false	Nicht remanent
▼ positionswerte_speichern	"Speicherfunktion"		
▼ Input			
wert	Variant		
trigger	Bool	false	Nicht remanent
speicherort_array	Variant		
▼ Output			
aktueller_index	Int	0	Nicht remanent
datentypfehler	Bool	false	Nicht remanent
kopierfehler	Int	0	Nicht remanent
InOut			

Totally Integrated Automation Portal		

Name	Datentyp	Defaultwert	Remanenz
Static			
index	Int	0	Nicht remanent
trigger_alt	Bool	false	Nicht remanent
positionswerte_rücksetzen	"Speicherfunktion"		
Input			
wert	Variant		
trigger	Bool	false	Nicht remanent
speicherort_array	Variant		
Output			
aktueller_index	Int	0	Nicht remanent
datentypfehler	Bool	false	Nicht remanent
kopierfehler	Int	0	Nicht remanent
InOut			
Static			
index	Int	0	Nicht remanent
trigger_alt	Bool	false	Nicht remanent
Temp			
Constant			

```
0001 (*****************************************************************************
0002 ************************** Berechnung der Jalousieposition *****************
0003 *****************************************************************************)
0004
0005 // Index für den letzten Wert im Array
0006 IF #i = 0 THEN
0007     #i_minus_1 := 10;
0008 ELSE
0009     #i_minus_1 := #i - 1;
0010 END_IF;
0011
0012 // Werte speichern
0013 #auffahrzeiten_speichern(wert := #auffahrzeit,
0014                          trigger := #stoppsignal,
0015                          speicherort_array := #speicher_auffahrzeit,
0016                          aktueller_index => #i);
0017
0018 #abfahrzeiten_speichern(wert := #abfahrzeit,
0019                         trigger := #stoppsignal,
0020                         speicherort_array := #speicher_abfahrzeit,
0021                         aktueller_index => #i);
0022
0023 //Fahrtimer
0024 #auffahrtimer(IN := #motor_hoch,
0025               PT := #max_auffahrzeit,
0026               ET => #auffahrzeit);
0027
0028 #abfahrtimer(IN := #motor_runter,
0029              PT := #max_abfahrzeit,
0030              ET => #abfahrzeit);
0031
0032 // Zur Anzeige der letzten aktiven Fahrzeit
0033 IF #stoppsignal AND (#auffahrzeit = T#0s OR #abfahrzeit = T#0s) THEN
0034     #letzte_auffahrzeit := #speicher_auffahrzeit[#i_minus_1];
0035     #letzte_abfahrzeit := #speicher_abfahrzeit[#i_minus_1];
0036 END_IF;
```

Totally Integrated Automation Portal		

```
0037
0038 #Bearbeitung.normiert_auf := NORM_X(MIN := 0, VALUE := #auffahrzeit, MAX :=
     TIME_TO_DINT(#max_auffahrzeit));
0039 #Bearbeitung.normiert_ab := NORM_X(MIN := 0, VALUE := #abfahrzeit, MAX :=
     TIME_TO_DINT(#max_abfahrzeit));
0040
0041 // Wertberechnung der normierten Werte
0042 IF #Bearbeitung.normiert_auf <> 0.0 THEN
0043     #Bearbeitung.normierte_Position := #speicher_positionswerte[#i_minus_1] -
     #Bearbeitung.normiert_auf;
0044 END_IF;
0045
0046 IF #Bearbeitung.normiert_ab <> 0.0 THEN
0047     #Bearbeitung.normierte_Position := #speicher_positionswerte[#i_minus_1] +
     #Bearbeitung.normiert_ab;
0048 END_IF;
0049
0050 #positionswerte_speichern(wert := #Bearbeitung.normierte_Position,
0051                           trigger := #stoppsignal,
0052                           speicherort_array := #speicher_positionswerte,
0053                           aktueller_index => #i);
0054
0055 // Anzeige begrenzen
0056 IF #Bearbeitung.normierte_Position >= 1.0 THEN
0057     #Bearbeitung.normierte_Position := 1.0;
0058     #obere_endlage := false;
0059     #untere_endlage := true;
0060 ELSIF #Bearbeitung.normierte_Position <= 0.0 THEN
0061     #Bearbeitung.normierte_Position := 0.0;
0062     #obere_endlage := true;
0063     #untere_endlage := false;
0064 ELSE
0065     #obere_endlage := false;
0066     #untere_endlage := false;
0067     #Bearbeitung.Position := SCALE_X(MIN := 0, VALUE := #Bearbeitung.nor-
     mierte_Position, MAX := 100);
0068 END_IF;
0069
0070 #oben_verzögerung(IN := #obere_endlage,
0071                   PT := T#2s,
0072                   Q => #auffahrverriegelung);
0073
0074 #unten_verzögerung(IN := #untere_endlage,
0075                    PT := T#2s,
0076                    Q => #abffahrverriegelung);
0077
0078 // Reset der Anzeige
0079 IF #reset AND #Bearbeitung.normierte_Position <> 0.0 THEN
0080     #Bearbeitung.normierte_Position := 0;
0081     FOR #j := 0 TO 10 DO
0082         #speicher_positionswerte[#j] := 0;
0083     END_FOR;
0084     #reset := false;
0085 END_IF;
0086
0087 // Position Ausgeben
0088 #"position_%" := #Bearbeitung.Position;
0089
0090
```

B.5 Referenzfahrt [FB66]

Totally Integrated Automation Portal		

Referenzfahrt [FB66]

Referenzfahrt Eigenschaften					
Allgemein					
Name	Referenzfahrt	**Nummer**	66	**Typ**	FB
Sprache	SCL	**Nummerierung**	automatisch		
Information					
Titel		**Autor**		**Kommentar**	
Familie		**Version**	0.1	**Anwenderdefinierte ID**	

Name	Datentyp	Defaultwert	Remanenz
Input			
▼ Output			
auffahren	Bool	false	Nicht remanent
abfahren	Bool	false	Nicht remanent
▼ InOut			
referenzfahrt	Bool	false	Nicht remanent
bild	Int	0	Nicht remanent
stopp	Bool	false	Nicht remanent
start	Bool	false	Nicht remanent
neustart	Bool	false	Nicht remanent
fehler	Bool	false	Nicht remanent
gerichtet	Bool	false	Nicht remanent
verwerfen	Bool	false	Nicht remanent
abbrechen	Bool	false	Nicht remanent
max_auffahrzeit	Time	T#0ms	Nicht remanent
max_abfahrzeit	Time	T#0ms	Nicht remanent
▼ speicher	Array[0..1] of Time		
speicher[0]	Time		
speicher[1]	Time		
▼ Static			
▼ Timer_Richten	TON_TIME		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
▼ Timer_max_Abfahrzeit	TON_TIME		Nicht remanent
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
neue_max_Abfahrzeit	Time	T#0ms	Nicht remanent
max_Abfahrzeit_speicherort	Int	0	Nicht remanent
abfahren_fertig	Bool	false	Nicht remanent
▼ Timer_max_Auffahrzeit	TON_TIME		Nicht remanent

Totally Integrated Automation Portal

Name	Datentyp	Defaultwert	Remanenz
ST	Time	T#0ms	Nicht remanent
PT	Time	T#0ms	Nicht remanent
ET	Time	T#0ms	Nicht remanent
RU	Bool	false	Nicht remanent
IN	Bool	false	Nicht remanent
Q	Bool	false	Nicht remanent
neue_max_Auffahrzeit	Time	T#0ms	Nicht remanent
max_Auffahrzeit_speicherort	Int	0	Nicht remanent
auffahren_fertig	Bool	false	Nicht remanent
motor_hoch	Bool	false	Nicht remanent
motor_runter	Bool	false	Nicht remanent
Speicherfunktion	"Speicherfunktion"		
Input			
wert	Variant		
trigger	Bool	false	Nicht remanent
speicherort_array	Variant		
Output			
aktueller_index	Int	0	Nicht remanent
datentypfehler	Bool	false	Nicht remanent
kopierfehler	Int	0	Nicht remanent
InOut			
Static			
index	Int	0	Nicht remanent
trigger_alt	Bool	false	Nicht remanent
Temp			
Constant			

```
0001 (*****************************************************************************
0002 ****************************** Referenzfahrt ********************************
0003 *****************************************************************************)
0004
0005 (* Init *)
0006 IF #bild = 0 THEN
0007     #stopp := false;
0008     #motor_hoch := false;
0009     #auffahren := false;
0010     #motor_runter := false;
0011     #abfahren := false;
0012     #abfahren_fertig := false;
0013     #auffahren_fertig := false;
0014 END_IF;
0015
0016
0017 (* Abbruch *)
0018 IF #abbrechen THEN
0019     #motor_hoch := false;
0020     #auffahren := false;
0021     #motor_runter := false;
0022     #abfahren := false;
0023     #gerichtet := false;
0024     #neustart := false;
0025     #bild := 0;
0026
0027     #abbrechen := false;
0028
```

Totally Integrated Automation Portal		

```
0029      #referenzfahrt := false; (*Zuweisung MUSS an LETZTE Stelle der IF-Anwei-
      sung!!!
0030       führt zur Bausteinbeendigung*)
0031  END_IF;
0032
0033  (* Fehler (Timer Ablauf) *)
0034  IF #fehler THEN
0035      #start := false;
0036      #gerichtet := false;
0037      #neustart := false;
0038      #bild := 8;
0039
0040      #fehler := false; //zur Einmaligen Ausführung der IF-Abfrage
0041  END_IF;
0042
0043  (* Neustart *)
0044  IF #neustart THEN
0045      #start := false;
0046      #gerichtet := false;
0047      #fehler := false;
0048      #bild := 1;
0049
0050      #neustart := false; //zur Einmaligen Ausführung der IF-Abfrage
0051  END_IF;
0052  (*****************************************************)
0053
0054  IF #referenzfahrt AND #bild = 0 THEN
0055      #bild := 1;
0056  END_IF;
0057
0058  (*****************************************************)
0059  (* Richtvorgang *)
0060  IF #start AND NOT #gerichtet AND NOT #fehler THEN
0061
0062      #bild := 2;
0063
0064      #Timer_Richten(IN := #motor_hoch,
0065                     PT := T#2m,
0066                     Q => #fehler);
0067
0068      IF #stopp THEN
0069          #motor_hoch := false;
0070          #gerichtet := true;
0071          #start := false;
0072          #stopp := false;
0073      ELSE
0074          #motor_hoch := true;
0075      END_IF;
0076
0077      #auffahren := #motor_hoch;
0078
0079  END_IF;
0080  (*****************************************************)
0081
0082  IF #gerichtet AND (#bild = 1 OR #bild = 2) THEN
0083      #bild := 3;
0084  END_IF;
0085
0086  (*****************************************************)
```

Totally Integrated Automation Portal		

```
0087 (* neue Abfahrzeit bestimmen *)
0088 IF #start AND NOT #abfahren_fertig AND #gerichtet AND NOT #fehler THEN
0089
0090     #bild := 4;
0091
0092     #Timer_max_Abfahrzeit(IN := #motor_runter,
0093                           PT := T#2m,
0094                           Q => #fehler,
0095                           ET => #neue_max_Abfahrzeit);
0096
0097     #Speicherfunktion(wert := #neue_max_Abfahrzeit,
0098                       trigger := #stopp,
0099                       speicherort_array := #speicher,
0100                       aktueller_index => #max_Auffahrzeit_speicherort); //
     gibt immer den nächsten index aus
0101
0102     IF #stopp THEN
0103         #motor_runter := false;
0104         #abfahren_fertig := true;
0105         #start := false;
0106         #stopp := false;
0107         #bild := 5;
0108     ELSE
0109         #motor_runter := true;
0110     END_IF;
0111
0112     #abfahren := #motor_runter;
0113
0114 END_IF;
0115 (*****************************************************)
0116
0117 (*****************************************************)
0118 (* neue Auffahrzeit bestimmen *)
0119 IF #start AND NOT #auffahren_fertig AND #abfahren_fertig THEN
0120
0121     #bild := 6;
0122
0123     #Timer_max_Abfahrzeit(IN := #motor_hoch,
0124                           PT := T#2m,
0125                           Q => #fehler,
0126                           ET => #neue_max_Auffahrzeit);
0127
0128     #Speicherfunktion(wert := #neue_max_Auffahrzeit,
0129                       trigger := #stopp,
0130                       speicherort_array := #speicher,
0131                       aktueller_index => #max_Abfahrzeit_speicherort); //gibt
     immer den nächsten index aus
0132
0133     IF #stopp  THEN
0134         #motor_hoch := false;
0135         #auffahren_fertig := true;
0136         #start := false;
0137         #stopp := false;
0138         #bild := 7;
0139     ELSE
0140         #motor_hoch := true;
0141     END_IF;
0142
0143     #auffahren := #motor_hoch;
```

Totally Integrated Automation Portal		

```
0144
0145 END_IF;
0146 (*****************************************************)
0147
0148 (*****************************************************)
0149 (* Zeiten schreiben... *)
0150 IF #start AND #abfahren_fertig AND #auffahren_fertig THEN
0151     #max_auffahrzeit := #speicher[#max_Auffahrzeit_speicherort];
0152     #max_abfahrzeit := #speicher[#max_Abfahrzeit_speicherort];
0153     #start := false;
0154     #bild := 0;
0155     #referenzfahrt := false;
0156
0157 (* ... oder verwerfen *)
0158 ELSIF #verwerfen AND #abfahren_fertig AND #auffahren_fertig THEN
0159     #verwerfen := false;
0160     #bild := 0;
0161     #referenzfahrt := false;
0162 END_IF;
0163
0164 (*****************************************************)
```