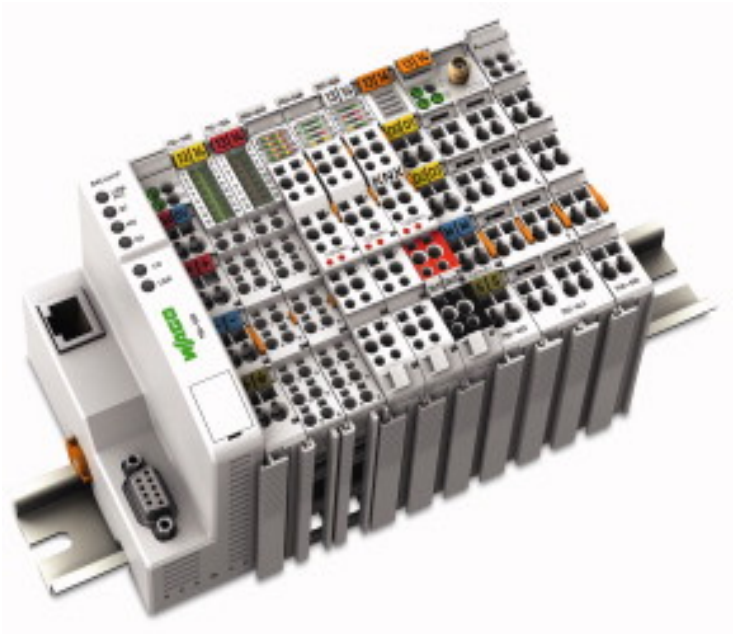
Hochschule **Rosenheim**
University of Applied Sciences

WAGO (mit CoDeSys V2.x)



Tutorial für das Praktikum „Gebäudeautomation“

Version 02, 19. Juni 2018

Hochschule Rosenheim · Hochschulstrasse 1 · 83024 Rosenheim
www.fh-rosenhheim.de · *michael.kroedel@fh-rosenheim.de*

Tutorial: WAGO

Inhaltsverzeichnis

Tutorial: WAGO

Dieses Tutorial wurde an der Hochschule Rosenheim im Rahmen von Projektarbeiten unter Leitung von Herrn Prof. Dr. Michael Krödel erstellt.

Mitgewirkt haben:

- Simon Hartmann, Deni Hemen (Ersterstellung 2014)
- Stephan Bröcker, Michael Höß (Überarbeitung 2015)

Tutorial: WAGO

1 Allgemeine Infos zu den Praktikumsteilen WAGO & WAGO-KNX/MBus

Praktikumsteil WAGO

Für diesen Praktikumsteil wird ein WAGO-Controller in Betrieb genommen.
Dabei sollte vorbereitend möglichst das ganze Tutorial durchgelesen werden, um beim eigentlichen Termin möglichst viele Funktionen testen zu können.

Einige der Kapitel des Tutorials kommen im praktischen Teil nicht vor, diese sind dann extra markiert („nur INFO“ in den Überschriften). Trotzdem sollten auch diese gelesen werden um ein besseres Verständnis für die Arbeit mit einem Controller zu bekommen

Praktikumsteil WAGO-KNX

Für diesen Praktikumsteil wird ein WAGO-Controller in Betrieb genommen und mit einem bestehenden KNX-Netzwerk gekoppelt.

Das Tutorial wurde entsprechend auf das Praktikum zugeschnitten, d.h. es wurden einige Teile entfernt und etwaige Zusatzinformationen und Anmerkungen wurden farblich abgehoben hinzugefügt.

Die Verknüpfungen zu den benötigten Programmen befinden sich Falle auf dem Desktop. Hier finden sich auch alle benötigten Projektdateien.

Der hier benutzte Lenovo Laptop hat die IP-Adresse **192.168.0.10.** Dies sollte unter *Systemsteuerung/Netzwerk und Internet/Netzwerkverbindungen/LAN-Verbindung /TCP-IPv4* kontrolliert werden.

2 HW-Aufbau (nur INFO)

Grundsätzlich sollte das entsprechende Handbuch des verwendeten Controllers zu Aufbau, Handhabung und Anschluss beachtet werden.

Abbildung 1: Netzteil und Controller (Quelle: WAGO)

Das WAGO-System ist modular aufgebaut und besteht aus dem Controller sowie einer beliebigen Anzahl an (Erweiterungs-)Klemmen. Als letzte Klemme ist eine Endklemme (750-600) einzusetzen.

Zur Spannungsversorgung wird ein 24V-DC-Netzteil benötigt. Die Versorgungsspannung für den Controller ist am Controller an die Kontakte „24V“, bzw. „0V“ anzuschließen. Die Spannungsversorgung für die Klemmen ist am Controller an die Klemmen „+“ und„-“ anzuschließen. Falls eine einzige Spannungsversorgung für Controller und Klemmen verwendet wird, ist zwischen den Klemmen „24V“ und „+“ sowie zwischen „0V“ und „-“ eine Leiterbrücke zu installieren!

Tutorial: WAGO

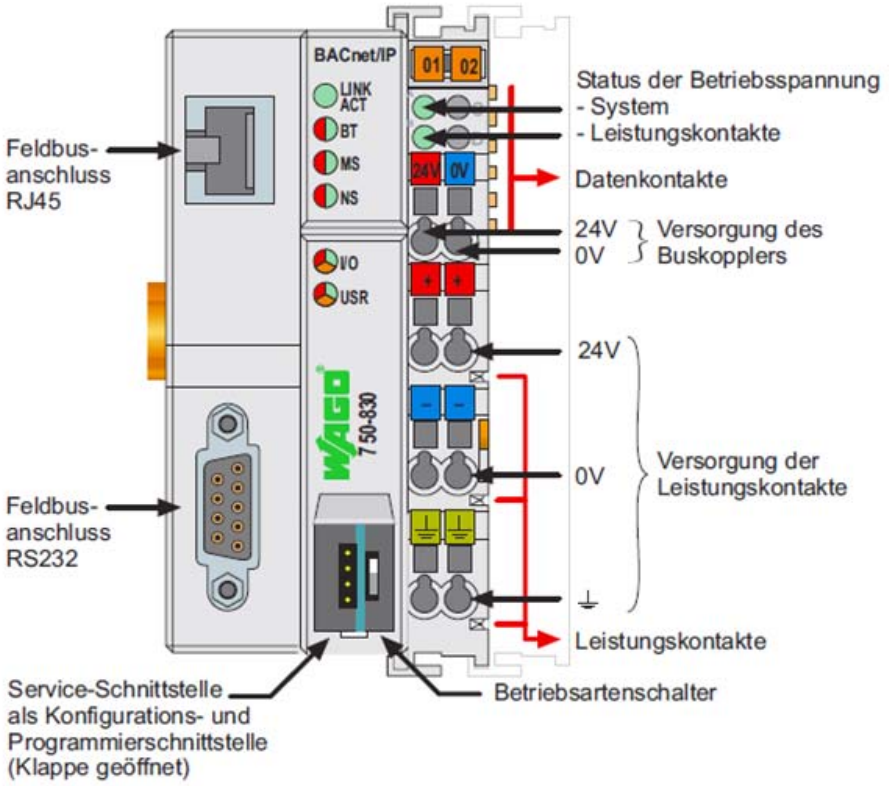


Abbildung 2: BACnet Controller (Quelle: WAGO)

Tutorial: WAGO

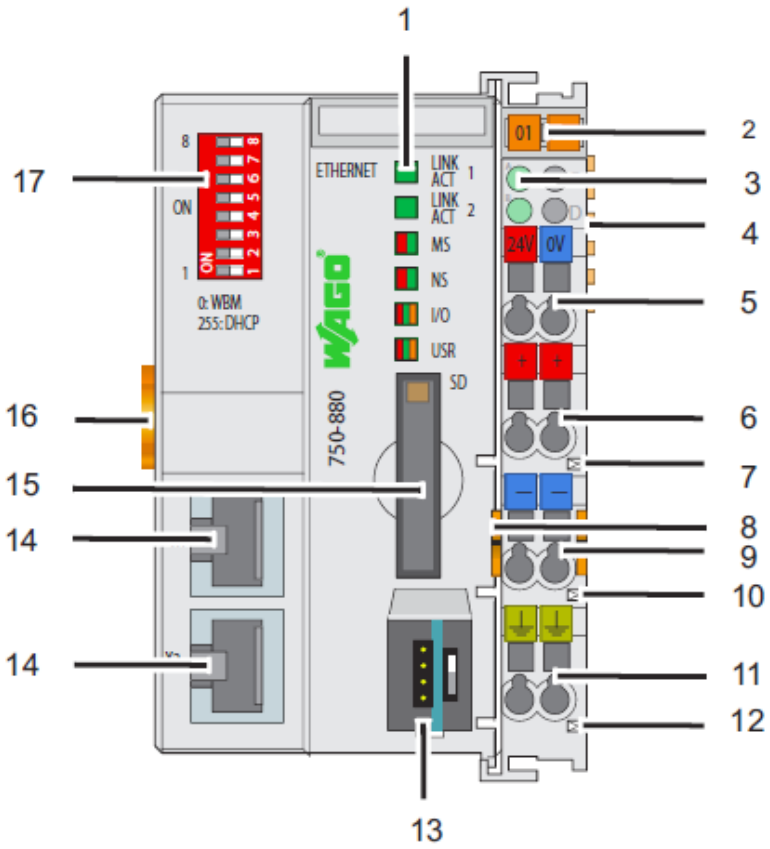


Abbildung 3: Ethernet-Controller (Quelle: WAGO)

Der Betriebsartenschalter (unter der Klappe) muss in der obersten Position stehen (Modus RUN). Ein Drücken des Betriebsartenschalters nach unten bewirkt einen Reset.

Beim Ethernet-Controller (750-880) müssen die DIP-Schalter alle auf OFF stehen (entspricht der Adresse „0“). Das bedeutet in Konsequenz, dass der Controller seine IP-Adresse softwaremäßig zugewiesen bekommt.

Die Klemme „Erde/Schutzerde“ des Controllers wird auf die Schutzerde der 230V-Versorgung gelegt. Zusätzlich ist die Tragschiene/Hutschiene zu erden (z.B. über Klemme 283-609). Der Erdungsleiter sollte einen Querschnitt von mind. 4 mm² haben.

3 Netzwerkkonfiguration des Controllers

Zum grundsätzlichen Setup gehört das Konfigurieren des Knotens inkl. Netzwerkkonfiguration. Diese kann grundsätzlich über zwei unterschiedliche Arten durchgeführt werden:

- USB-Service-Kabel (empfohlen!)
- BootP

3.1 IP-Adresse über USB Service Kabel

Die Adresszuweisung über das USB Servicekabel ist die Standardvariante. Der Controller ist über das USB-Service-Kabel mit einem PC zu verbinden (am Controller auf die Programmierschnittstelle unter der Klappe stecken).
Das Aufstecken oder Abziehen des Servicekabels muss im spannungsfreien Zustand des Controllers durchgeführt werden!

Anschließend ist das Programm „Ethernet Settings“ zu starten. Im Programm ist zunächst der COM-Port anzugeben, der bei der Installation des Treibers für das USB-Kabel angelegt wurde.

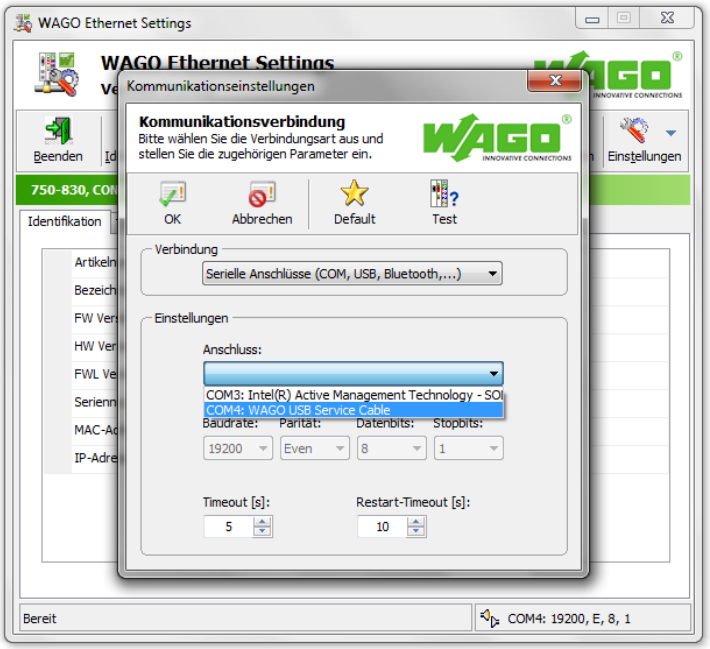


Abbildung 4: Verbindung zum Controller über das Servicekabel

Dann ist die Schaltfläche „Identifizieren“ anzuwählen. Die Daten des Controllers werden ausgelesen und im Reiter „Netzwerk“ erscheint die IP-Konfiguration.

Hinweis: Der Betriebsartenschalter am Controller (unter der kleinen Plastiklappe) muss auf RUN, d.h. der obersten Position stehen.

Dann kann eine fixe IP-Adresse angegeben werden (Achtung: diese muss zu dem Netzwerk passen) oder die Vergabe über DHCP ausgewählt werden. Über die Anwahl von „Schreiben“ wird diese in den Controller geschrieben. Anschließend erscheint die vergebene Adresse im Feld „Aktuelle IP-Adresse“. Die IP-Adresse ist auf **192.168.0.100** einzustellen! (Aufgabe 1)

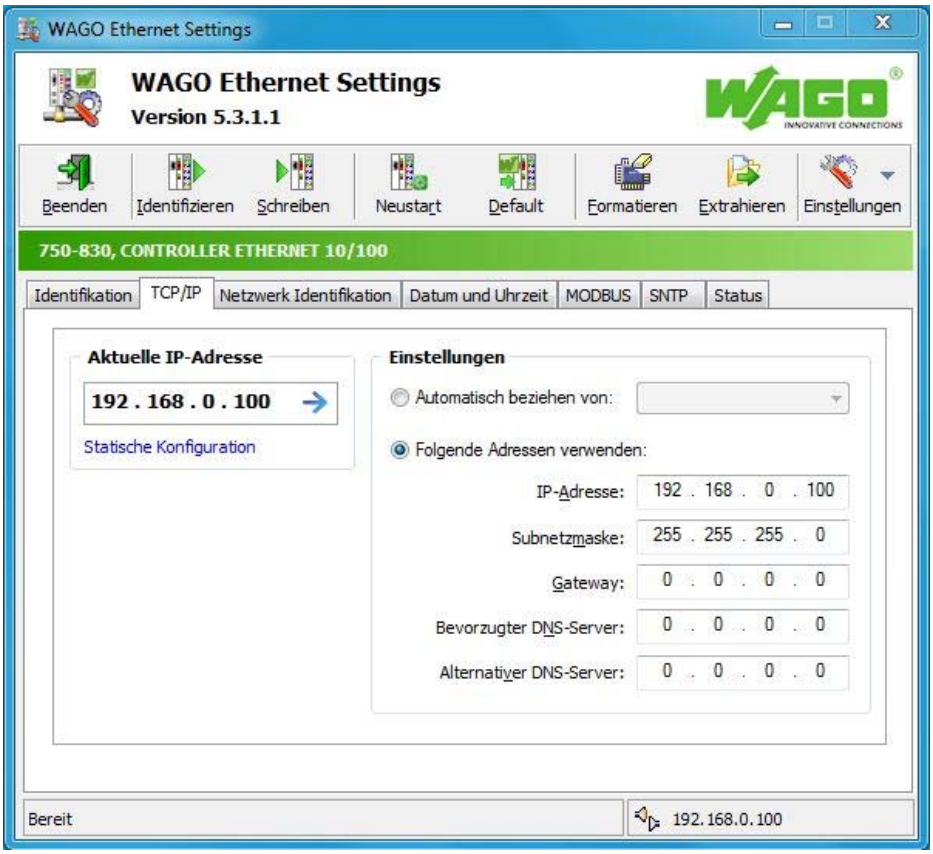


Abbildung 5: IP-Adresskonfiguration des Controllers

Als nächstes sollte im Reiter „Datum und Uhrzeit“ die Uhrzeit kontrolliert und bei Bedarf neu gesetzt werden (Auswahl von „Übernehmen“).

Vor dem Schreiben kann das System zurückgesetzt werden (Zunächst Schaltfläche „Formatieren“ anwählen und anschließend Schaltfläche „Extrahieren“). Damit wird das interne Dateisystem bereinigt.

Parallel könnten alle Einstellungen in den Werkszustand zurückgesetzt werden (Anwahl der Schaltfläche „Default“). → Nicht im Praktikum durchführen!

Nach dem Schreiben der IP-Adresse ist das Servicekabel zu entfernen (sicherheitshalber den Controller spannungsfrei schalten) und der Controller über den LAN-Anschluss anzuschließen.

Falls der Controller an ein LAN (d.h. Switch/Router) angeschlossen wird, muss lediglich vorher beachtet worden sein, dass die IP-Adresse des Controllers zur Konfiguration des LAN’s passt (d.h. den DHCP-Einstellungen).

Der PC muss nicht konfiguriert werden. Seine IP-Adresse lautet **192.168.0.10** und ist bereits fest eingestellt (Kontrolle über Adaptereinstellungen!).

Tutorial: WAGO

Die Kommunikation zum Controller kann getestet werden über:

- Start eines normalen Browsers und Eingabe der IP-Adresse in die URL-Zeile → Die Webseite des Controllers sollte erscheinen:

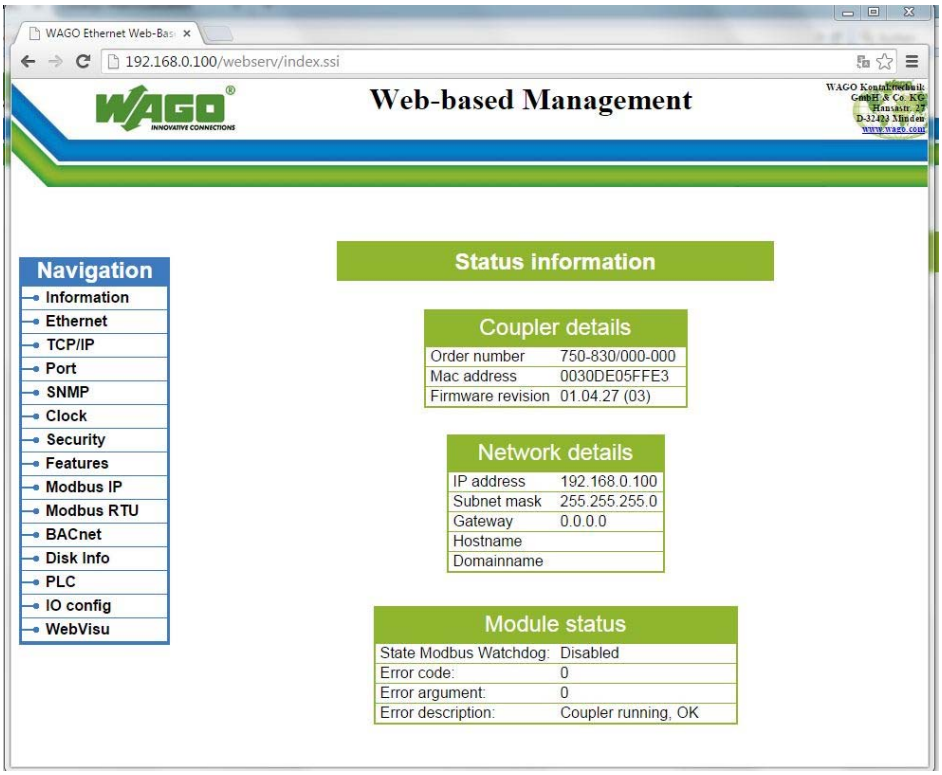


Abbildung 6: Zugriffstest zwischen PC und Controller (Browser)

3.2 IP-Adresse über BootP (nur INFO)

Alle IO-Controller mit Ausnahme des KNX-IP-Controllers (750-849) werden vom Werk mit der Grundeinstellung „BootP“ ausgeliefert. Über das Programm „Wago BootPServer“ kann dem Controller über eine Ethernet-Verbindung eine IP-Adresse zugewiesen werden.

Nach der Installation „BootP“ starten und auf die Schaltfläche „ Edit bootptab“ klicken. Diese enthält neben dem zu übertragenden Knotennamen auch die zukünftige IP-Adresse.

Der PC ist mit dem IO-Controller zu verbinden (über Kreuzkabel direkt oder über normale LAN-Kabel und einen Switch). Das BootServer-Programm ist zu starten (Programmstart sowie Schaltfläche „Start“). Der Controller ist zurückzusetzen (Stromversorgung ist kurzzeitig zu trennen oder der Schiebeschalter unter der Abdeckklappe kurz zu drücken). Das BootP-Programm muss den Knoten finden und programmiert Name und IP-Adresse.

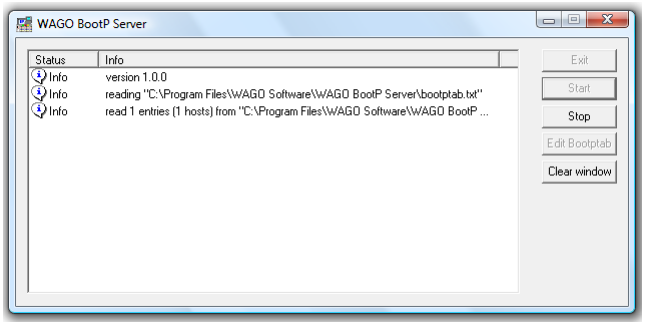


Abbildung 7: IP-Adressvergabe über BootP-Server

Anschließend kann der Knoten direkt über die neue IP-Adresse angesprochen werden; es öffnet sich das Web-Based-Management (WBM). Dort ist die Einstellung des BootP zurückzusetzen.

4 Web-Based-Management (WBM) (nur INFO)

Über das WBM werden die grundsätzlichen Einstellungen des Controllers vorgenommen. Zum Starten ist die IP-Adresse des Controllers in die URL-Zeile eines Webbrowsers einzugeben.

Werkseitig sind folgenden Logins zum Zugriff auf Unterseiten möglich:

- User: „admin“ mit Password „wago“
- User: „user“ mit Password „user“
- User: „guest“ mit Password „guest“

Sofern die Uhrzeit noch nicht gesetzt wurde (siehe Netzwerkkonfiguration über USB-Kabel) ist dies hier nachzuholen (Lasche „Clock“).

5 IO-Check

Das Programm „IO-Check“ ermittelt die konkrete Konfiguration eines Knotens samt angeschlossenen Klemmen. Das Ergebnis kann später von CoDeSys importiert werden. Es macht dann Sinn, wenn der Knoten bereits fertig aufgebaut vorliegt. Sollte das nicht der Fall sein, muss unter CoDeSys die Konfiguration manuell eingegeben werden.

Nach dem Start des Programms ist zunächst die Kommunikationsart über „Einstellungen-Kommunikation“ auf **Ethernet** einzustellen. Dort ist die IP-Adresse des Controllers **(192.168.0.100)** einzugeben oder es kann der Controller auch innerhalb eines IP- Adressbereichs gesucht werden.

Tutorial: WAGO

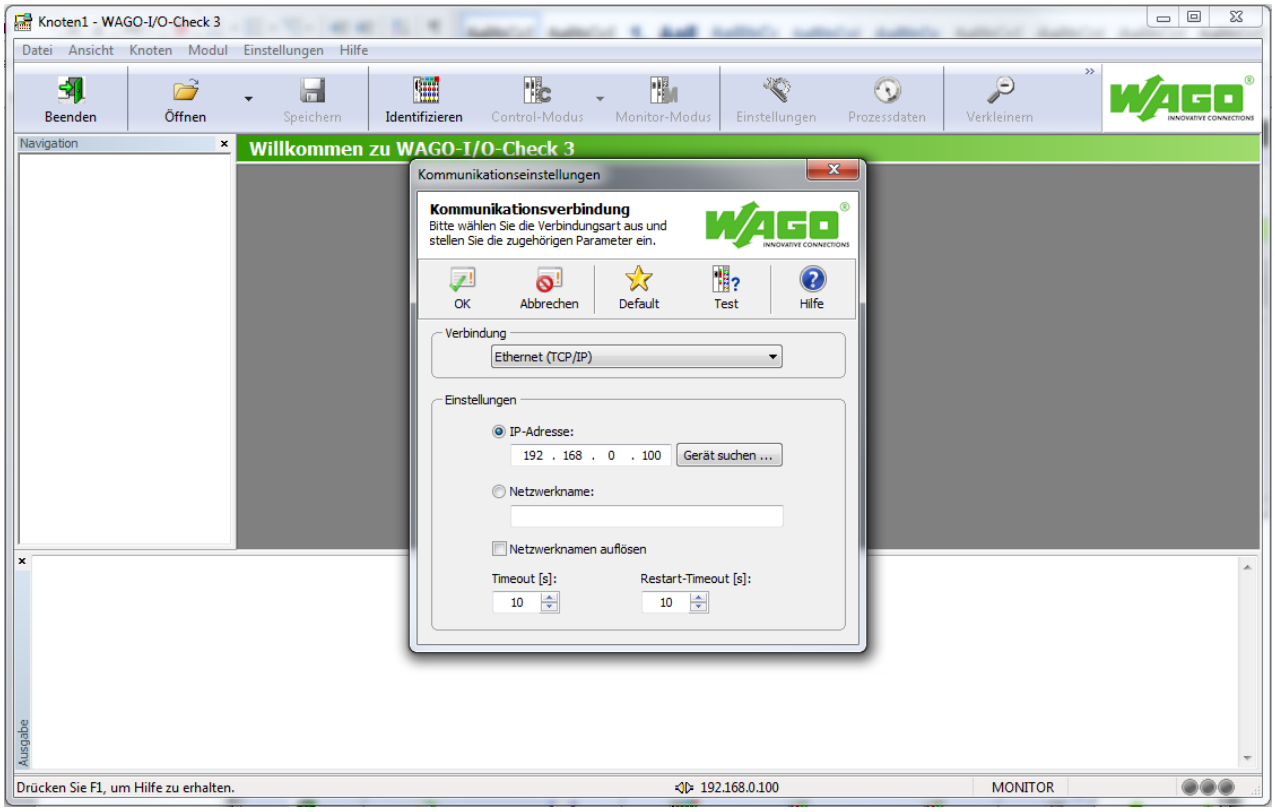


Abbildung 8: IO-Check: Einstellung der IP-Adresse des Controllers

Anschließend ist der Punkt **„Identifizieren“** anzuwählen. Der Aufbau des Knotens samt allen Klemmen wird erkannt und angezeigt. Das Ergebnis sollte aber auf jeden Fall manuell überprüft werden!

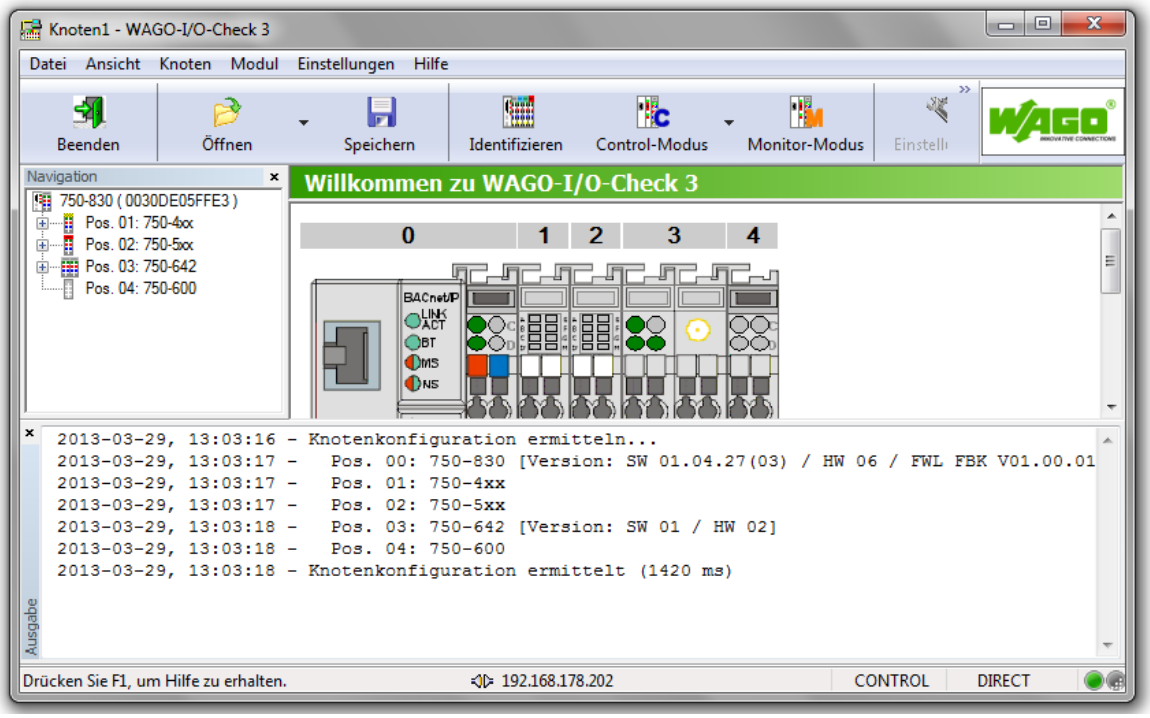


Abbildung 9: Automatische Ermittlung der Controller-Bestückung

Tutorial: WAGO

Im vorliegenden Fall wurden zwei Klemmen nicht richtig erkannt (Pos. 01 und Pos 02.; zu erkennen an den Bezeichnungsendungen „xx“). Dies muss manuell ergänzt werden. Dazu wählt man die Klemme aus und wählt **„Umbenennen“**. Anschließend kann der exakte Untertyp festgelegt werden.

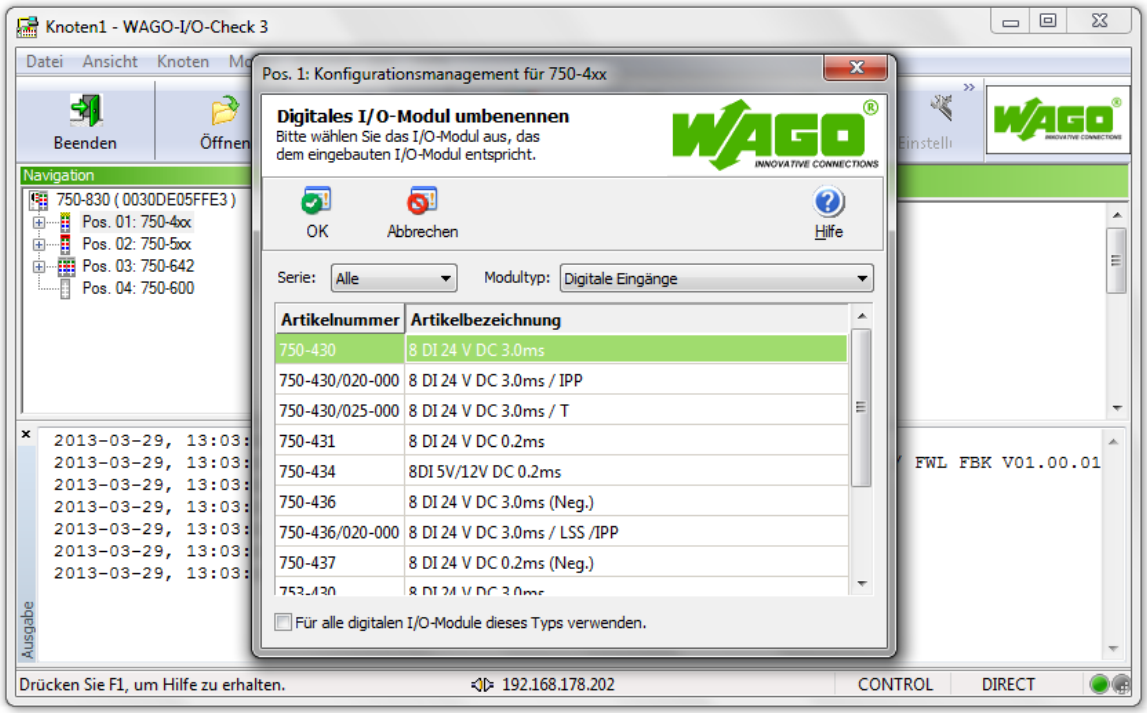


Abbildung 10: Festlegung der exakten Untervariante einer Klemme

Hinweis: Manche Klemmen können, bzw. müssen noch weiter konfiguriert werden (insb. z.B. Widerstandsklemmen, d.h. Klemmen mit der Bezeichnung „adjustable“). Dies erfolgt ebenso im Programm „IO-Check“. Die Klemme ist auszuwählen und „Einstellungen“ ist auszuwählen. Falls dieser Punkt nicht auswählbar ist, kann die Klemme nicht weiter konfiguriert werden oder der Control-, bzw. Monitor-Modus ist aktiv. Sobald man das Konfigurationsmenü der Klemme erhalten hat, ist dies zunächst mit Standardwerten gefüllt. Um die aktuelle Konfiguration der Klemme zu erhalten, ist „Lesen“ auszuwählen. Danach lassen sich die Einstellungen vornehmen und die Konfiguration mit „Schreiben“ in die Klemme übertragen. Über ein erneutes Lesen kann getestet werden, ob die Werte korrekt geschrieben wurden.

Anschließend kann die Konfiguration als XML-Datei über „Speichern“ gespeichert werden. Diese Datei kann später unter CoDeSys eingelesen werden; d.h. die Konfiguration kann automatisch übernommen werden.

Tutorial: WAGO

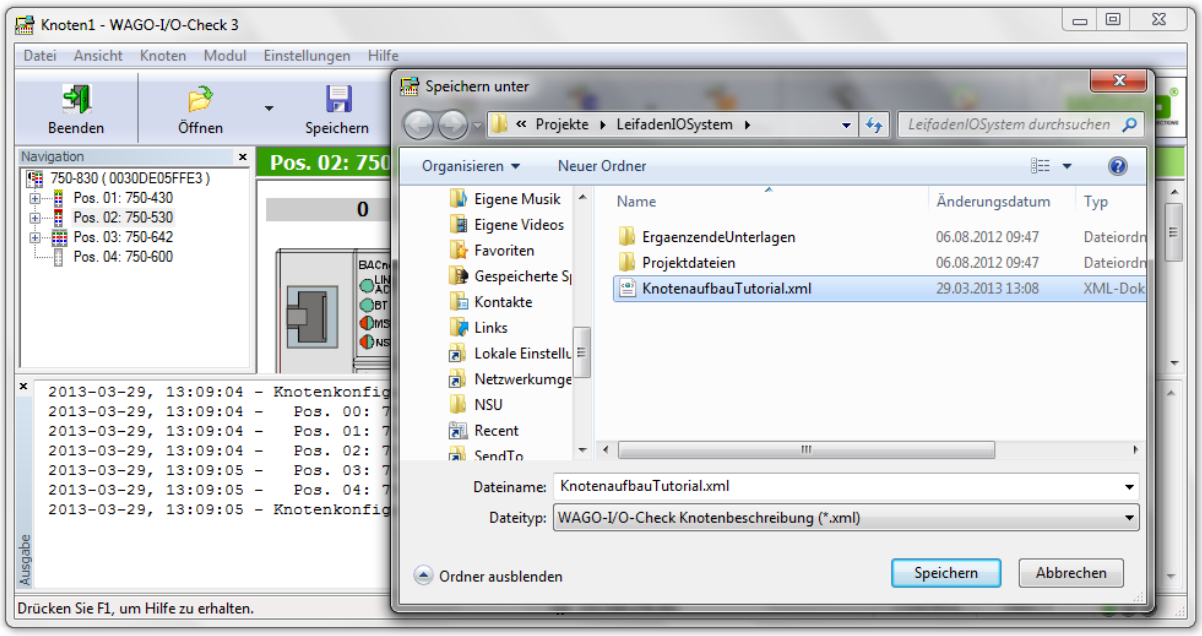


Abbildung 11: Speichern der ermittelten Konfiguration

6 SPS-Software – Grundsetup

Der Controller wird gemäß SPS (IEC61131) programmiert; konkret über das Entwicklungssystem **CoDeSys**. Dabei ist CoDeSys auf das Zielsystem „Wago“ angepasst.

Zur Programmierung unter CoDeSys ist ein Grund-Know-How der SW erforderlich.

6.1 Bibliotheken

Für den IO-Controller ist eine Vielzahl an Bibliotheken verfügbar. Diese sind im Ordner

\Programme(x86)\WAGO Software\CoDeSys V2.3\Targets\WAGO\Libraries

und in den folgenden Unterordnern gespeichert.

Bei Bedarf können eigene Bibliotheken erstellt und als interne Bibliotheksdatei gespeichert werden.

Zusätzlich existieren unter **www.wago.com** weitere freie Bibliotheken. Aber Achtung: diese sind nicht formell getestet und somit haben einige davon Hobbybastler-Qualität.

Vom IO-System können bis zu 1023 Bibliotheken geladen werden; deshalb ist die Strategie von WAGO die Wahl vieler kleiner Bibliotheken als wenige große.

6.2 Source-Code / Kompilat (nur INFO)

Die SW kann immer nur dann verändert werden, wenn der Original-Source-Code zugänglich ist. Beim Überspielen des Programms auf den Controller wird üblicherweise nur das Kompilat überspielt, so dass ein Reverse-Engineering nicht möglich ohne weiteres möglich ist. Der Source-Code gehört lizenzrechtlich dem Ersteller; er muss nicht zwangsläufig bei der Übergabe des Systems an den Kunden mit übergeben werden. Optional kann auf den Controller auch eine Kopie des Source-Codes gespielt werden. Dann sollte aber sinnvollerweise in Verbindung mit einem Password-Schutz erfolgen.

Dabei ist darauf zu achten, dass die Version von CoDeSys sowie der Bibliotheken zueinander passen (insbesondere bei großen Projekten zu beachten).

6.3 Konfiguration des Controllers sowie Task-Zykluszeit

Unter CoDeSys muss die Konfiguration des Controllers hinterlegt werden (Knotenaufbau) und die Ein- und Ausgänge sollten zusätzlich symbolische Namen erhalten. Zusätzlich ist die Task-Zeit des Controllers vorzugeben.

6.4 Knotenaufbau

Dazu ist zunächst CoDeSys zu starten. Beim Start ist das Zielsystem auszuwählen **(750-830).** Der darauf erscheinende Menüpunkt kann ohne Veränderungen bestätigt werden.

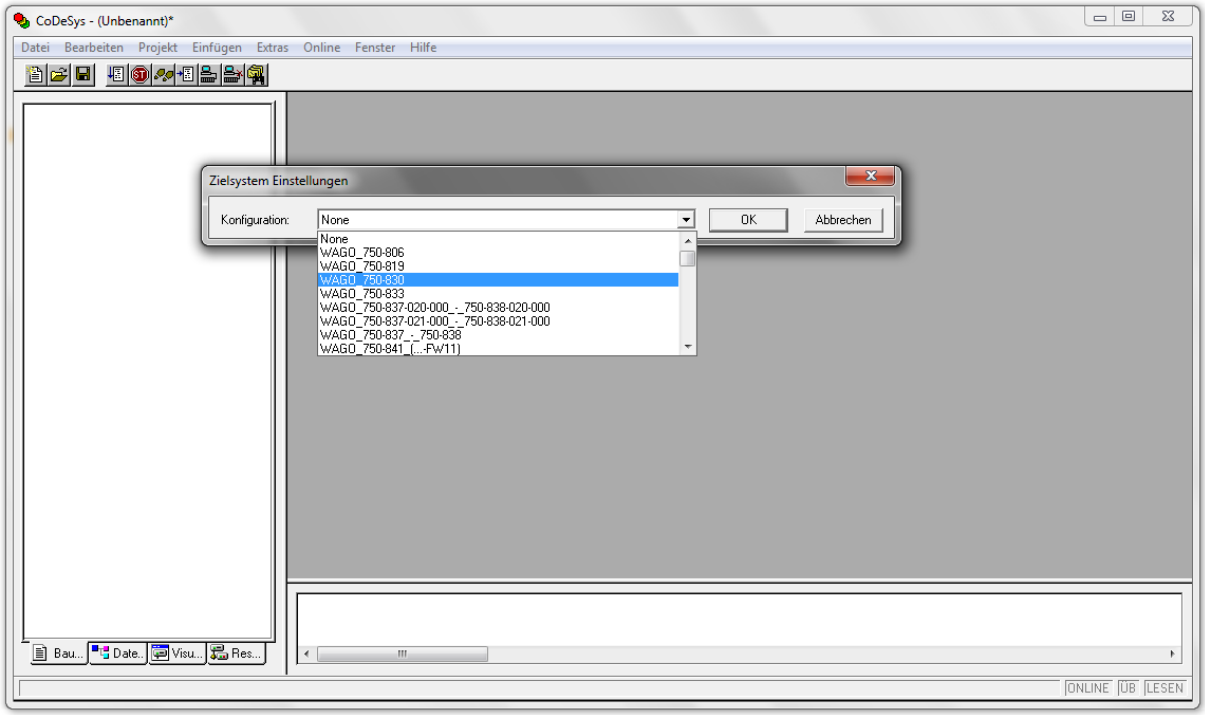


Abbildung 12: Angabe des Zielsystems (Controller-Typ)

Im nächsten Dialog ist der Name des Haupt-Programm-Bausteins (immer PLC_PRG) zu bestätigen und die Wahl der Programmiersprache zu treffen. Für einfache Programme ist CFC (Continuous Flow Chart) zu empfehlen, wodurch später eine einfache graphische Programmierung über Funktionsbausteine möglich wird. Eine anderweitig häufige Variante ist ST (Structured Text), welches einer textlichen Programmierung (ähnlich C oder Java – aber einfacher und weniger funktional) entspricht.

Die Beispiele im Tutorial sind in CFC geschrieben, gerne kann aber auch eine andere Programmiersprache genutzt werden!

Tutorial: WAGO

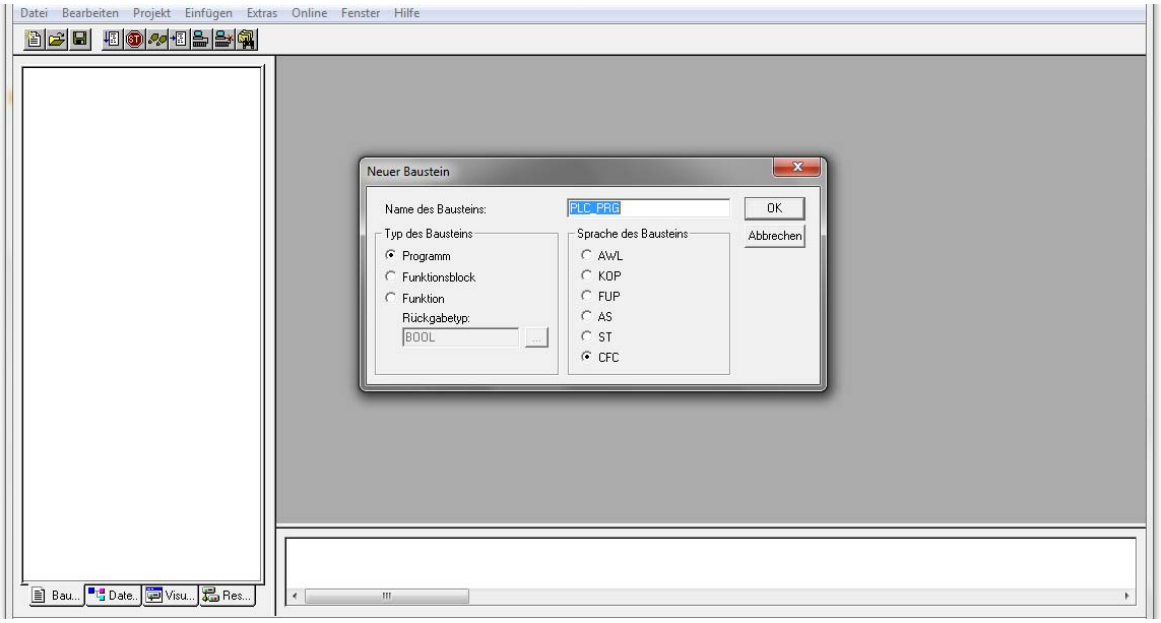


Abbildung 13: Festlegung der Programmiersprache

Danach muss im Menüpunkt „Ressourcen/Steuerungskonfiguration“ (siehe Abbildung 14) der Knotenaufbau des Controllers hinterlegt werden. (im linken unteren Teil des Bildschirms den entsprechenden Reiter auswählen).

Mit der rechten Maustaste auf das Element „K-Bus“ gehen und „Unterelement anhängen“ wählen. Dort in den Reiter „Konfiguration“ wechseln und die vorhin erstellte XML-Datei (vom Tool IO-Check) über die „Import“-Schaltfläche laden.

Tutorial: WAGO

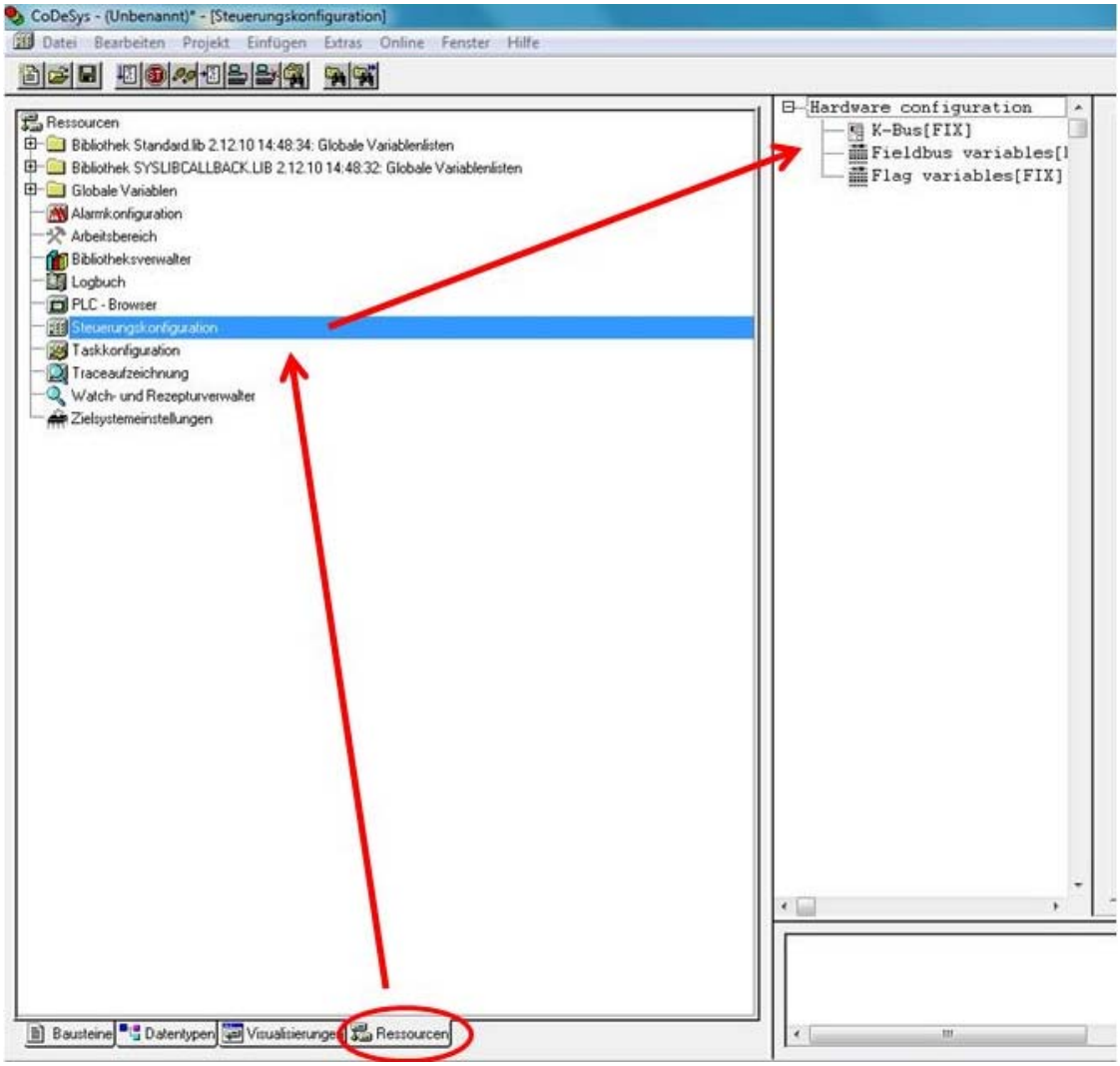


Abbildung 14: Import der .xml-Datei

Nach dem Import der Konfiguration muss der Aufbau des Controllers mit allen Klemmen ersichtlich sein!

6.4.1 Benennung der IO-Ports

Um später mit Namen auf die IO-Ports zugreifen zu können, müssen den IO-Ports symbolische Namen zugeordnet werden. Dies erfolgt dadurch, dass im Reiter „Ein-/Ausgänge“ die entsprechende Klemme angewählt wird. Dort kann jedem Ein-/Ausgang nun ein Name zugewiesen werden. Der Name, der dort vergeben wird, ist später für Ein- oder Ausgänge als Globale Variable auswählbar.

Die benötigen IO-Ports können zusätzlich kommentiert werden. Dazu sind die entsprechenden IO-Ports anzuwählen und es kann jeweils ein Kommentar eingegeben werden.

Artikel 750-430: E1 **„Taster_01“** und E2 **„Taster_02“** benennen

Artikel 750-530: A1 **„Relais_01“** und A2 **„Relais_02“** benennen

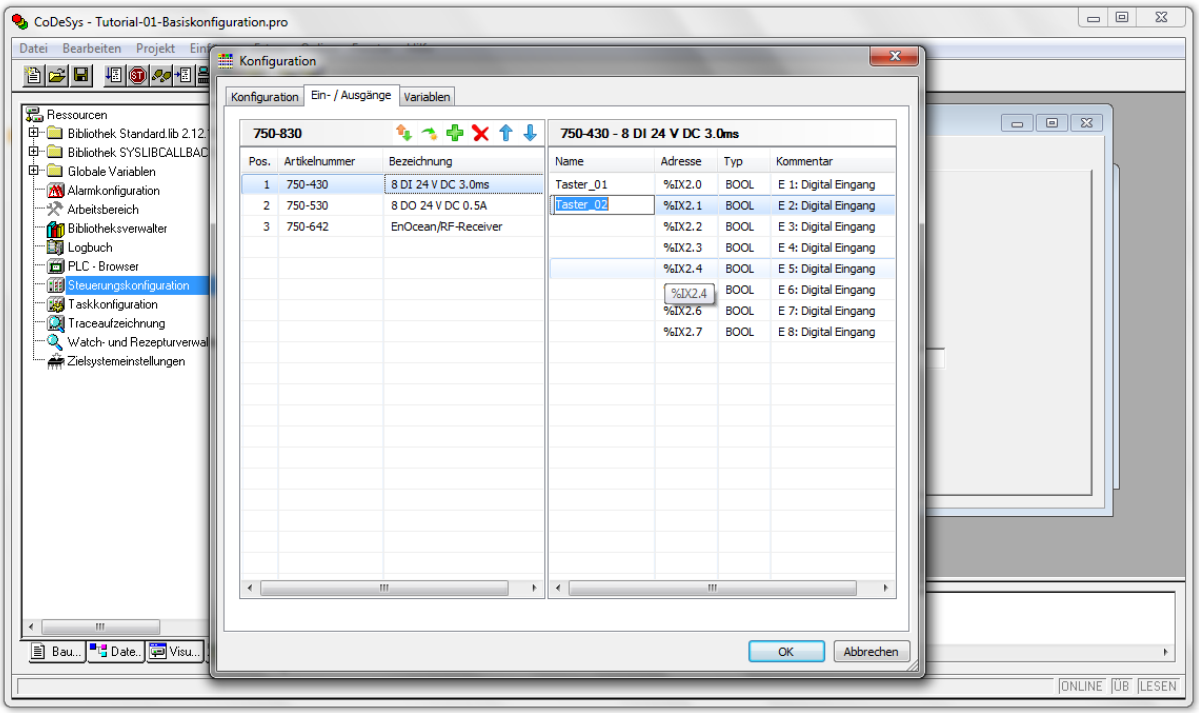


Abbildung 15: Benennung der Ein-/Ausgänge

6.4.2 Task-Konfiguration

Anschließend ist die Zykluszeit des Controllers zu setzen, d.h. der Taktgeber des Systems.

Tasks sind „Schrittmacher“ und können mit den Funktionsbausteinen verknüpft werden. Dadurch werden diese periodisch aufgerufen und können die Eingangssignale verarbeiten und Ausgangssignale aktualisieren. Es können mehrere Tasks erstellt werden. So verlangen manche Prozesse eine sehr schnelle periodische Aktualisierung (z.B. alle 10 ms); manche Prozesse können in größeren Zeitabständen aktualisiert werden (z.B. alle 15 Minuten).

Über „Ressourcen/Taskkonfiguration“ kann eine neue Task erstellt werden (Task anhängen)

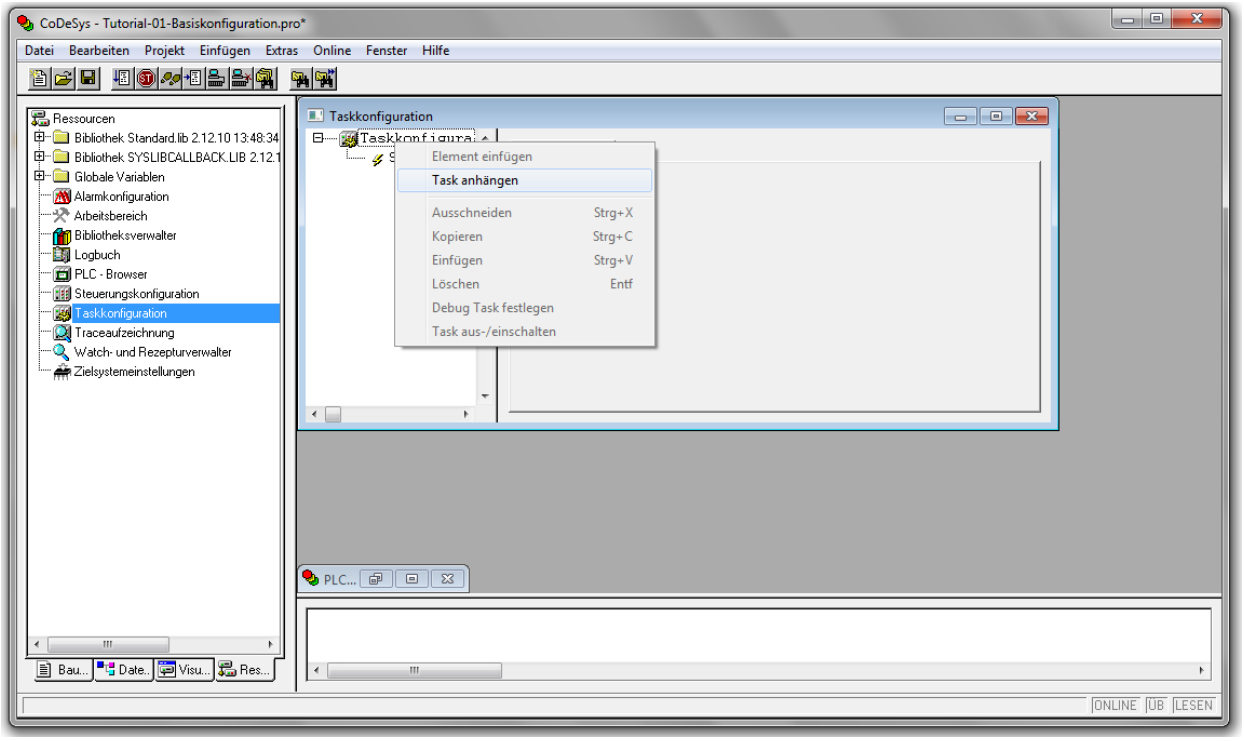


Abbildung 16: Definition eins neuen Tasks

Diese Task ist nun zu konfigurieren. Zum einen ist die Zykluszeit vorzugeben – im vorliegenden Fall 15 ms - und parallel wird der Name „Systemtakt“ vergeben.

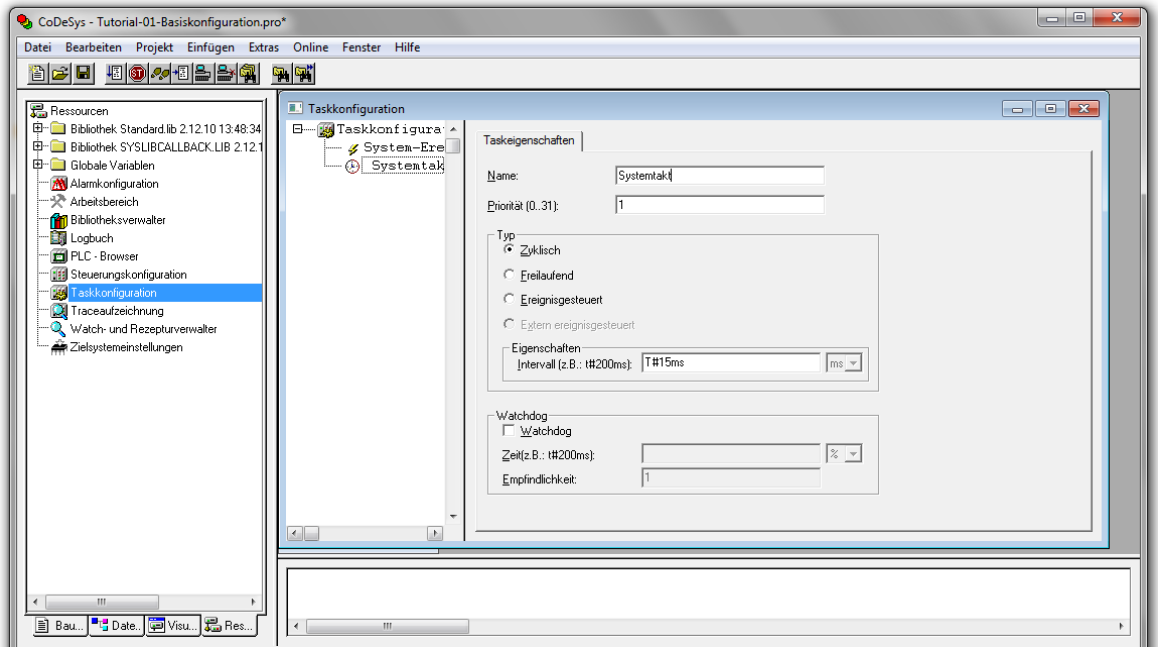


Abbildung 17: Konfiguration des Tasks

Dabei ist zu beachten, dass zwischen den periodischen Aufrufen genug Zeit für Programmausführung und Systemsettings erlaubt. Für kleine überschaubare Programme kann eine Zykluszeit von 15 ms gesetzt werden. Später kann die tatsächlich benötige Zeit angezeigt werden und die Zykluszeit muss notfalls nachoptimiert werden.

Diese Task ist mit dem Funktionsbaustein PLC_PRG zu verknüpfen (anzuhängen). Dazu ist die Task mit der rechten Maustaste anzuwählen und der Unterpunkt „Programmaufruf anhängen“ auszuwählen. Dann ist der betreffende Baustein (hier: PLC_PRG) auszuwählen.

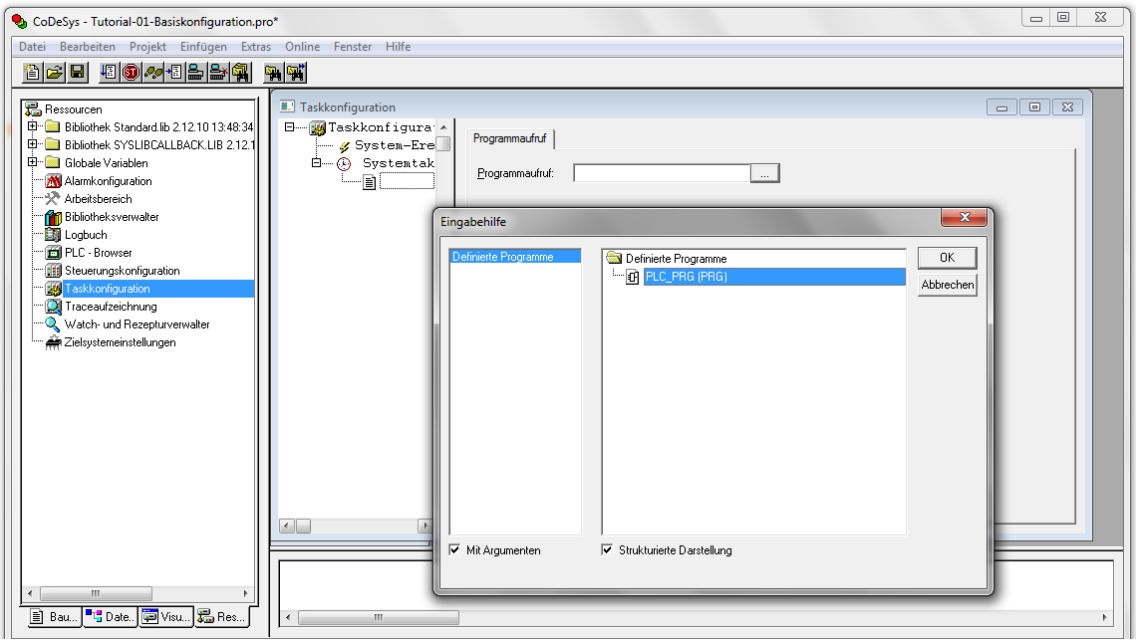


Abbildung 18: Verknüpfung des Tasks mit dem Programm

6.5 Export der Grundkonfiguration

Die bisher erstellte Grundkonfiguration kann exportiert werden, um diese später erneut zu verwenden. Dies ist dann hilfreich, wenn für das gleiche Zielsystem unterschiedliche Programme geschrieben werden und öfters auf eine Basiskonfiguration zurückgegriffen werden soll.

Aufruf über „Projekt-Exportieren“ und Markieren der gewünschten Komponenten **(hier sollte nur die Steuerungskonfiguration und Taskkonfiguration exportiert werden, da sonst noch keine Konfigurationen vorgenommen wurden).**

6.6 Übertragung (Konfiguration/Programme) in den Controller

Zum Übertragen von Konfiguration und Programmen in den IO-Controller ist unter CoDeSys der Punkt **„Online – Kommunikationsparameter“** anzuwählen. Dort ist über **„Neu“** ein neuer Kommunikationskanal anzulegen. Dabei ist die IP-Adresse des Controllers einzutragen und der Port 2455 (TCP) auszuwählen. Diese Festlegung erfolgt ein einziges Mal und muss erst dann wiederholt werden, wenn sich die IP-Adresse des Controllers ändert.

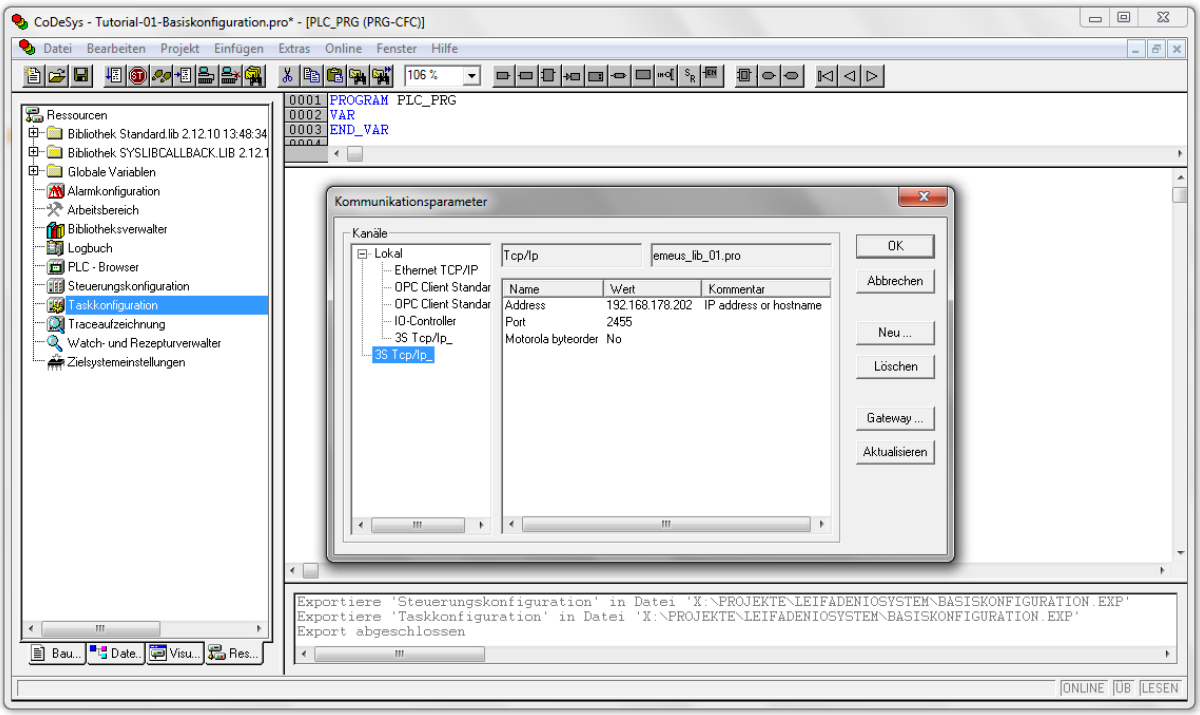


Abbildung 19: Einstellung der IP-Adresse des Controllers

Über **„Online-Einloggen“** (bzw. Alt-F8) wird das Programm übertragen. Es erscheint ein Fenster, welches man mit „Ja“ bestätigt. Das BACnet-Abbild muss nicht aktualisiert werden (→ „Nein“!). Über **„Online-Start“** (bzw. F5) wird es gestartet.

Tutorial: WAGO

Wenn das Programm auch nach Reset/Abschalten des Controllers automatisch geladen und gestartet werden soll, muss folgendes durchgeführt werden:

- Unter **„Zielsystemeinstellungen - Allgemein“** den Haken bei **„Bootprojekt automatisch laden“** setzen
- In CoDeSys einloggen und dann im eingeloggten Zustand unter **„Online“** den Punkt **„Bootprojekt erzeugen“** auswählen. Das Bootprojekt „Default.cfr“ wird erzeugt und direkt in die Steuerung geschrieben.

7 Einfache Demo-Programme

Im Folgenden werden einige einfache Demo-Programme beschrieben, um einige grundsätzliche Dinge im Umgang mit dem IO-System zu demonstrieren. Dabei werden zwei binäre Eingänge sowie zwei binäre Ausgänge genutzt. Die Ausgänge sind mit je einem Relais verbunden, der auf eine Steckdose wirkt. Über diese Steckdosen wird in diesen Beispielen je eine Leuchte angesteuert.

Es wird auf die Basiskonfiguration wie bisher beschrieben zurückgegriffen. D.h. es liegt ein Controller mit einer binären Eingangsklemme vor, an die zwei Taster angeschlossen sind. An einer binären Ausgangsklemme ist an je einem Ausgang ein Relais angeschlossen. Die Ein- und Ausgänge haben die in „7.3.2.Benennung der IO-Ports“ festgelegten Namen **„Taster_01“** und **„Taster_02“** sowie **„Relais_01“** und **„Relais_02“**.

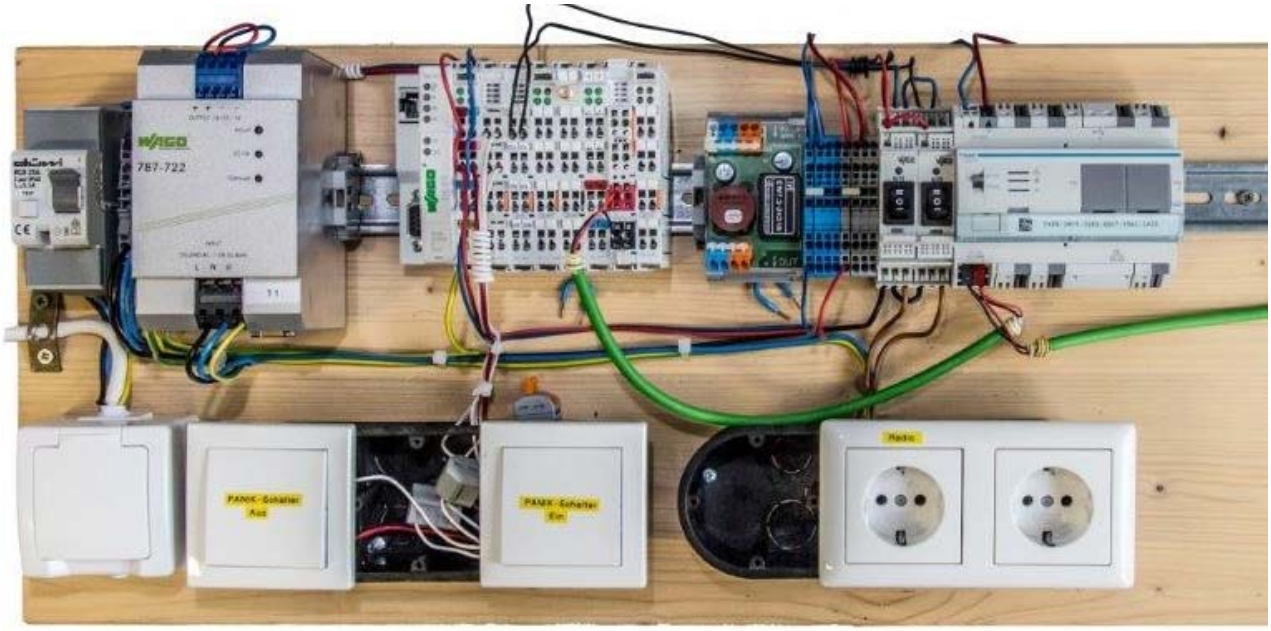


Abbildung 20: Demoaufbau (Komponenten)

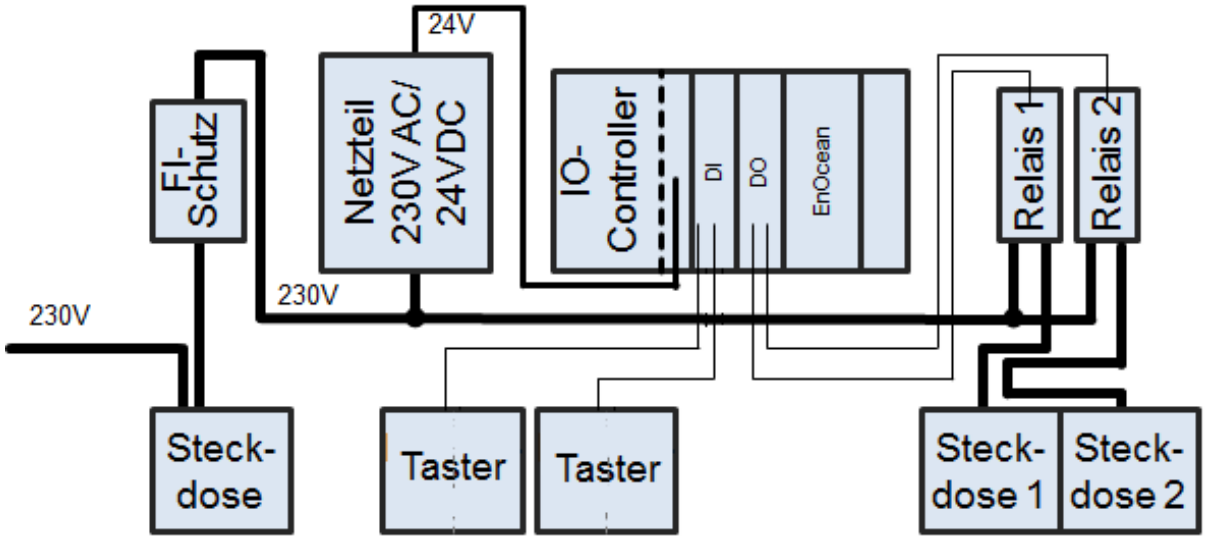


Abbildung 21: Demoaufbau (Blockschaltbild)

7.1 Und-Verknüpfung zweier Eingänge auf einen Ausgang

Im CFC-Fenster des Programms PLC_PRG ist über „Strg-B“ oder „rechte Maustaste – Baustein“ ein Baustein einzufügen. Standardmäßig wird immer zunächst ein AND-Baustein eingefügt. Über „F2“ kann dieser ein einen beliebigen anderen Baustein ausgetauscht werden - im vorliegenden Fall bleibt es zunächst beim eingefügten AND-Baustein.

Zusätzlich zum AND-Baustein werden ebenfalls über „rechte Maustaste“ und „F2“ zwei Eingänge und ein Ausgang eingefügt (alternativ mit 2 x Strg-E und 1 x Strg-A). Diese werden zunächst mit drei Fragezeichen dargestellt.

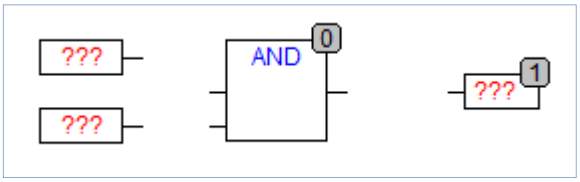


Abbildung 22: AND-Baustein

Mit der Maus können die ausgehenden Linien der Ausgänge auf die Eingänge des AND-Bausteins gezogen werden. Ebenso wird mit der Maus der Ausgang des AND-Bausteins auf den Eingang gezogen. Damit werden die Verknüpfungen hergestellt.

Nun muss noch festgelegt werden, dass die Eingänge die Kontakte der entsprechenden Klemmen sind. Dazu ist jeder Eingang und Ausgang anzuwählen und über „F2“ kann der entsprechende Eingang, bzw. Ausgang ausgewählt werden.

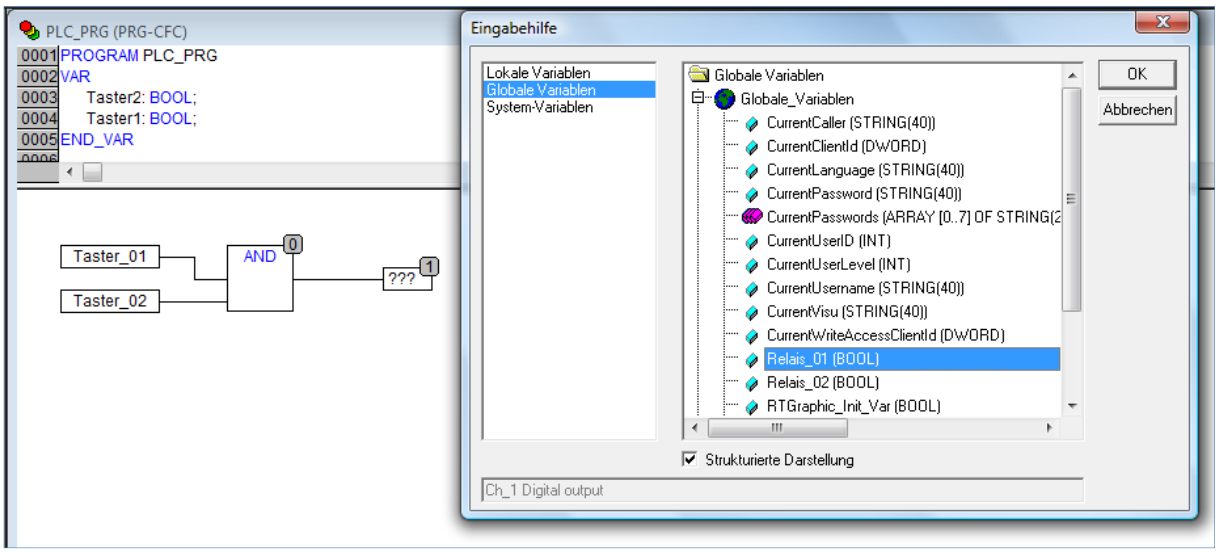


Abbildung 23: Zuweisung des Ausgangs

Im vorliegenden Fall, werden die Eingänge „Taster_01“ und „Taster_02“ und der Ausgang „Relais_01“ ausgewählt.

Zunächst sollte geprüft werden, ob das Programm fehlerfrei in Maschinencode übersetzt werden kann. Die Ausführung von „Projekt – Übersetzen (bzw. F11)“ oder „Projekt – Alles übersetzen“ muss ohne

Fehler durchgeführt werden – das Ergebnis wird im Meldungsfenster rechts unten angezeigt und sollte auf „0 Fehler“ lauten.

7.2 Test in der Simulation

Zunächst soll das Programm im Simulationsmodus getestet werden. Dazu ist kein echter Controller nötig! Im Menüpunkt „Online“ muss dazu ein Haken in der Zeile „Simulation“ gesetzt sein.

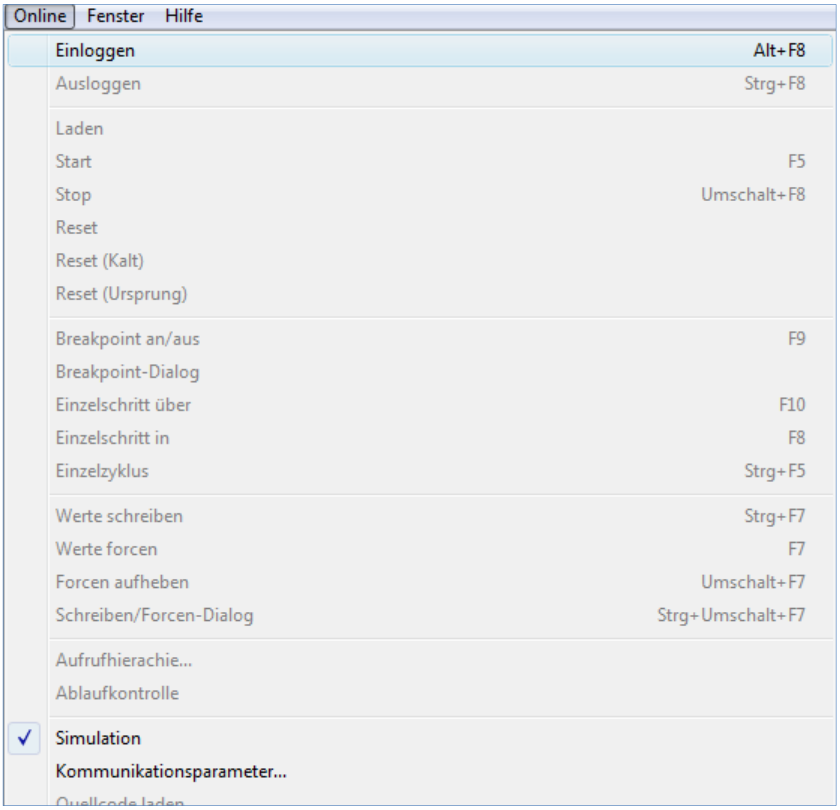


Abbildung 24: Test der Simulation

Anschließend ist erneut der Menüpunkt „Online“ und dort der Punkt „Einloggen“ auszuwählen (Alt-F8). Anschließend ist die Simulation noch zu starten („Online“ – „Start“ bzw. F5). Rechts unten auf dem Bildschirm erscheint die Info „Sim“ sowie „Läuft“.

Im CFC werden Signale „0“ bzw. FALSE schwarz und die Signale „1“ bzw. TRUE blau dargestellt. Mit einem Doppelklick der Maus auf einen Eingang kann ein Eingangssignal von FALSE auf TRUE umgeschaltet werden. Wichtig zu beachten ist, dass der Ausgang noch nicht automatisch neu berechnet wird. Dies erfolgt erst immer nach Ausführung von „Online“ – „Werte schreiben“, bzw. Strg-F7. Damit kann die Funktion der Und-Verknüpfung getestet werden.

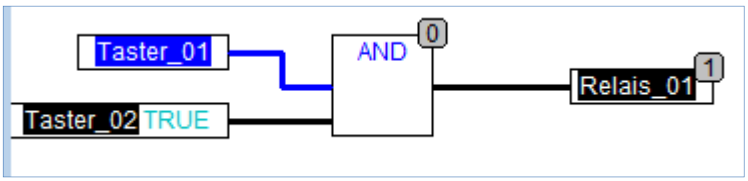


Abbildung 25: Test der Simulation

Über „Online“ – „Ausloggen“ bzw. Strg-F8 wird die Simulation beendet und man loggt sich aus der Simulation aus.

7.3 Test mit dem echten IO-Controller

Der Test mit dem echten Controller erfolgt ähnlich. Dabei ist darauf zu achten, dass unter „Online“ kein Häkchen bei „Simulation“ gesetzt ist. Über „Online“ – „Einloggen“ bzw. Alt-F8 wird das Programm in den Controller übertragen (dabei werden die eingegebenen Kommunikationsparameter von „Online“ – „Kommunikationsparameter“ verwendet). Der Start des Programms erfolgt über „Online – Start“ bzw. F5.

Wenn beide Eingänge mit einer positiven Spannung versehen werden, werden aufgrund der Und-Verknüpfung beide Eingänge als auch der Ausgang blau dargestellt.

Zusätzlich leuchten die LED’s des entsprechenden Eingangs sowie die LED des entsprechenden Ausgangs (bei den Klemmen des IO-Systems) auf.

Das Beenden des Programms erfolgt über „Online – Stopp“. Alternativ besteht die Möglichkeit, sich lediglich auszuloggen (d.h. es besteht keine direkte Verbindung zwischen PC und dem IO-System aber das Programm im IO-System wird weiterhin ausgeführt).

7.4 Werte “forcen”

Sowohl in der Simulation als auch in Verbindung mit dem echten Projekt lassen sich Werte „forcen“. Das bedeutet, dass ein Eingangs- oder Ausgangswert auf feste Werte setzen lassen (d.h. unabhängig vom echten Signalstatus aufgrund vom Eingangssignal oder den Ausgängen vorausliegender Bausteine.

Dazu ist ein Doppelklick mit der Maus auf das Signal nötig (es erscheint die Bezeichnung TRUE bzw. FALSE). Nun ist „Online“ – „Werte forcen“ bzw. F7 auszuwählen. Geforcte Werte werden im CFC rot dargestellt. Über „Online“ – „Forcen aufheben“ bzw. Umschalt-F7 wird die Forcierung aufgehoben. Über „Online“ – „Schreiben/Forcen – Dialog“ erscheint ein Dialog, in dem die Force-Liste (eventuell werden mehrere Signale geforct) bearbeitet werden kann.

7.5 Funktionsbaustein einfügen

Gemäß SPS-Programmiersprache wird unterschieden in Funktionen und Funktionsbausteine. Funktionen berechnen das Ausgangssignal ausschließlich und unmittelbar aufgrund der Eingangssignale und liefern genau eine Ausgangsvariable. Der AND-Baustein ist ein Beispiel für eine Funktion. Funktionsbausteine

selber berechnen Ausgangssignale auch aufgrund innerer Zustände und haben somit ein „Gedächtnis“. Dabei kann ein Funktionsbaustein mehrere Ergebnisvariablen verändern.

Im diesem Schritt soll ein *RS-Flipflop* eingesetzt werden, d.h. ein Funktionsbaustein der gesetzt und zurückgesetzt werden kann. Dabei merkt sich dieser Funktionsbaustein den zuletzt aktivierten Zustand.

Dazu wird der AND-Baustein angewählt und über Taste F2 eine Auswahl von Alternativen zur Anzeige gebracht. Unter „Standard-Funktionsblöcke“ werden die SPS-Funktionsbausteine angezeigt und unter „Bistable Function Blocks“ erscheint das RS-Flipflop. Dieses wird ausgewählt.

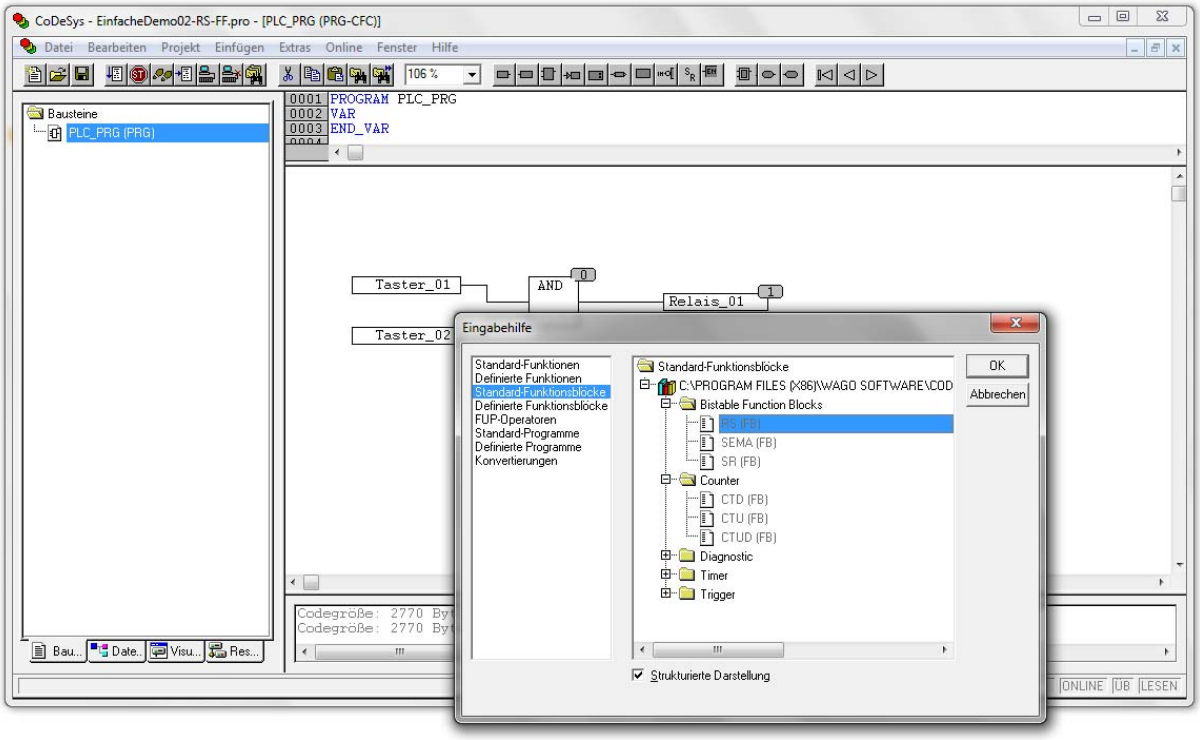


Abbildung 26: Auswahl RS-Flipflop

Das Flipflop wird anstelle des AND-Bausteins eingefügt. Oberhalb erscheinen drei Fragezeichen. Diese sind anzuwählen und durch einen Namen zu ersetzen (hier: Flipflop). Anschließend erscheint der folgende Dialog, über den das Anlegen einer Instanz (Programmelement) bestätigt wird. An dieser Stelle werden keine weiteren Änderungen vorgenommen und der Dialog wird über OK geschlossen. Details zu den Eingabemöglichkeiten werden später 8.7 (Variablen und WEB-Visualisierung) erklärt.

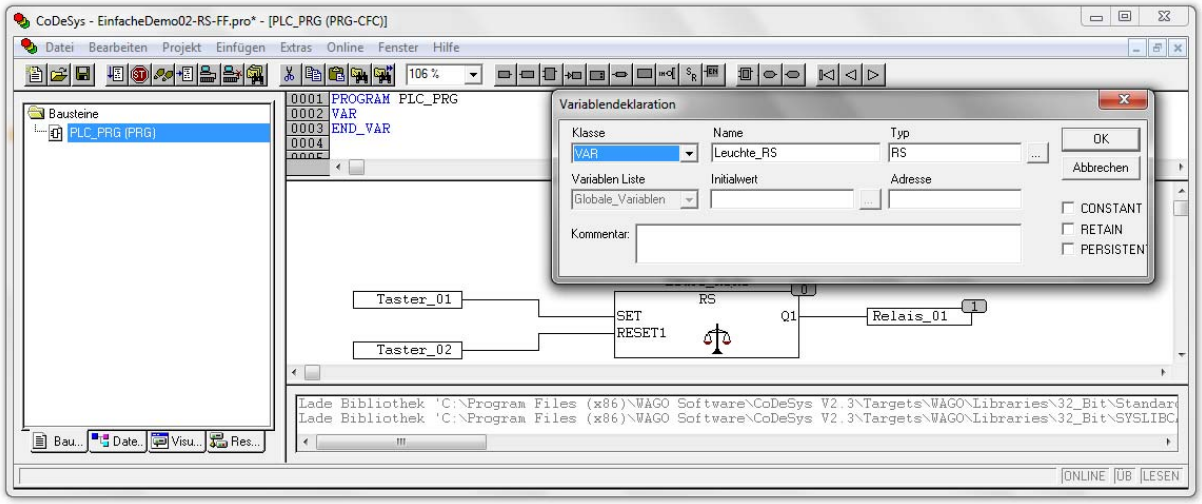


Abbildung 27: Variablendeklaration

Als Ergebnis sollte folgende die nachfolgende Beschaltung vorliegen. Dies ist in den Controller zu übertragen und zu starten. Damit lässt sich die Leuchte ein- und wieder ausschalten.

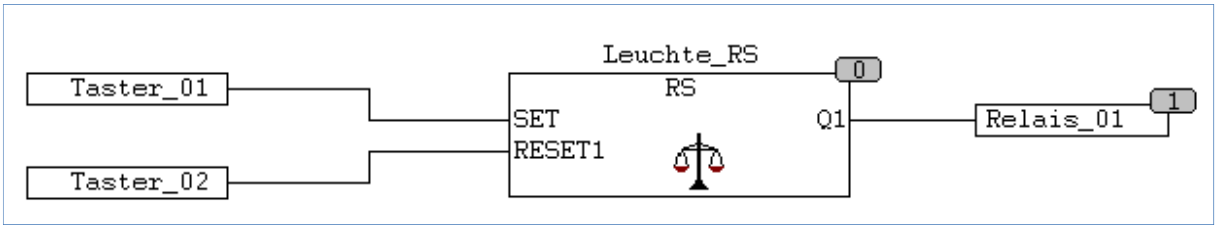


Abbildung 28: RS-Flipflop

7.6 Einfache Zeitschaltung

Die Schaltung soll nun dahingehend verändert werden, dass anstelle eines RS-Flipflops ein Timer zum Einsatz kommt. Konkret: Über den ersten Taster wird das Licht eingeschaltet und fällt nach 10 Sekunden automatisch wieder in den Aus-Zustand zurück. Der zweite Taster wird nicht mehr benötigt.

Anschließend wird aus dem Bausteinkatalog ein Timer-Baustein (TOF) eingefügt.

Dabei benötigt der Timer eine Eingangsvariable für die Verzögerungszeit. Dazu wird ein neuer Eingang in das CFC eingefügt und anstelle eines Namens wird gleich die Zeit im Format TIME eingetragen. In diesem Fall t#10s (bedeutet 10 Sekunden). Ansonsten können auch andere Werte wie „t#3h“ für 3 Stunden, „t#4m“ für 4 Minuten oder „t#10ms“ für 10 ms eingegeben werden. Auch Kombinationen wie z.B. „t#3h15m10s“ sind möglich.

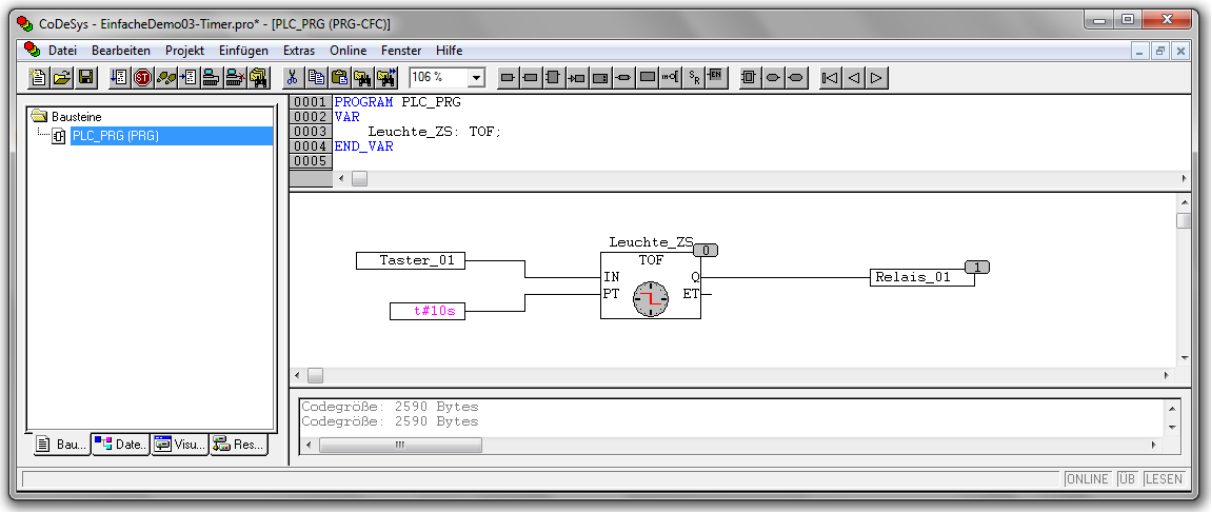


Abbildung 29: Timer-Baustein

Parallel sollte darauf geachtet werden, dass im oberen Fenster die Deklarationszeilen mit der Programmierung im graphischen Fenster übereinstimmt. Werden im Grafikfenster Bausteine gelöscht, bleiben im oberen Teil die Deklarationszeilen oft noch bestehen. Diese „Leichen“ sollten manuell gelöscht werden.

Das Programm kann nun gestartet und getestet werden: Nach Drücken des Tasters schaltet das Relais „Relais_01“ ein und nach 10 Sekunden wieder aus.

7.7 Variablen und WEB-Visualisierung

Die Schaltung soll im nächsten Schritt sowohl in CoDeSys als auch über eine Web-Oberfläche visualisiert werden. Zusätzlich soll auch eine Schaltfläche in der Visualisierung aufgenommen werden, die parallel zu den echten Tastern auf den vorhandenen Timer -Baustein wirkt.

7.7.1 Vorbereitungen

Dazu ist die Schaltung zunächst um einen Eingang („Button_01“) zu erweitern. Über <rechte Maustaste> und „Einfügen – Eingang“ erscheint wieder ein Eingang mit drei Fragezeichen. Dieser ist unterhalb des Eingangs „Taster_01“ zu platzieren. Diesmal wird dieser Eingang nicht mit einem physikalischen Eingang des IO-Systems, sondern mit einer Variablen verbunden. Dazu ist an der Stelle der drei Fragezeichen der Name „Button_01“ einzugeben. Nach Bestätigung erscheint folgendes Auswahlfenster:

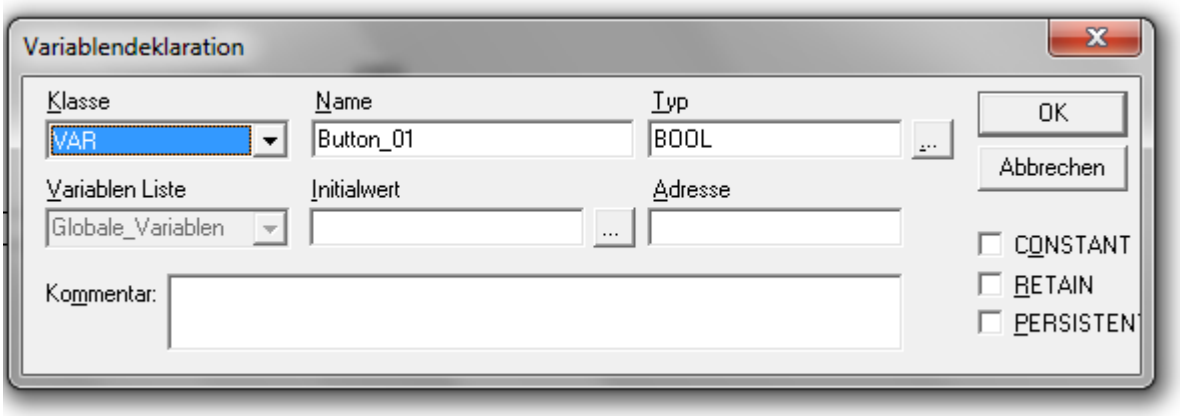


Abbildung 30: Variablendeklaration

Unter Klasse kann VAR belassen werden (Allg. Bezeichnung für Variable).

Hinweis: Alternativ kann in VAR_INPUT oder VAR_OUTPUT unterschieden werden. Dies hat keine Auswirkung auf die Programmausführen). Unter Name erscheint der gewählte Name (Button_01) und unter Typ ist der Variablentyp anzugeben (z.B. BOOL, REAL, INT, etc.). Unter Adresse könnte die physikalische Adresse eines Eingangs angegeben werden. Dadurch, dass echte Eingänge / Ausgänge Namen erhalten und diese als Globale Variable angesprochen werden, verliert dieses Feld hier an Bedeutung.

CONSTANT: der optionale Initialwert später nicht mehr geändert werden (Konstante)

RETAIN: der Inhalt der Variablen bleibt auch bei Abschaltung des Controllers erhalten

PERSISTENT: der Inhalt der Variablen bleibt auch bei SW-Update erhalten

Nun soll der Timer entweder durch den echten Taster ODER über die Webvisualisierung gestartet werden. Deshalb wird ein OR-Baustein eingefügt und zwischen die Taster/Buttons und den Timer geschaltet. Parallel soll der Zustand des Timers nicht nur auf das Relais wirken, sondern später auch in einer Visualisierung angezeigt werden. Deshalb wird eine Ausgangsvariable „Zustand_01“ angelegt und parallel zum Relais mit dem Ausgang des Timers verknüpft.

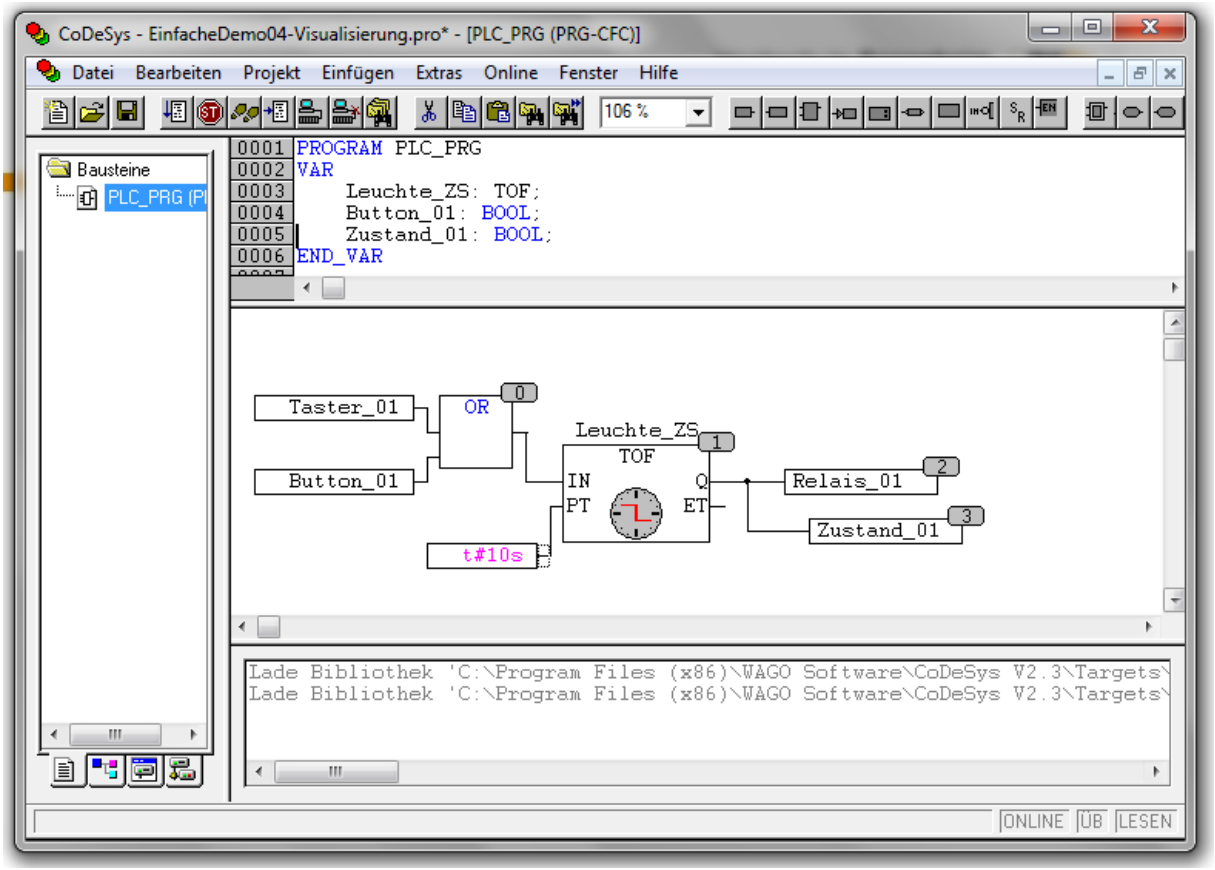


Abbildung 31: Vorbereitung für Visualisierung

Die Ziffern in den oberen rechten Ecken der Bausteine geben an, in welcher Reihenfolge die Ausgangssignale der Bausteine berechnet werden. Dies sollte immer von links nach rechts erfolgen. Unter „Extras“ – „Reihenfolge“ – „Alles nach Datenfluss anordnen“ erfolgt eine neue Anordnung der Reihenfolge (Nummerierung) von links nach rechts.

7.7.2 Visualisierung unter CoDeSys

Jetzt kann die Visualisierung angelegt und mit den Variablen verbunden werden. Dazu muss zunächst auf dem Bildschirm links unten in die Registerkarte „Visualisierung“ gewechselt werden. Links oben erscheinen unter „Visualisierungen“ die bereits angelegten Visualisierungen (eventuell über Doppelklick aufblättern lassen), wobei aktuell noch kein einziger Eintrag erscheint. Über das Kontextmenü (rechte Maustaste über „Visualisierungen“ kann der Befehl „Objekt einfügen“ ausgewählt werden. Es muss ein Name vergeben werden, der beim WAGO-IO-System zunächst immer „PLC_VISU“ lauten muss. Falls mehrere Webseiten gewünscht werden, können diese zusätzlich angelegt werden. „PLC_VISU“ ist dann die Hauptseite, von der die weiteren Unterseiten aufgerufen werden können.

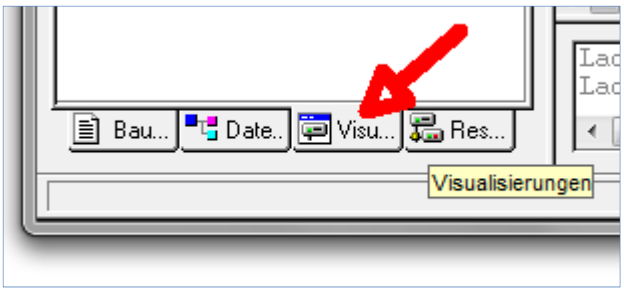


Abbildung 32: Auswahl Visualisierung

In dieser Visualisierung können nun entsprechende Elemente eingefügt werden. In diesem Beispiel wird ein Rechteck für die Aufnahme der Überschrift, eine Schaltfläche (Command Button) sowie ein Kreis zur Signalisierung des Ausgangs verwendet. Über einen Doppelklick auf das jeweilige Element kann dies entsprechend konfiguriert werden.

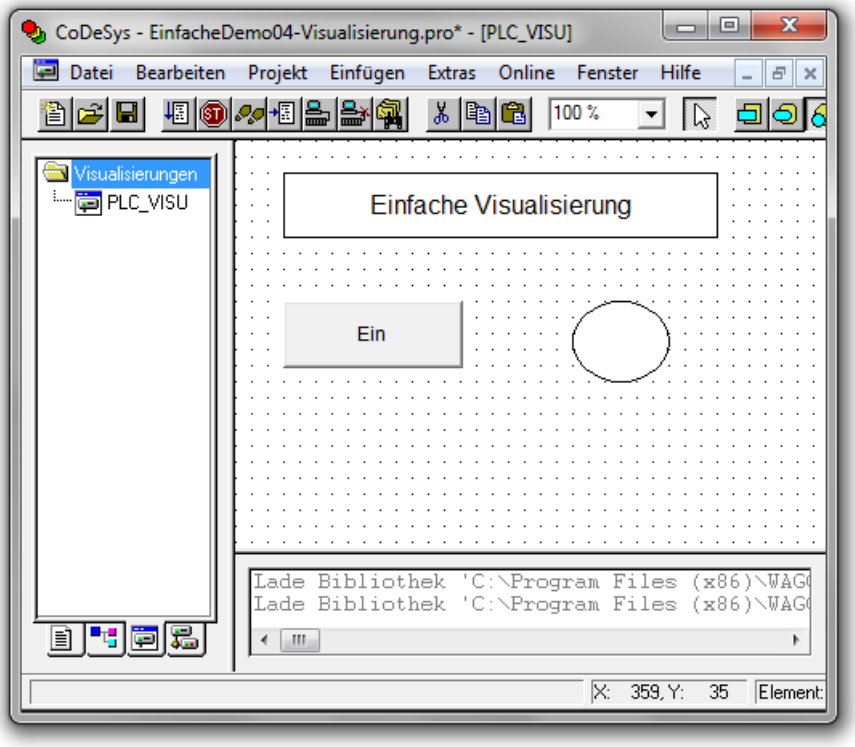


Abbildung 33: Visualisierung

Das Rechteck erhält die Überschrift sowie die Angabe, keine Rahmenfarbe zu verwenden. Zusätzlich wird die Schriftgröße für die Überschrift erhöht.

Die Schaltfläche (Elemente #1) erhält den Text „Ein“ als Beschriftung. Zusätzlich wird der Schaltflächen die entsprechende Variable aus dem Programm zugeordnet. Das geschieht über Doppelklick auf die Schaltflächen, Auswahl des Unterpunkts „Eingabe“, Anwahl des Punktes „Variablen tasten“ und Wechsel in das dahinterliegende Feld. Dort kann über F2 die Eingabehilfe aufgerufen werden. Unter dem Punkte „PLC_PRG (PRG)“ können die im Programm angelegten Variablen aufgelistet und die passende Variable ausgewählt werden. Dies ist für beide Schaltflächen durchzuführen.

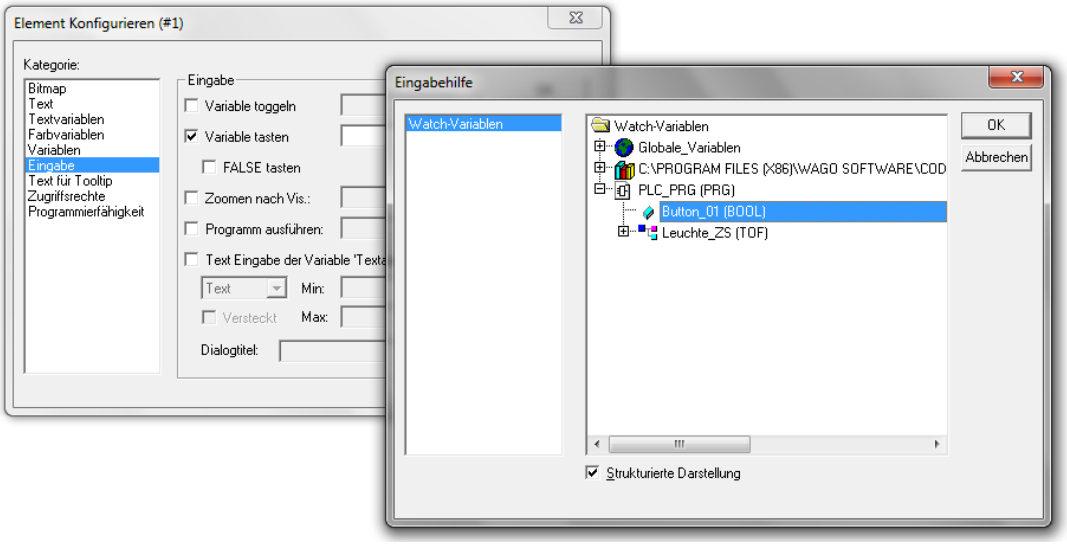


Abbildung 34: Variable zuordnen

Dem Kreis (Element #2) ist ebenfalls eine Variable zuzuweisen – in diesem Fall der Ausgangsvariablen „Zustand_01“ des Programms. Das geschieht ebenso über Doppelklick auf den Kreis und Auswahl des Unterpunkts „Variablen“. Dort wird unter „Farbwechsel“ über F2 die Variable „Zustand_01“ ausgewählt, denn diese soll die Basis für einen Farbwechsel des Kreises sein.

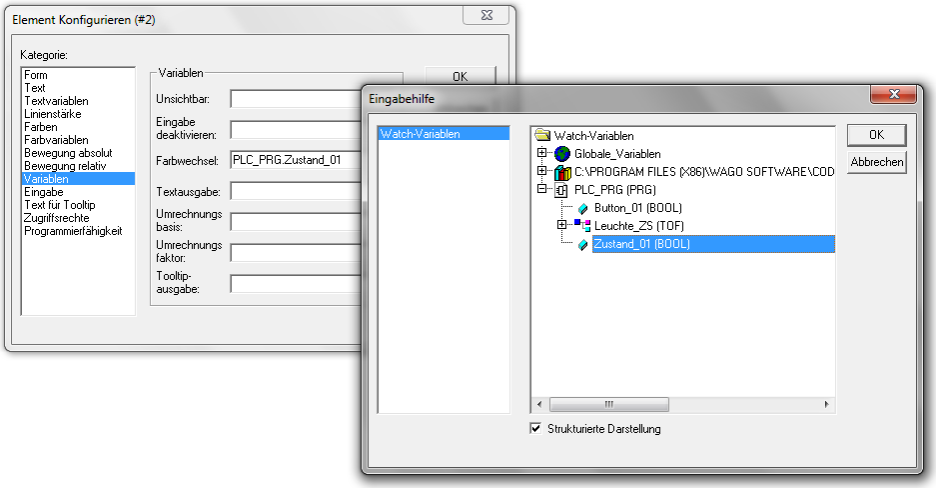


Abbildung 35: Variable zuweisen

Zum Schluss müssen für den Kreis noch die beiden Farben festgelegt werden, die je nach Zustand der ausgewählten Variablen angezeigt werden sollen. Dazu wird zunächst unter „Farben“ die Farbe für „Innen“ festgelegt (hier: Schwarz). Dies ist die Standardfarbe im „Ruhezustand“, d.h. immer dann wenn die Variable „.Status“ den Wert FALSE einnimmt. Dann wird die Farbe für „Alarmfarbe – Innen“ festgelegt (hier: Gelb).Dies ist der Zustand, der genau dann vorliegt, wenn die mit dem Element verknüpften Variable den Zustand TRUE einnimmt.

Wenn nun das System gestartet wird, kann die Ansteuerung des Relais sowohl über die Taster als auch über die Schaltflächen der Visualisierung durchgeführt werden. Parallel zum Relais wechselt die Innenfarbe des Kreises von Schwarz auf Gelb.

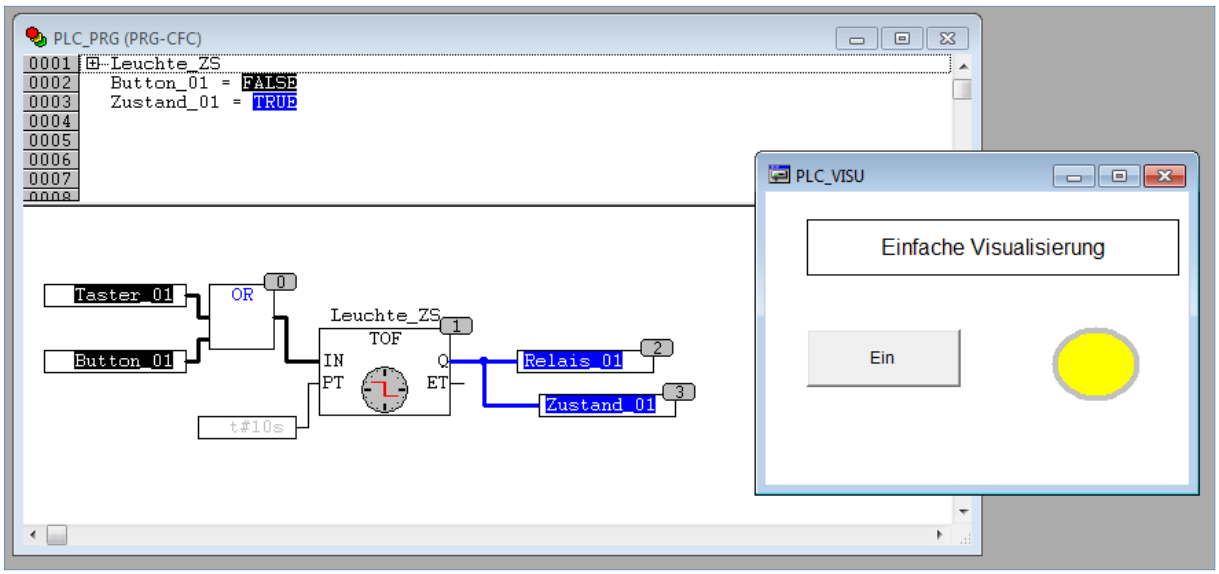


Abbildung 36: Visualisierung mit Farbfestlegung

7.7.3 Web-Visualisierung

Eine Visualisierung unter CoDeSys kann auch direkt in den Controller übertragen werden und von jedem gängigen HTML-Browser aufgerufen werden. Dazu ist in CoDeSys im Reiter „Ressourcen“ der Unterpunkt „Zielsystem“ aufzurufen. Dort ist wiederum der Reiter „Visualisierung“ anzuwählen und es ist ein Häkchen beim Unterpunkt „Web-Visualisierung“ zu setzen. Um die Ressourcen des Controllers zu sparen, sollte auch der Punkt „Komprimierung“ angewählt werden. Dies bedeutet, dass die Daten der Visualisierung komprimiert übertragen werden, was meist einen Geschwindigkeitsvorteil bei der Übertragung bedeutet. Manchmal kommt es bei der Übertragung zu einem Fehler – dieser kann ignoriert werden.

In diesem Dialog, wie im Folgenden ersichtlich, kann notfalls auch die Voreinstellung Größe von 800x600 Pixel geändert werden.

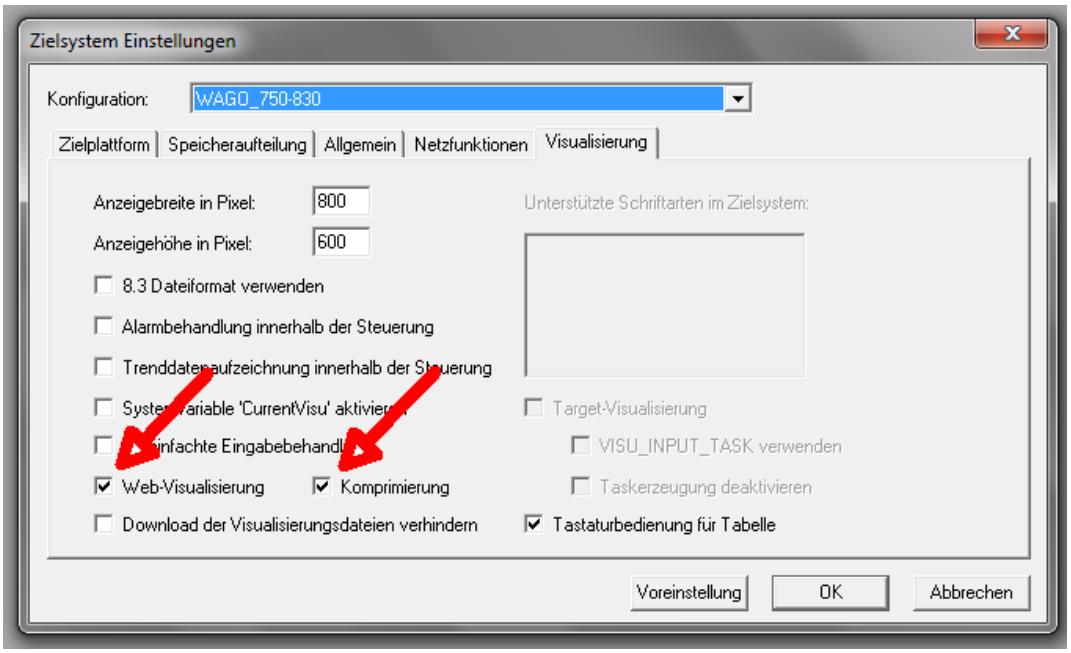


Abbildung 37: Anlegen einer Web-Visualisierung

Es erscheint eine **Warnmeldung**, dass durch Aktivieren der Web-Visualisierung jeder Login einen vorherigen Download erfordert. Diese ist zu bestätigen! → „Ja“

Wenn nun der Login (inkl. Download) und Start des Programms erfolgt, kann über einen normalen Browser die Web-Visualisierung genutzt werden – d.h. zur Anzeige von Status als auch Betätigung der Schaltflächen. Im Detail erfolgt der Aufruf über Angabe der IP-Adresse samt „/plc/webvisu.htm“ (z.B. „192.168.0.100/plc/webvisu.htm“). Alternativ kann einfach die IP-Adresse des Controllers angegeben werden, um die Hauptseite des Controllers anzuzeigen - dort kann der Unterpunkt „Visualisierung“ aufgerufen werden.

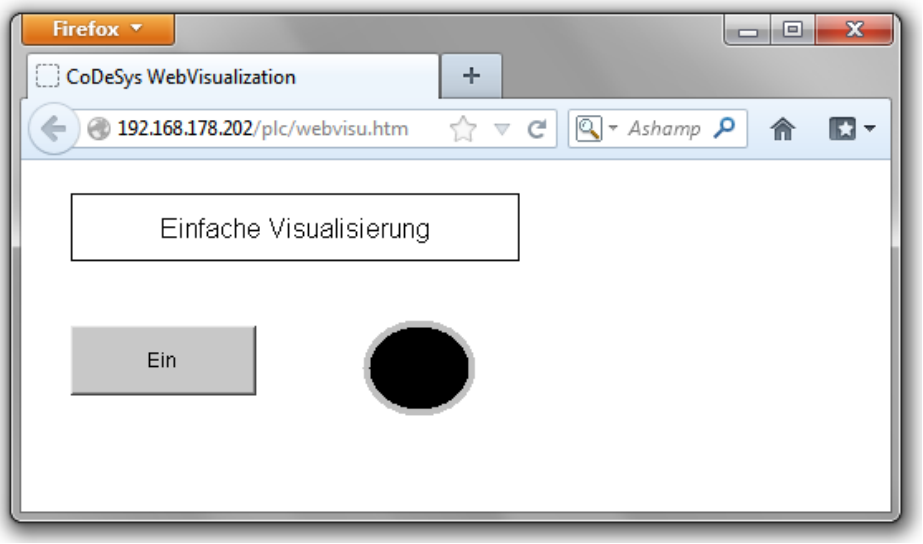


Abbildung 38: Web-Visualisierung über Java-Applet

Die Web-Visualisierung wird über ein Java-Applet abgebildet, welches alle 200 ms neu geladen wird.

Achtung/Hinweis: Wenn die Web-Visualisierung verändert wird, werden die Änderungen oft NICHT erkannt und es erfolgt kein automatisches Update in den Controller. Um dies zu erzwingen muss unter „Projekt“ der Menüpunkt „Alles bereinigen“ ausgeführt werden. Dies erzwingt, dass beim nächsten Login in den Controller die Web-Visualisierung ebenso übertragen wird. Das ist nicht nötig, wenn seit dem letzten Übertragen der Visualisierung eine Variabel eingefügt oder gelöscht wurde.

Hinweis: die bis hierher durchgeführte Konfiguration wurde auch als „EinfacheDemo04-Visualisierung.pro“ gespeichert.

7.8 Erweiterte Visualisierung (optional)

In der Visualisierung soll über einen Zeiger die verbleibende Restzeit in Prozent angezeigt werden, d.h. ein Zeiger soll sich von einer 100%-Marke bis zu einer 0%-Marke verändern. Dazu wird eine Variable benötigt, die diesen Wert, d.h. einen Wert zwischen 100 und 0, berechnet. Dazu wird im CFC eine Variable vom Typ

REAL mit dem Namen „VergangeneZeitInProzent“ eingefügt. Diese wird über mathematische Glieder an den Ausgang des TOF-Timers sowie der Variablen für die Verzögerungszeit verbunden. Davor müssen die Variablen vom Typ TIME noch in den Typ REAL umgewandelt werden.

Im Detail wird der Wert der Variablen „VergangeneZeitInProzent „ wie folgt berechnet:

$$VergangeneZeitInProzent = 100 \cdot (tBeleuchtung - tElasped)/ tBeleuchtung$$

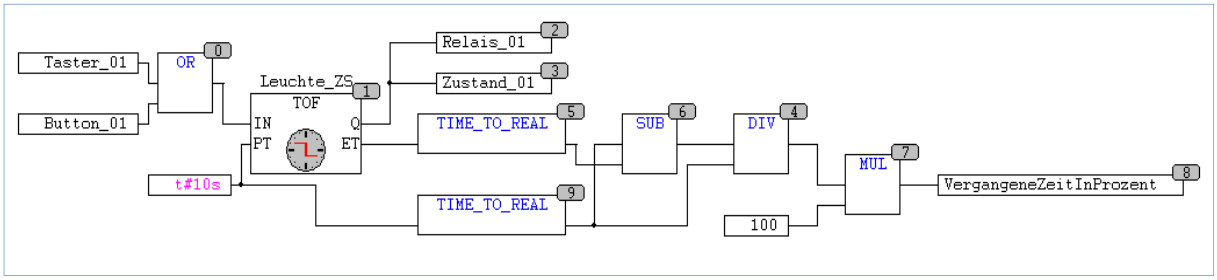


Abbildung 39: Visualisierung

Nun wird in der Web-Visualisierung noch ein Zeiger eingefügt und wie nachstehend gezeigt parametrisiert. Im Feld „Variable“ wird über „F2“ die Variable „VergangeneZeitInProzent“ ausgewählt.

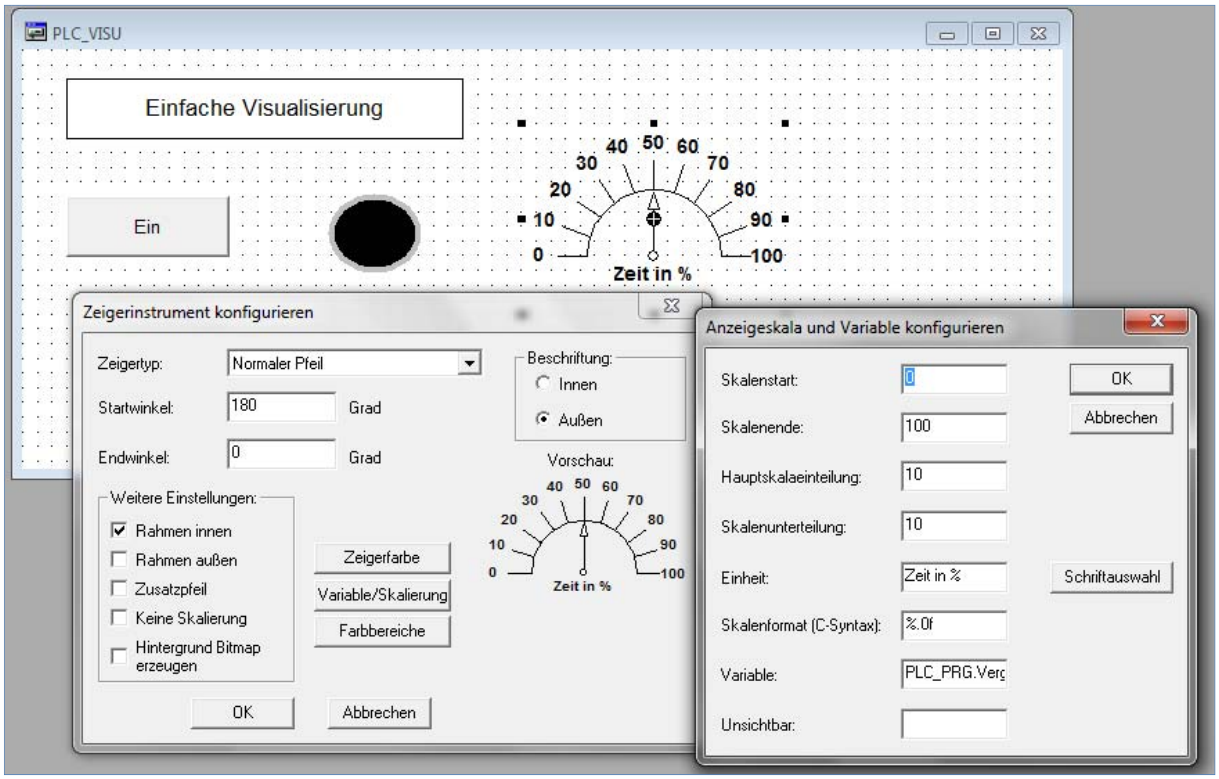


Abbildung 40: Zeigereinstellungen

Das Programm wird gestartet und der Zeiger bewegt sich nach dem Einschalten rückwärts von 100 auf 0.

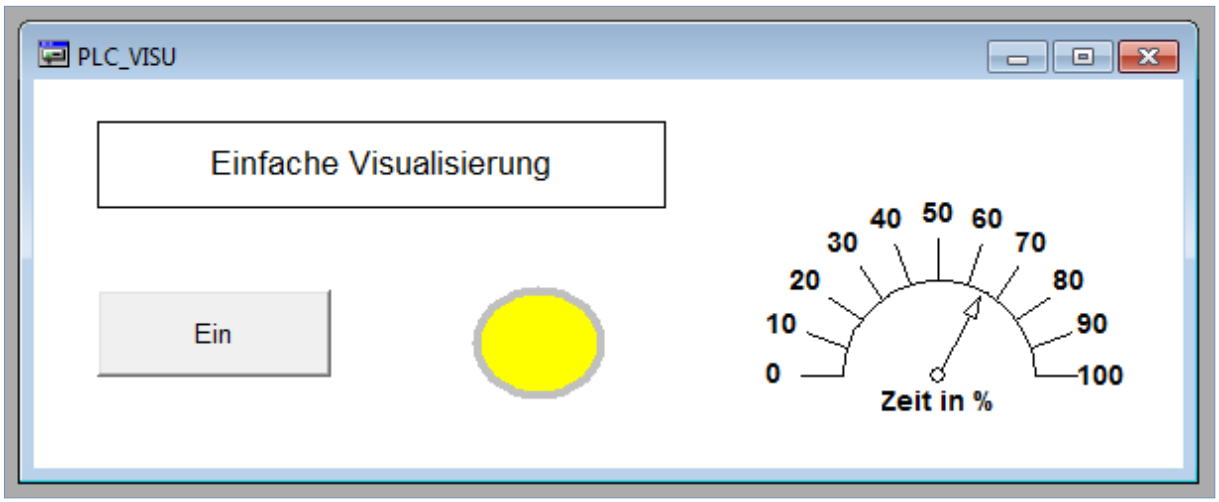


Abbildung 41: Visualisierung mit Zeigerdarstellung

Hinweis: die bis hierher durchgeführte Konfiguration wurde auch als „EinfacheDemo05-ErweiterteVisualisierung.pro“ gespeichert.

7.9 Gruppenschaltungen (optional)

Werden Gruppen- oder Zentralschaltungen benötigt, können diese über eine Kombination aus Bausteinen programmiert werden. Aufbauend auf 8.7 (Variablen und WEB-Visualisierung) sollen nun über die Weboberfläche beide Leuchten sowohl einzeln als auch als Gruppe geschaltet werden.

Dazu wird im Programm die Programmierung kopiert; zusätzlich erhalten die OR-Gatter einen weiteren Eingang (Anwahl des Gatters mit der rechten Maustaste; dann „Einfügen Bausteineingang“ oder Strg-U). Zusätzlich wird eine dritte Eingangsvariable „Button_03“ eingefügt. In der Visualisierung werden in Summe drei Buttons und zwei Kreise zur Visualisierung der beiden Leuchten eingefügt.

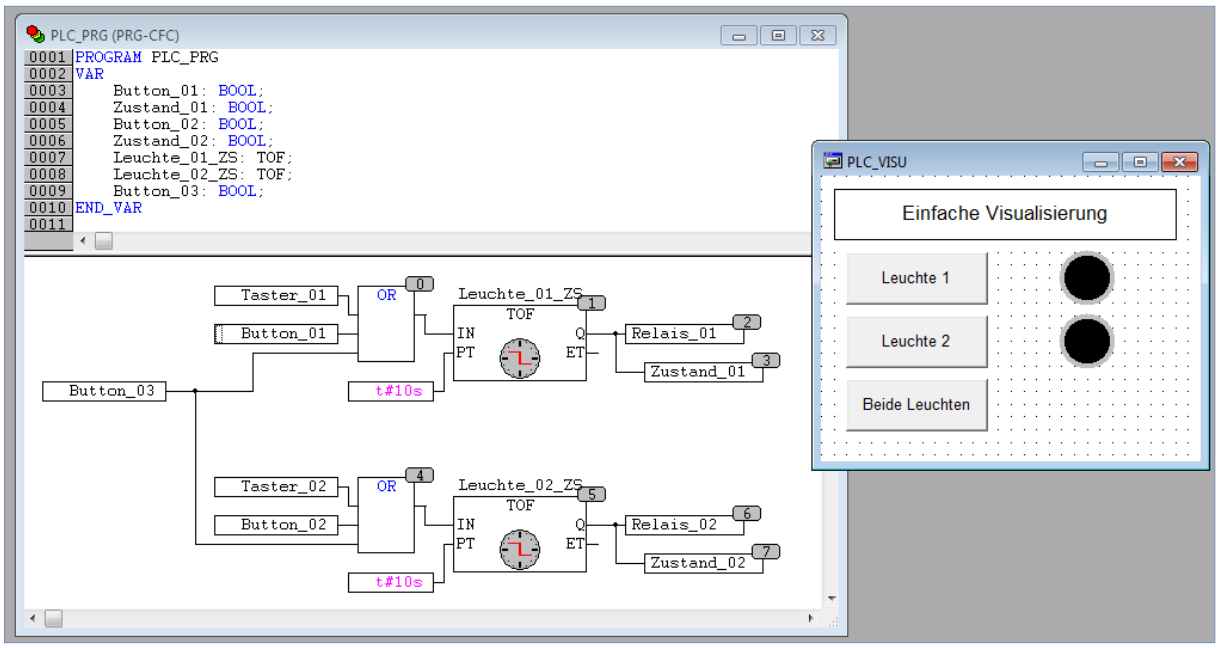


Abbildung 42: Visualisierung der Gruppenschaltung

Hinweis: die bis hierher durchgeführte Konfiguration wurde auch als „EinfacheDemo07-Gruppenschaltung.pro“ gespeichert.

7.10 Verwendung von Bibliotheksbausteinen (optional)

In diesem Beispiel soll exemplarisch die Verwendung von **leistungsfähigeren Bausteinen** aus der WAGO-Bibliothek gezeigt werden. Dazu wird für die Lichtsteuerung ein Baustein ausgewählt, der vor dem endgültigen Ausschalten eine Vorwarnung erzeugt. Konkret soll damit erreicht werden, dass nach Ablauf der Beleuchtungszeit das Licht erst kurz aus und wieder angeht, bevor es komplett ausgeschaltet wird. Dies hat in der Praxis den Vorteil, eine Vorwarnung zu erhalten und über rechtzeitiges Drücken des Tasters die Beleuchtungszeit erneut zu verlängern.

Im Grunde wird auf die Schaltung von 8.7 (Variablen und WEB-Visualisierung) aufgebaut. Statt dem einfachen Timer TOF soll der Baustein „Fb_Treppe2“ aus der WAGO-Bibliothek „Gebäude-Allgemein“ verwendet werden. Zunächst muss die entsprechende Bibliotheksdatei „Gebaeude_allgemein“ im Verzeichnis „*C:\Programme\WAGO Software\CoDeSys V2.3\Targets\WAGO\Libraries\Building*“ vorliegen (siehe Readme.txt der Bibliotheksdateien).

Die Funktion sowie das Verhalten ist der Dokumentation wie folgt zu entnehmen:

Tutorial: WAGO

Funktionsbeschreibung:

Bei einer steigenden Flanke am Eingang ***"xTaster"*** wird der Ausgang ***"xAktor"*** auf 1 gesetzt. Nach Ablauf der einstellbaren Treppenhauszeit ***"dwT_10tel_s"*** erfolgt eine Ausschaltvorwarnung, indem der Ausgang für 1 s auf 0 zurückgesetzt wird. Anschließend wird die Ausgang für die Vorwarndauer ***"dwTv_10te_s"*** eingeschaltet. Die Treppenhauszeit kann durch eine steigende Flanke am Eingang ***"xTaster"*** jederzeit retriggert werden. Wenn die Zeit nicht retriggert wird, dann schaltet der Ausgang nach Ablauf der Vorwarnzeit ***"dwTv_10tel_s"*** auf 0 zurück. Solange am Eingang ***"xHand"*** ein EIN - Signal anliegt, wird der Ausgang gesetzt (Dauerlicht).

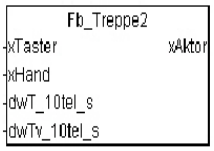
WAGO-I/O-*PRO CAA* Elemente der Bibliothek

Kategorie:	Gebäudetechnik		
Name:	Fb_Treppe2		
Typ:	Funktion ☐	Funktionsblock ☒	Programm ☐
Name der Bibliothek:	Gebaeude_allgemein.lib		
Anwendbar für:	Alle programmierbaren Feldbus-Controller		

Eingangsparameter:	**Datentyp:**	**Kommentar:**
xTaster	BOOL	Eingang für Tastsignal
xHand	BOOL	Schaltbefehl Dauerlicht
dwT_10tel_s	DWORD	Treppenhauszeit Wertebereich 10 – 65535 [0,1 s] Voreinstellung = 1200
dwTv_10tel_s	DWORD	Vorwarnzeit 50 – 300 [0,1 s] Voreinstellung = 150

Rückgabewert:	**Datentyp:**	**Kommentar:**
xAktor	BOOL	Ausgangsschaltsignal

Grafische Darstellung:



Tutorial: WAGO

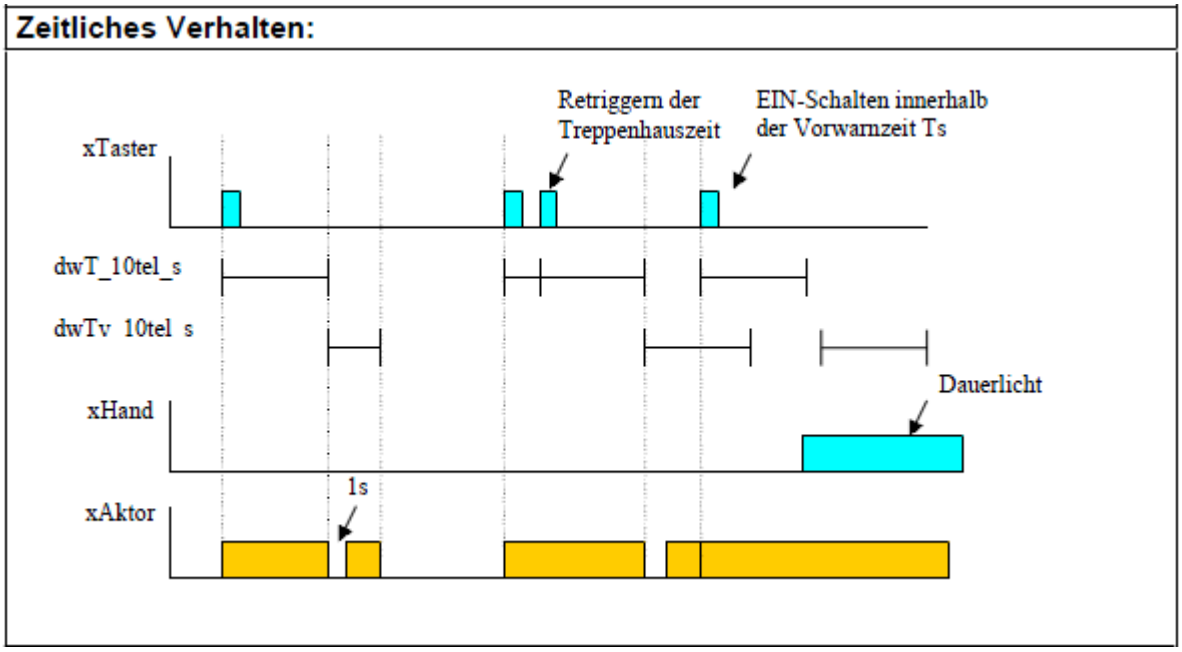


Abbildung 43: Zeitliches Verhalten

Unter CoDeSys ist diese Bibliothek einzubinden, was über die Auswahl von „Ressourcen“ – „Bibliotheksverwalter“ – „Kontextmenü – Weitere Bibliothek“ gestartet wird. Es öffnet sich ein Datei-Dialog, über den das Verzeichnis gewechselt und die oben erwähnte Bibliotheksdatei ausgewählt werden kann.

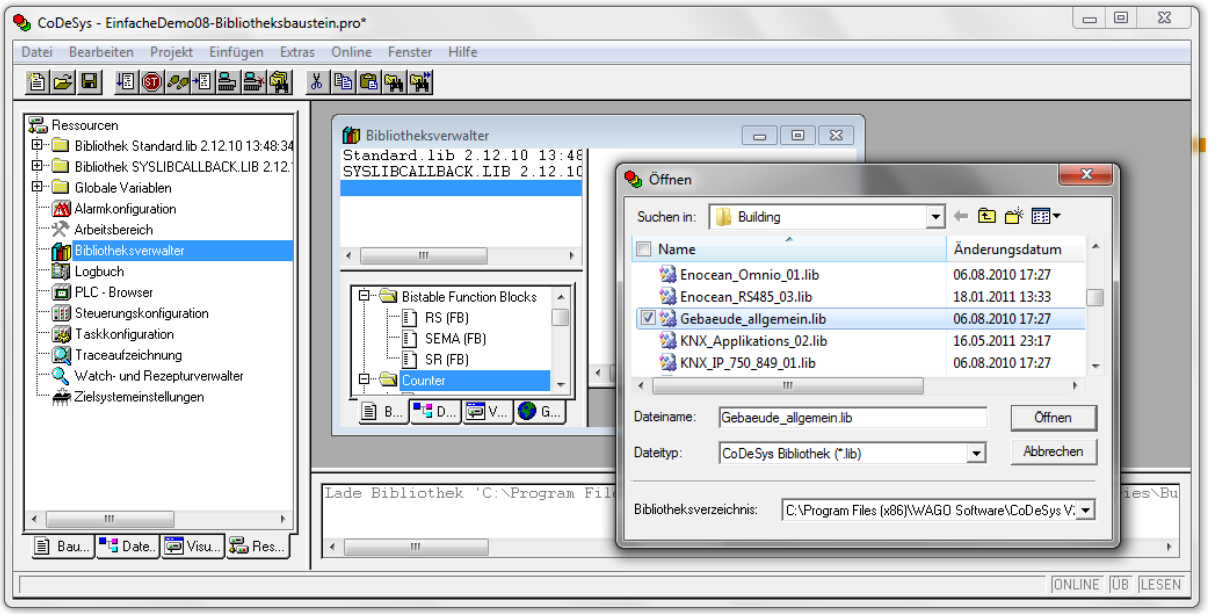


Abbildung 44: Bibliothek einbinden

Im CFC wird der Timer gelöscht und ein neuer Baustein eingefügt sowie in den Baustein „Fb_Treppe2 (FB)“ umgewandelt. Im Detail befindet sich dieser innerhalb der Eingabehilfe (F2) im Unterpunkt „GEBAUEDE_ALLGEMEIN.LIB“ und „Treppe“.

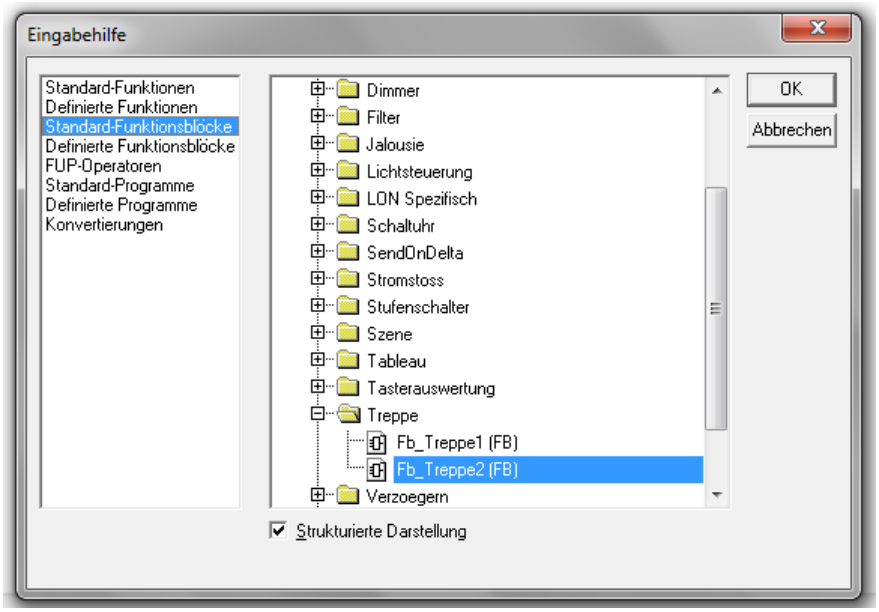


Abbildung 45: Baustein „Fb_Treppe2“ einfügen

Anschließend wird der Baustein wie folgt angeschlossen:

An den Eingang „xTaster“ wird der Ausgang des OR-Bausteins angeschlossen – d.h. entweder der echte Taster oder der Button der Visualisierung schaltet das Licht zunächst an. Der Eingang „xHand“ bleibt ohne Beschaltung. Gemäß Beschreibung könnte man über diesen Eingang den Baustein auf Dauerlicht schalten. Der Eingang „dwT_10tel_s“ gibt an, wie lange das Licht zunächst eingeschaltet wird (Wert im Bereich von 10 – 65535 und Angabe der Zeit in 1/10 Sekunden). Am Eingang „dwTv_10tel_s“ wird angegeben, wie lange die nachgeschaltete Verzögerung ist. Konkret geht das Licht nach Ablauf der ersten Zeit zunächst 1 Sekunde aus und geht dann wieder so lange an, wie es die zweite Zeit vorgibt. Erst nach Ablauf dieser Zeit, wird das Licht komplett ausgeschaltet. Sollte in der Zwischenzeit der Taster erneut betätigt werden, beginnt das Szenario von vorne.

Hinweis: Die bis hierher durchgeführte Konfiguration wurde auch als „EinfacheDemo08-Bibliotheksbaustein.pro“ gespeichert.

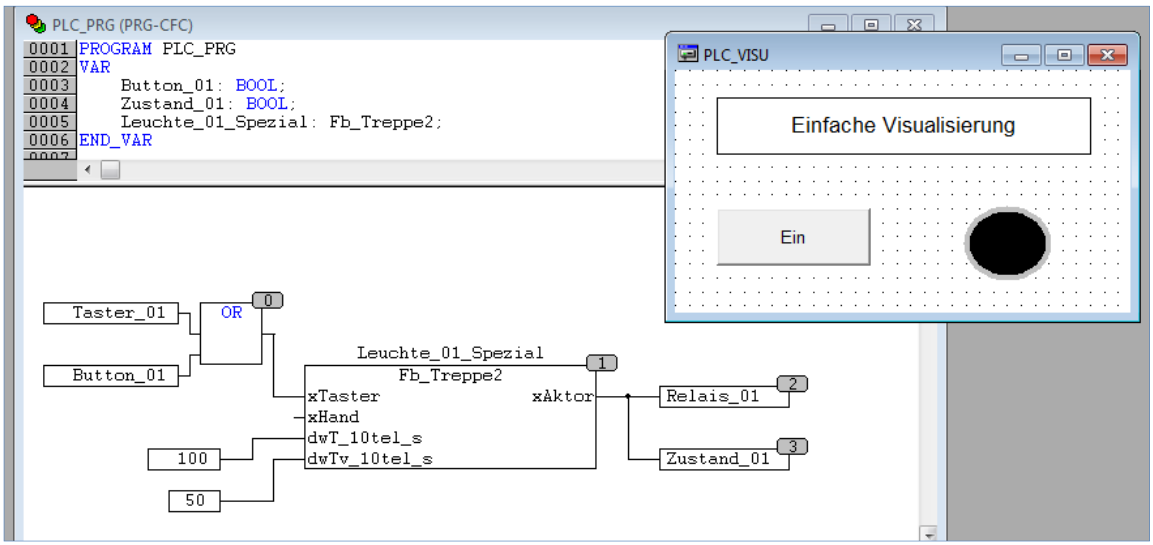


Abbildung 46: Schaltung mit nachgeschalteter Verzögerung

7.11 Systemlaufzeit überprüfen (optional)

Um eine korrekte Programmausführung zu ermöglichen, muss die Laufzeit des Programms kleiner als die Taktrate der Task sein. Ansonsten läuft zwar das Programm ohne offensichtlichen Fehler aber die korrekte Verarbeitung bzw. Ausgabe ist nicht gewährleistet.

Dazu ist in bei laufendem Projekt (d.h. „Online“ – „Einloggen“ und –„Start“ ausgeführt) unter „Ressourcen“ der Punkt „PLC-Browser“ auszuwählen. Dieser Punkt kann immer ausgewählt werden, d.h. erfordert nicht unbedingt das vorherige Anlegen einer Visualisierung wie im vorliegenden Beispiel.

Es öffnet sich ein leeres Fenster, welches oben mit einer Kommandozeile beginnt. Wenn dort das Wort „tsk“ eingegeben wird, werden die Taskzeiten (Min, Max, Average) angezeigt. Die Empfehlung ist, dass die Task-Zykluszeit mindestens den dreifachen Wert hat wie die „Average Cycletime“. Im vorliegenden Fall wurde bei der Task-Konfiguration ein Wert von 15 ms eingestellt und eine maximale Durchlaufzeit ergab sich zu 10 ms – der Durchschnitt liegt bei 4 ms.

Tutorial: WAGO

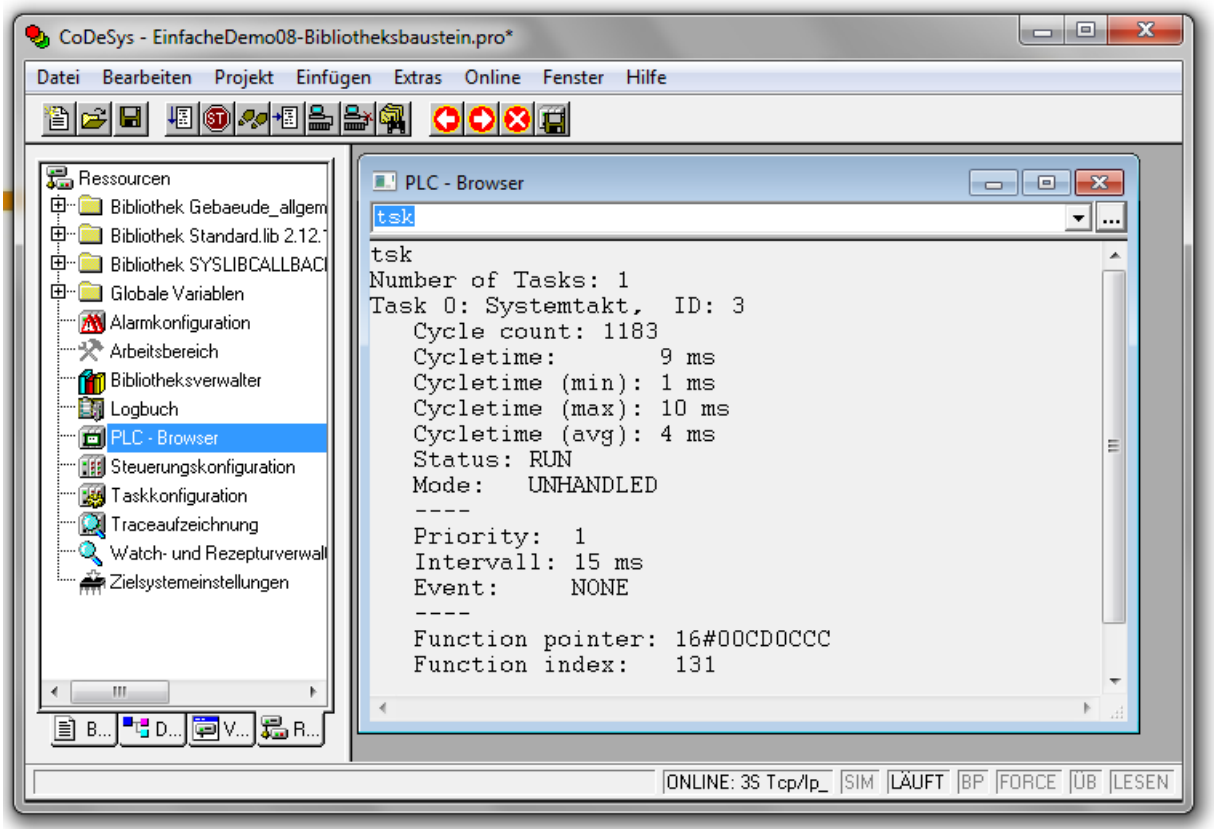


Abbildung 47: Systemlaufzeit überprüfen

Die Eingaben von „?“ in der Kommandozeile listet alle verfügbaren Kommandos inklusive kurzer Erklärung auf.

8 Integration EnOcean-Sensorik

8.1 HW-Ergänzung und Steuerungskonfiguration

Weiter aufbauend auf der einfachen Zeitschaltung im Kapitel „8.7 Variablen und WEB-Visualisierung“ soll zusätzlich ein EnOcean-Taster eingebunden werden.

Dazu muss zunächst der HW-Aufbau einen EnOcean Funkempfänger (Klemme -642) enthalten, d.h. dieser muss physikalisch eingesteckt sein und in der Steuerungskonfiguration eingetragen sein.

Im nächsten Schritt muss die WAGO-Bibliothek zum EnOcean-Funkempfänger geladen werden. Dies erfolgt ähnlich wie im Kapitel „8.10: Verwendung von Bibliotheksbausteinen“ beschrieben (d.h. Aufruf von „Ressourcen“, Bibliotheksverwalter“ und Einfügen einer neuen Bibliothek). Im Detail ist die Bibliothek „EnOcean_4.lib“ aus dem Verzeichnis „Building“ einzufügen.

8.2 Empfang der EnOcean Telegramme

Im Projekt ist nun ein Baustein einzufügen, der alle Funktelegramme von EnOcean bereitstellt. Dazu ist der Baustein „FbEnoceanReceive“ einzufügen. Diesem muss eine numerische Variable vorangestellt werden und über den Eingang „bModule“ verbunden werden. Dabei ist diese Zahl die Nummer des EnOcean Receivers am Controller (von links nach rechts durchgezählt). Bei nur einem EnOcean Receiver ist das damit immer die Zahl 1.

Am Ausgang wird eine Fehlermeldung ausgegeben, die zumindest in eine lokale Variable (Type Byte) münden sollte.

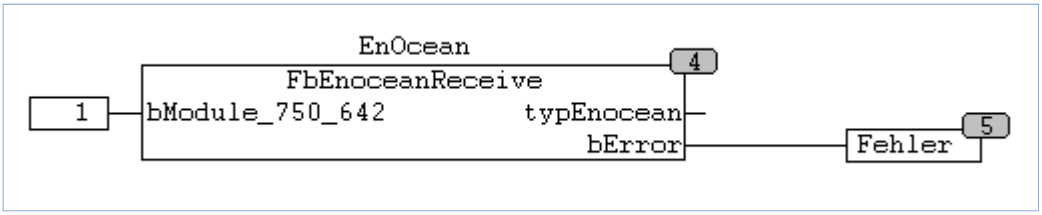


Abbildung 48: EnOcean in Visualisierung

8.3 Ermittlung der Sender-ID

Zur Ermittlung der Sender-ID eines Sensors ist der Baustein „FbShow_ID_By_Click“ einzufügen. Dieser erhält seine Daten vom Baustein „FbEnoceanReceive“. Über die zusätzlichen Parameter kann angegeben werden, was für eine Art Sensor ausgewertet wird (Standard-Einstellung „Piezo-basierte Sensoren“ und wie oft ein Taster gedrückt werden muss (Standard-Einstellung: Doppelklick).

Der Ausgang ist eine Variable vom Typ DWORD zu geben. Nach dem Start (Online, sowie Start) kann über CoDeSys die ID des Sensors abgeschrieben werden, der zu diesem Moment doppelt gedrückt werden muss.

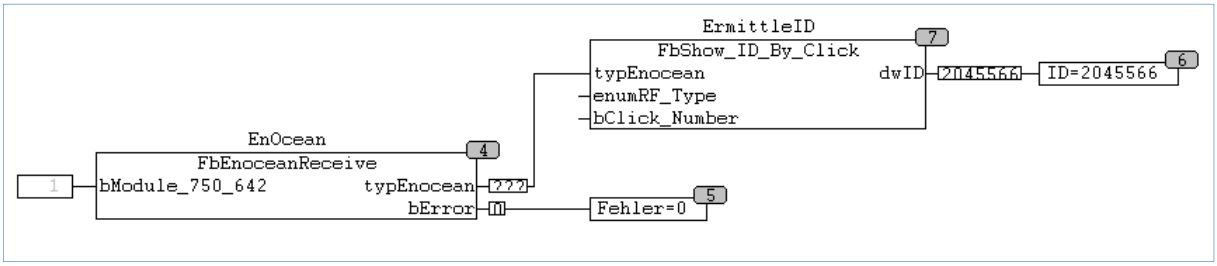


Abbildung 49: Baustein „FbShow_ID_By_Click“

Hinweis: die bis hierher durchgeführte Konfiguration wurde auch als „EnOcean01-ErmittlungID.pro“ gespeichert.

8.4 Einbinden eines Tasters

Im Folgenden wird genau der Taster eingebunden, dessen ID im vorgehenden Schritt ermittelt wurde. Dazu ist sowohl das EnOcean Telegramm als auch die ID an einen Baustein „FbButton_4_Channel“ zu geben, da im vorliegenden Fall der Taster ein 4-fach Taster ist.

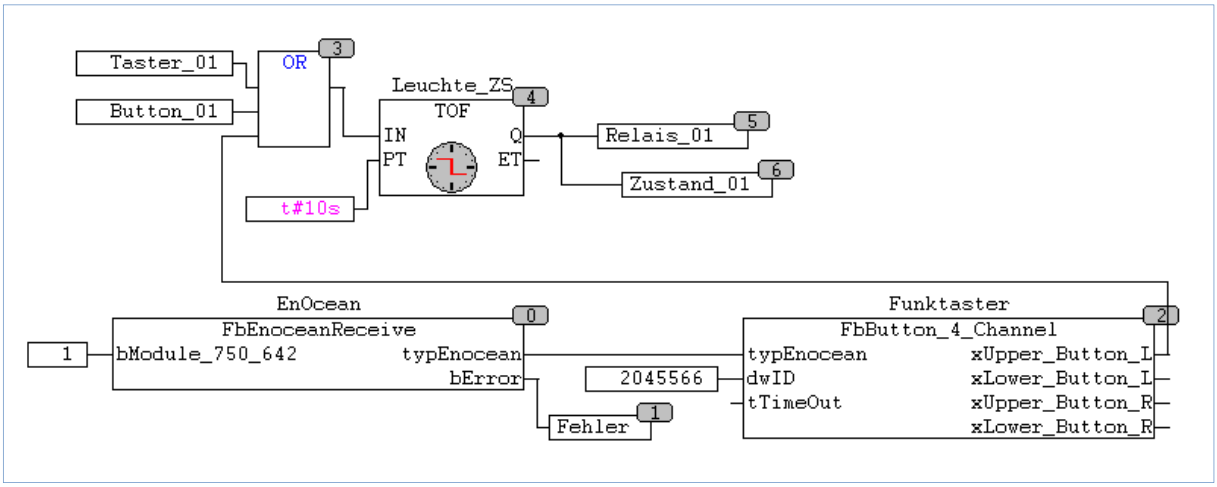


Abbildung 50: Schaltbild mit EnOcean Taster

Der neue Baustein erzeugt vier binäre Signale, die wie jedes andere binäre Signal weiterverarbeitet werden können. Im vorliegenden Fall wird eines davon auf das (erweiterte) OR-Gatter zum Ansteuern des Timers gegeben.