Einführung in den Codesys Visualisierungs-manager

ZANDER AACHEN

Handbuch zur Programmierung des MVisio HMI und des MVisio HMI Lite

Einführung in Codesys mit dem MVisio HMI als PROFINET-Master in Kombination mit der High-Speed SPS ZX20TP

Zander GmbH & Co. KG
Am Gut Wolf 15
52070 Aachen, Deutschland
info@zander-aachen.de
www.zander-aachen.de

Teile-Nr.: E61-360-00
Ausgabe: L02

Dieses Dokument ist das Originaldokument.

Technische Änderungen vorbehalten,
alle Angaben ohne Gewähr.

Inhaltsverzeichnis:

1 Zu diesem Dokument

Dieses Dokument soll den Einstieg in die Programmierplattform Codesys erleichtern. Dabei werden lediglich ausgewählte Teilaspekte der Codesyssoftware beleuchtet. Weitere Informationen zu Codesys sind auf der Webseite der 3S-Smart Software Solutions GmbH zu finden.

Bevor die im diesem Dokument dargestellte Beispielapplikation programmiert werden kann, ist die Installation eines Packages, welches eine Verbindung zwischen Codesys und dem HMI ermöglicht durchzuführen. Eine technisch aktuelle Installationsanleitung finden Sie unter www.zander-aachen.de.

Die folgenden Beispiele wurde mit dem MVisio HMI und der High-Speed Steuerung ZX20TP ausgeführt. Eine Umsetzung mit dem MVisio HMI Lite ist analog möglich. Dabei wurde die Codesys Version V3.5 SP12 verwendet.

2 Das MVisio HMI als PROFINET-Master betreiben

Der Quick Installation Guide soll einen schnellen Einstieg in die Welt der Visualisierung mit dem MVisio HMI ermöglichen. Dabei steht der Verbindungs- und Installationsaufbau im Vordergrund.

2.1 Aufbau und Anschließen

Für das folgende Beispielprogramm empfiehlt es sich das MVISIO HMI, die ZX20TP und den PC wie in Abbildung 1 zu verbinden.

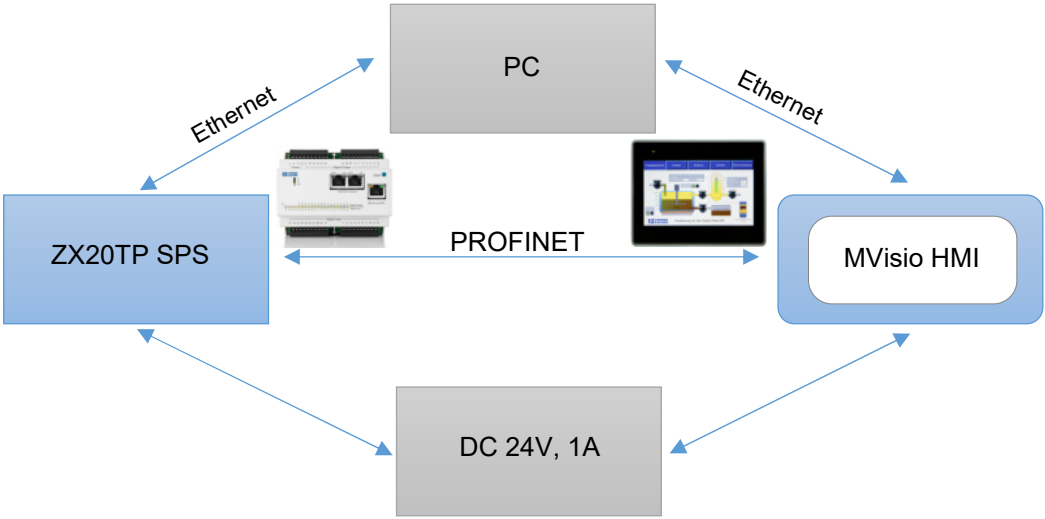
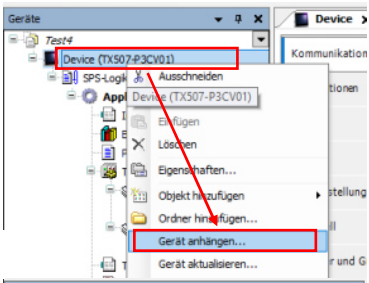


Abbildung 1: Systemskizze des Aufbaus zur Programmierung des MVisio HMI und der ZX20TP SPS

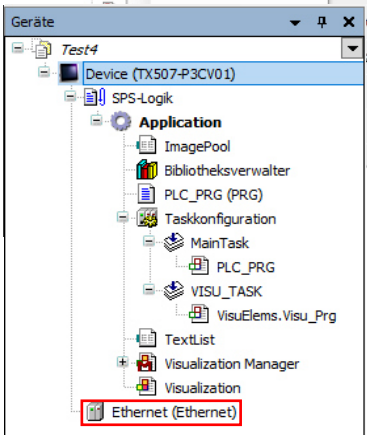
2.2 MVisio HMI Ethernet-Schnittstelle konfigurieren

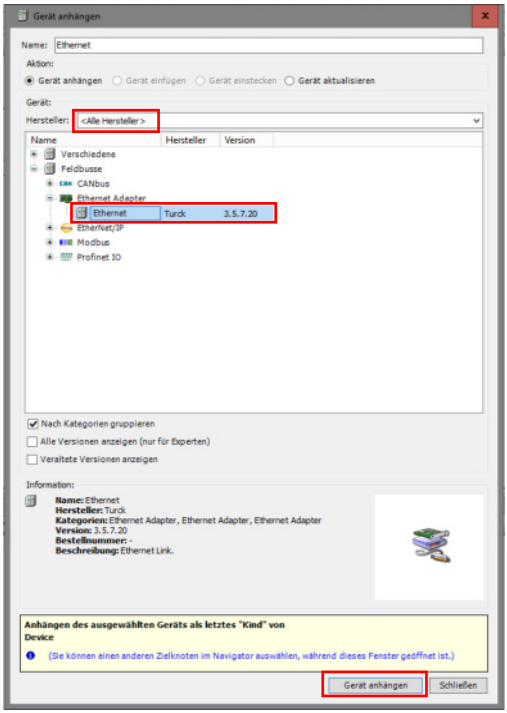
Im Gerätebaum mit einem Rechtsklick auf *Device (TX507-P3CV01)* klicken und *Gerät anhängen* auswählen, siehe Abbildung 2 a). In dem neuen Fenster bei Hersteller *<Alle Hersteller>* auswählen und dann unter *Feldbusse - Ethernet Adapter - Ethernet* auswählen, siehe Abbildung 2 b). Durch *Gerät anhängen* bestätigen. Danach erscheint der Ethernet-Port im Gerätebaum, siehe Abbildung 2 c). Um die IP Adresse zu konfigurieren wird auf den Ethernet-Port im Gerätebaum ein Doppelklick ausgeführt (siehe Abbildung 3 a) und dann auf *Netzwerkschnittstelle* geklickt, siehe Abbildung 3 b). Aus der Liste die MVisio HMI IP Adresse auswählen und über *OK* bestätigen, siehe Abbildung 3 c). Eine IP Adresse wird dem MVisio HMI im Installationsguide zugewiesen.

a)



c)





b)

Abbildung 2: Einbinden der Ethernetschnittstelle

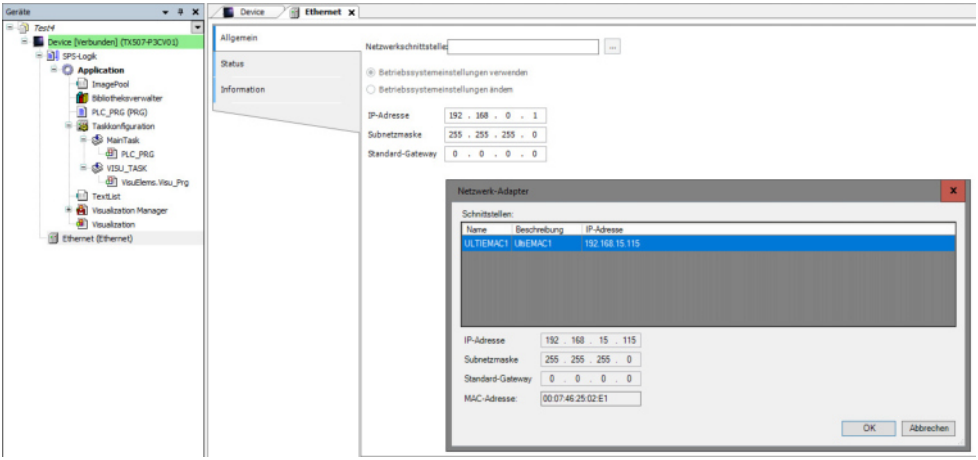


Abbildung 3: Konfiguration der IP Adresse

2.3 PROFINET-Master einbinden

Das MVisio HMI kommuniziert mit der ZX20 SPS über eine PROFINET-Schnittstelle. Diese wird im Folgenden konfiguriert. Zunächst muss der PROFINET-Controller eingebunden werden. Im Gerätebaum einen Rechtsklick auf *Ethernet* ausführen und dann auf *Gerät anhängen* klicken, siehe Abbildung 4 a). In dem Feld Hersteller *<Alle Hersteller>* angeben. Unter *Feldbusse - Profinet IO - Profinet IO Master - PN-Controller* auswählen und auf *Gerät anhängen* klicken, siehe Abbildung 4 b). Im Gerätebaum sollte unter dem Reiter Ethernet nun der PN-Controller erscheinen, siehe Abbildung 5.

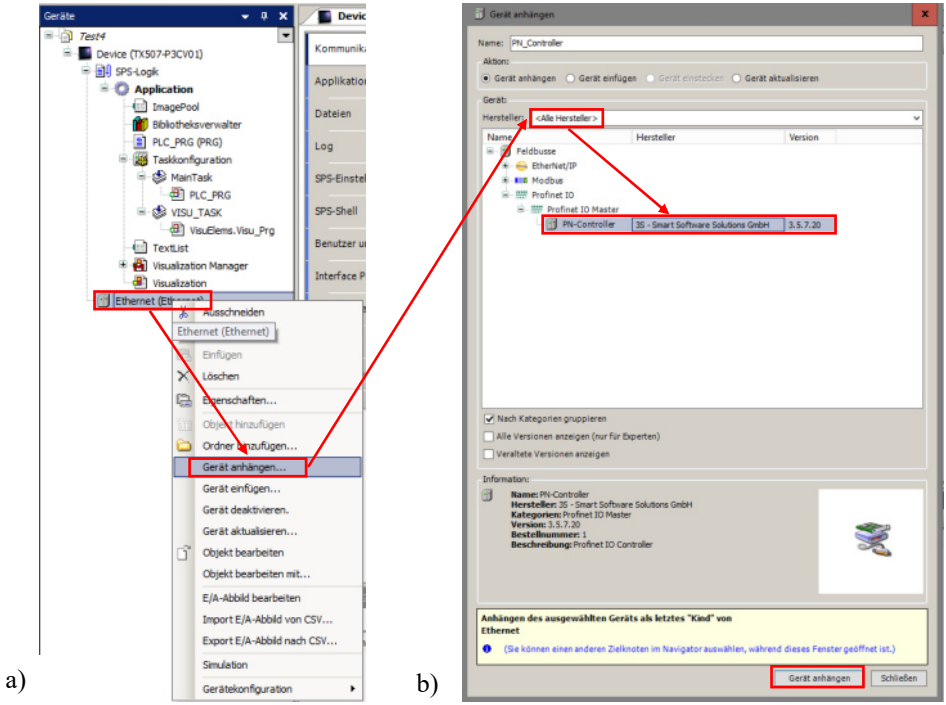


Abbildung 4: PN-Controller anhängen

Das PN-Controllerfenster mit einem Doppelklick auf *PN-Controller* im Gerätebaum öffnen, siehe Abbildung 5. In diesem Fenster den Reiter *Allgemein* auswählen.

Wichtig: Die Subnetzmaske des MVISIO HMI muss die gleiche Adresse aufweisen die Subnetzmaske des PN-Controllers. Die Subnetmaske des MVISIO HMI wird bei Gerätestart angezeigt.

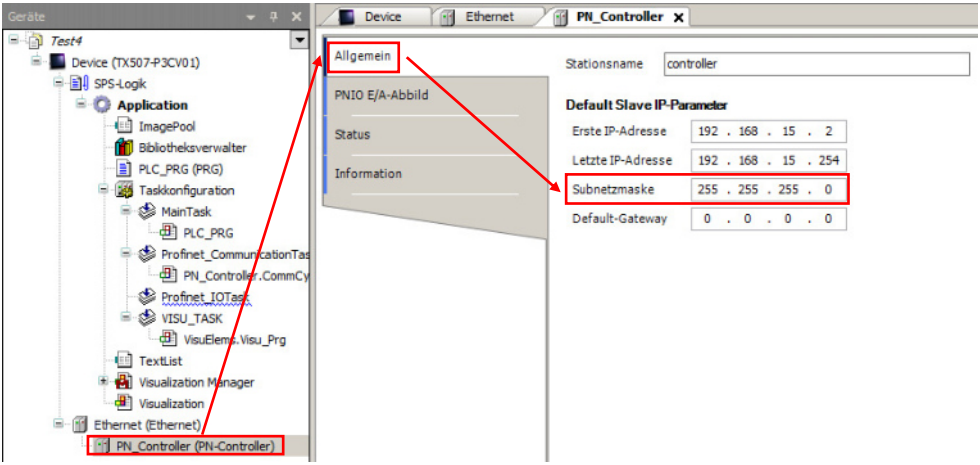


Abbildung 5: Überprüfung der Subnetzmaske

Im Folgenden wird die GSDML-Datei installiert. Dazu auf *Tools - Geräte-Repository* klicken, siehe Abbildung 6 a). In dem sich öffnenden Fenster unter *Feldbusse - Profinet IO - Profinet IO Master - PN Controller* auswählen, siehe Abbildung 6 b). Dann auf *Installieren* drücken und die gewünschte GSDML-Datei auswählen. Nach der Installation das Fenster schließen.

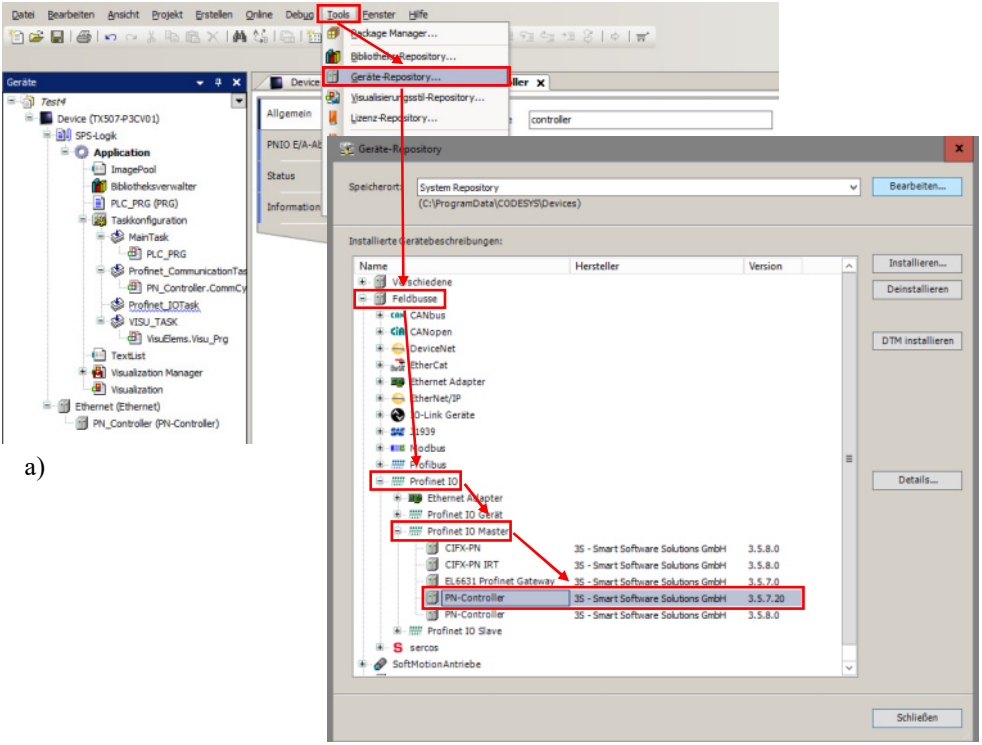


Abbildung 6: GSDML-Datei einbinden

Einen Rechtsklick auf *PN_Controller* im Gerätebaum ausführen und *Gerät anhängen* auswählen, siehe Abbildung 7 a). Bei Hersteller: *<Alle Hersteller>* auswählen und unter *Feldbuss - Profinet IO - Profinet IO Slave - COMX 100XX-RE/PNS V3.4.19 --V3.4.x* des Herstellers *Hilscher Gesellschaft für Systemautomation mbH* auswählen, siehe Abbildung 7 b). Dann auf *Gerät anhängen* und danach auf *Schließen* klicken. Im Gerätebaum sollte unter *PN_Controller* die Schnittstelle *COMX* erscheinen, siehe Abbildung 7 c).

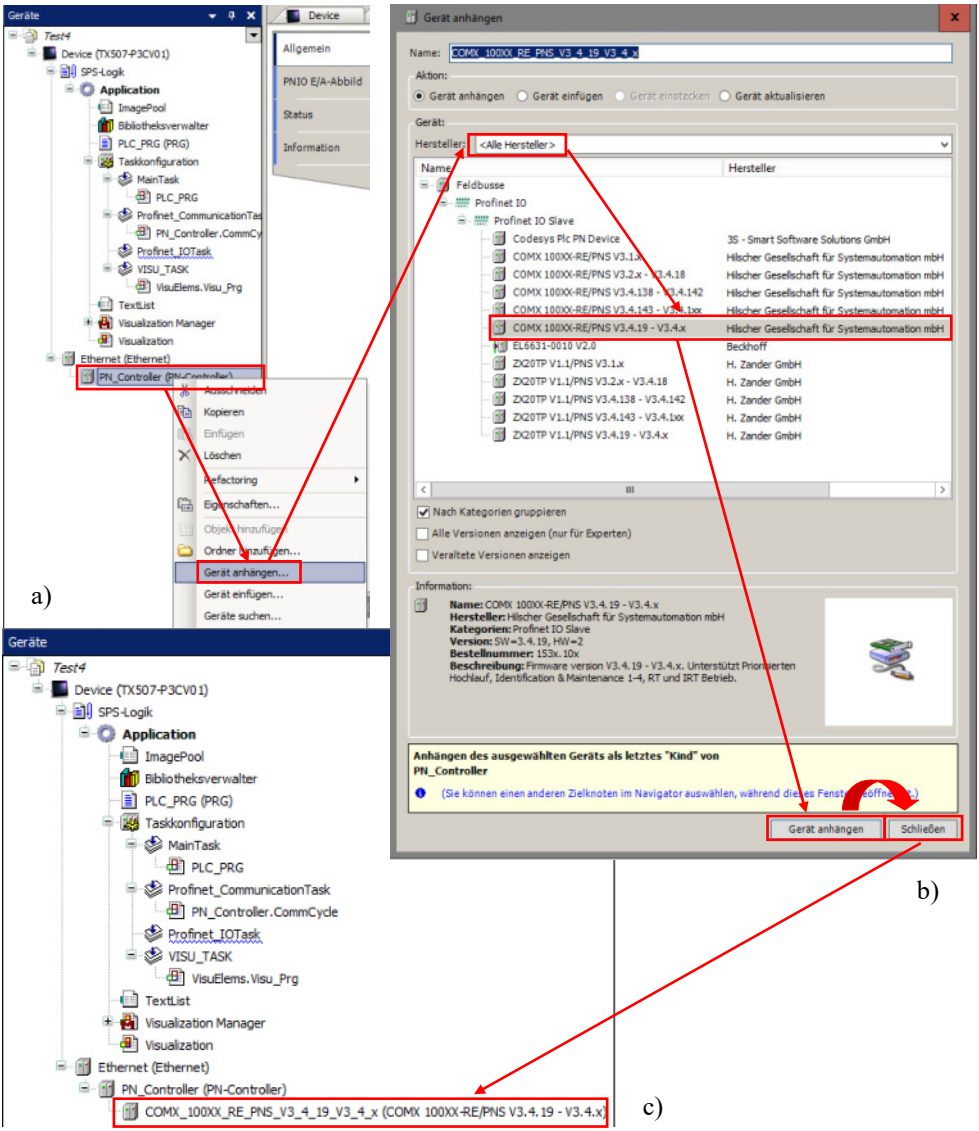
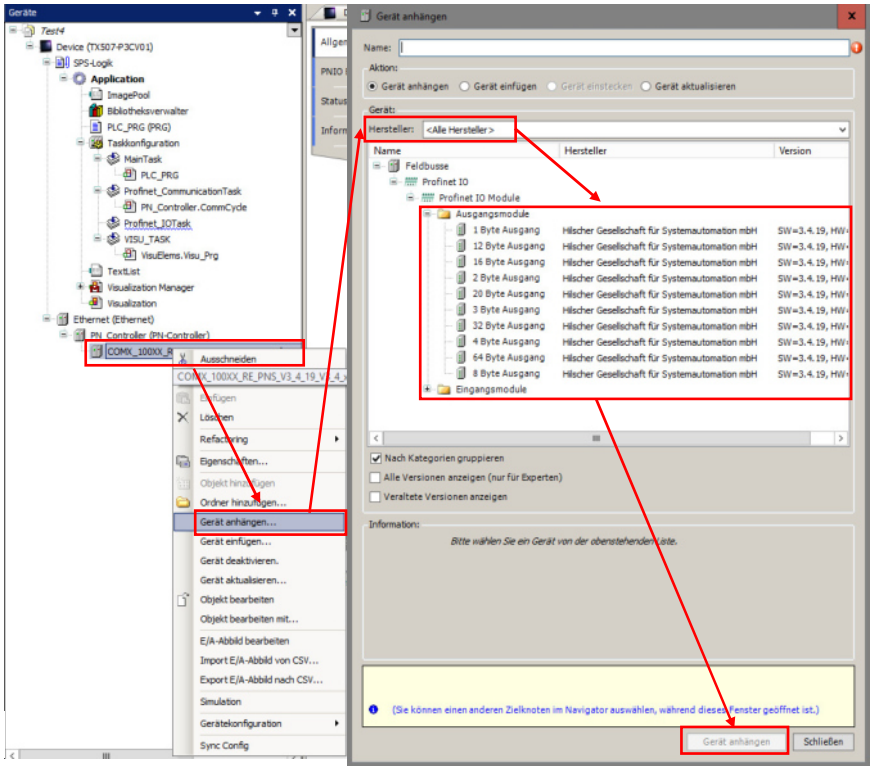


Abbildung 7: COMX Schnittstelle

Zuletzt müssen die Ein- und Ausgänge der PROFINET-Schnittstelle eingebunden werden. Dazu mit einem Rechtsklick im Gerätebaum auf *COMX* klicken und dann *Gerät anhängen* auswählen, siehe Abbildung 8 a). In dem aufgehenden Fenster bei Hersteller auf *<Alle Hersteller>* klicken. Unter *Feldbusse - Profinet IO - Profinet IO Module - Ausgangsmodul* bzw. *Eingangsmodul* können nun Ein- und Ausgänge gewählt werden, siehe Abbildung 8 b). Durch mehrfaches Klicken auf *Gerät hinzufügen* können mehrere Ein- und Ausgänge hinzugefügt werden. Um eine Kommunikation mit der ZX20 herzustellen, müssen folgende Ein- bzw. Ausgänge konfiguriert werden: 4 Ausgänge und 1 Eingang. Dabei ist die Reihenfolge der Ein- und Ausgänge zu beachten, siehe Abbildung 9 b) im Gerätebaum.

a)



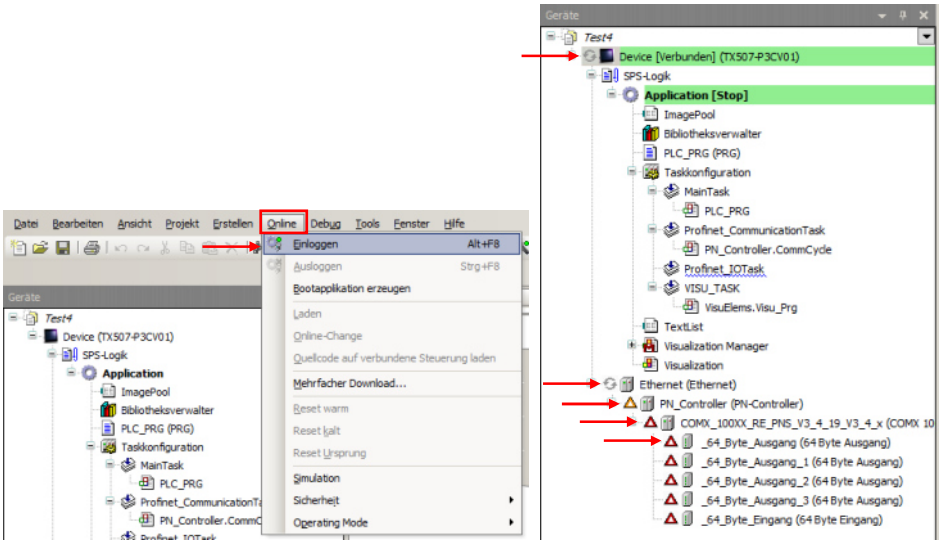
b)

Abbildung 8: Einbindung der PROFINET Ein- und Ausgänge

2.4 Netzwerktest

Im Folgenden soll ein kurzer Netzwerktest durchgeführt werden um sicherzustellen, dass die Geräte richtig mit einander verbunden sind. Dazu zunächst unter *Online* auf *Einloggen* klicken, siehe Abbildung 9 a). Daraufhin erscheint vor dem Device und dem Ethernet ein grauer Punkt und vor dem PN-Controller, COMX und den Ein- und Ausgängen verschieden farbige Dreiecke, siehe Abbildung 9 b).

a)



b)

Abbildung 9: Kommunikationstest zwischen HMI und SPS

Um den Test zu starten unter *Debug - Start* auswählen, siehe Abbildung 10 a). Wechseln die zuvor genannten Symbole (Punkte und Dreieck) auf grüne Symbole war der Test erfolgreich, siehe Abbildung 10 b). Um den Test zu beenden klicken Sie unter *Debug* auf *Stopp* und unter *Online* auf *Ausloggen.*

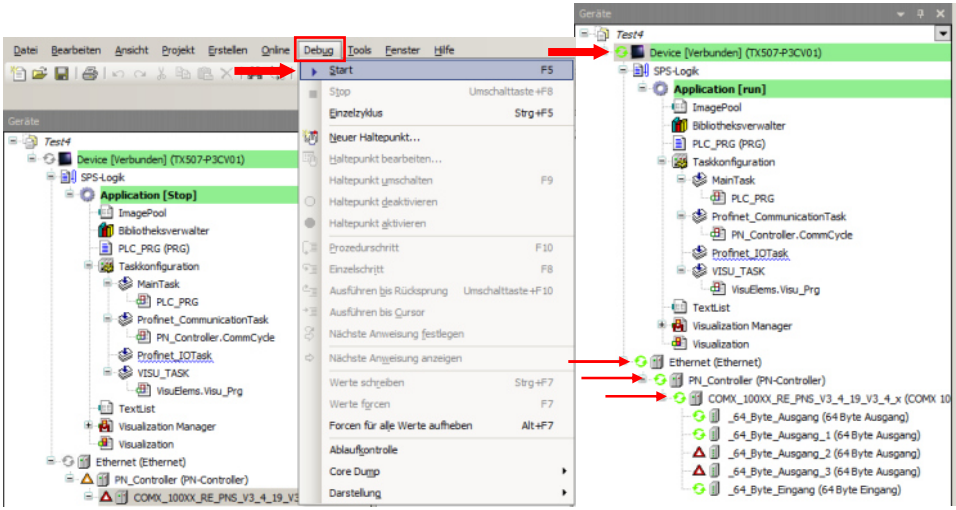


Abbildung 10: Erfolgreicher Netzwerktest

3. Einführung in die Oberflächenprogrammierung des HMI mit Codesys

Bevor eine Oberfläche in Codesys erzeugt wird, ist es notwendig das MVisio HMI und die SPS ZX20 richtig miteinander zu verbinden. Befolgen Sie hierfür schrittweise die Anweisungen des Kapitels 2. Nach erfolgreichem Abschluss der Installation sollte der Gerätebaum wie in Abbildung 11 dargestellt sichtbar sein.

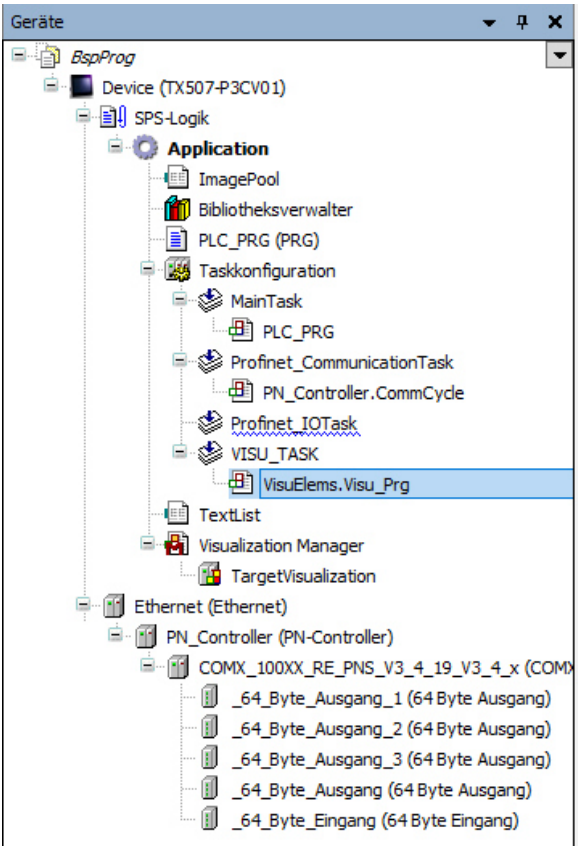


Abbildung 11: Gerätebaum nach Installationsabschluss

3.1 Erzeugen einer neuen Visualisierungsdatei

Klicken Sie mit einem Rechtsklick auf *Application - Objekt hinzufügen - Visualisierung*, siehe Abbildung 12 a). Geben Sie der Visualisierung in dem aufgehenden Fenster einen beliebigen Namen und klicken Sie auf *Hinzufügen* siehe Abbildung 12 b). Danach erscheint die neue Visualisierungsdatei in dem Gerätebaum, siehe Abbildung 12 c). Um mehrere Visualisierungsdateien zu erzeugen kann dieser Vorgang beliebig oft wiederholt werden. Jede einzelne Visualisierungsdatei stellte eine Seite dar auf der eine Visualisierung programmiert werden kann. Verschiedene Visualisierungsdateien lassen sich miteinander verknüpfen wodurch sich verschiedene Ansichten ergeben.

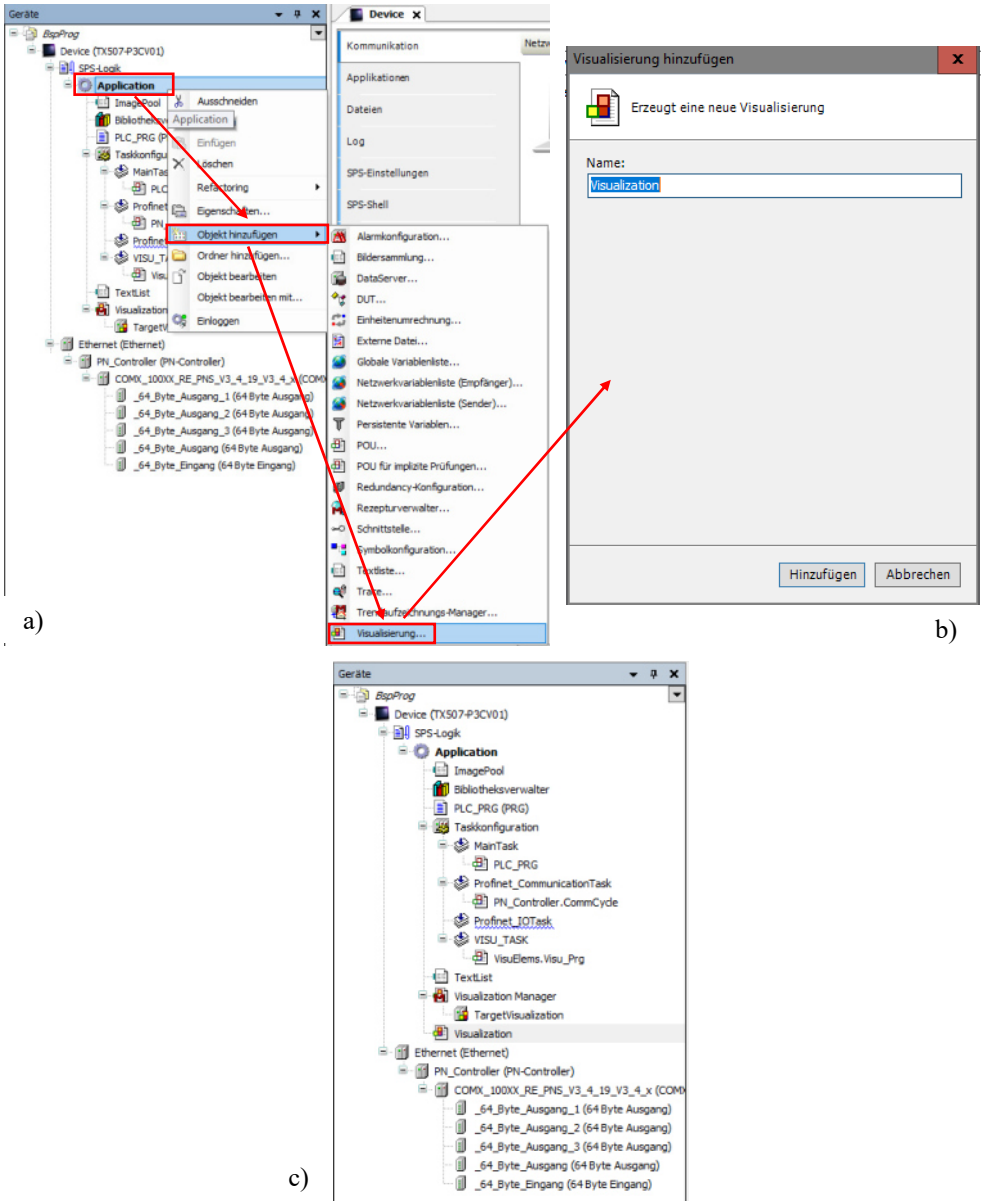


Abbildung 12: Erstellung einer neuen Visualisierungsdatei

3.2 Aktualisierungsrate und Format anpassen

Eine Anpassung der Aktualisierungsrate ist notwendig um zu gewährleisten, dass die Visualisierungen mit dem HMI abgestimmt sind. Andernfalls kann es zu ungewollten Verzögerungen beispielsweise beim Auslösen von Tastern oder Schaltern kommen. Dazu die *Target Visualization* im Gerätebaum per Doppelklick öffnen und die *Aktualisierungsrate* wie in Abbildung 13 z.B. auf 50 ms einstellen. Das Format stellt den Platz dar, welcher für eine Visualisierung zur Verfügung steht. Dieser wird unter *Skalierungsoptionen* angepasst. Dabei ist die Breite von 800 Pixel und eine Höhe von 480 Pixel zu beachten.

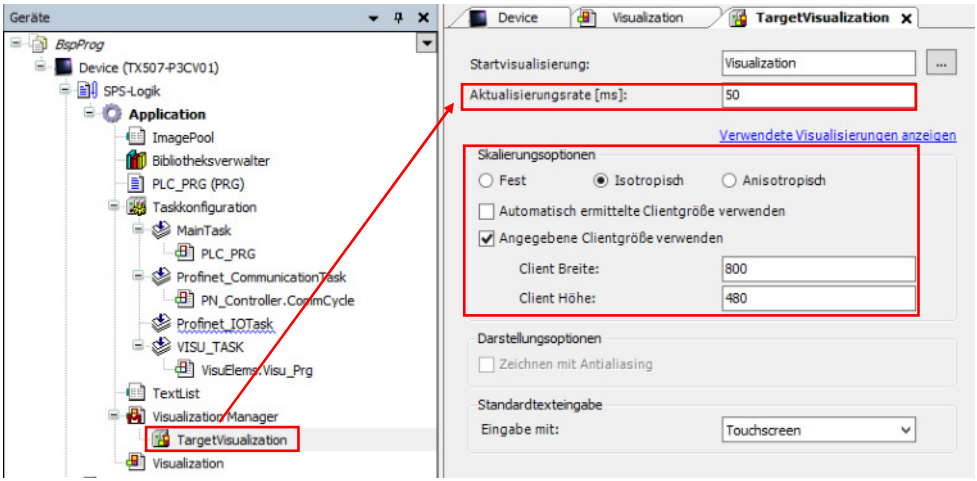


Abbildung 13: Aktualisierungsrate und Skalierungsoptionen

3.3 Buszykluszeit anpassen

Da das MVisio HMI in Kombination mit der ZX20 als Master betrieben wird, ist dieses für die Buszykluszeit zuständig. Die Buszykluszeit stellt ein Intervall dar, in welchem das HMI die Daten die von der ZX20 zur Verfügung gestellt werden, abfragt bzw. Daten an die ZX20 sendet. Um die Buszykluszeit anzupassen einen Doppelklick im Gerätebaum auf *COMX* ausführen, siehe Abbildung 14. Unter dem Reiter *Allgemein* kann nun unter dem Stichpunkt *Kommunikation - Send Clock* die Buszykluszeit angepasst werden:

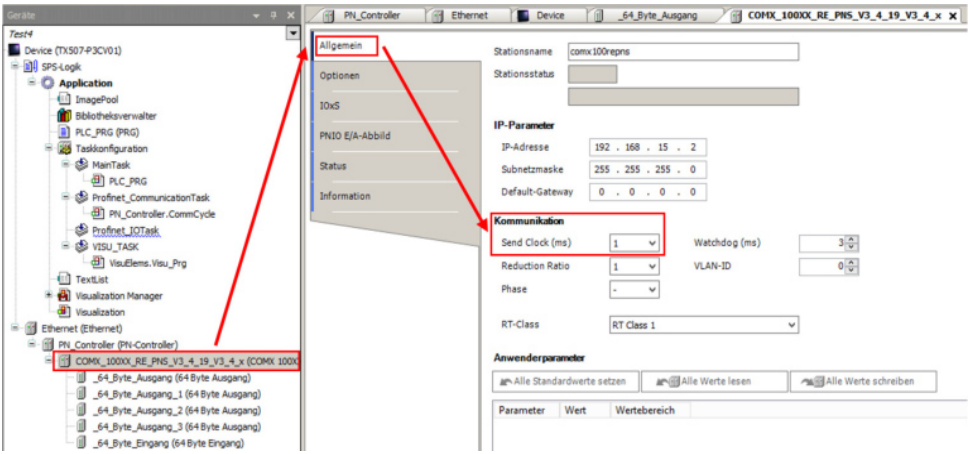


Abbildung 14: Anpassung der Buszykluszeit

3.4 Visualisierungselemente einfügen

Codesys stellt einige Standardelemente, welche zur Programmierung der Visualisierung genutzt werden können, zur Verfügungen. Dazu gehören beispielweise Taster, Schalter oder Lampen. Um diese zu verwenden die Visualisierungsdatei im Gerätebaum auswählen und per Doppelklick öffnen. Rechts erscheint ein *Werkzeuge - Feld* aus welchem die zu Verfügung stehenden Visualisierungselemente per Drag & Drop Prinzip eingefügt werden können, vergleiche Abbildung 15.

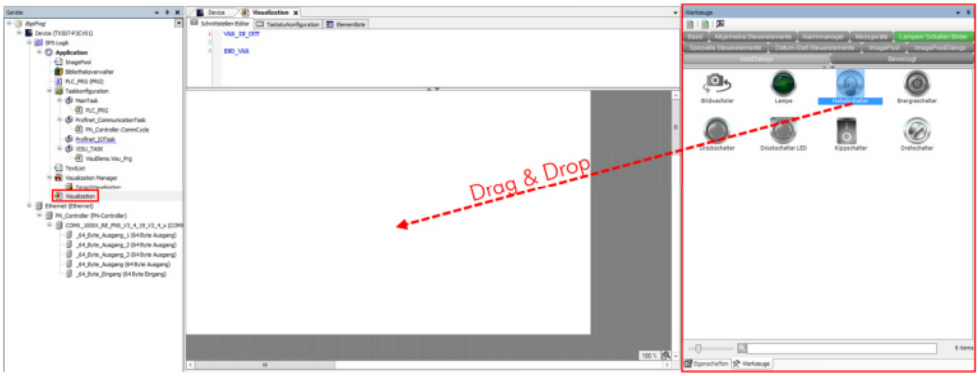


Abbildung 15: Visualisierungselemente einfügen per Drag & Drop-Prinzip

3.5 Visualisierungselemente konfigurieren

Nach dem Einfügen eines Visualisierungselementes öffnet sich das Konfigurationsmenü automatisch siehe Abbildung 16. Andernfalls können Sie dieses durch Klicken auf das jeweilige Elemente öffnen. Die Konfigurationsmöglichkeiten sind sehr umfangreich, sodass hier lediglich ein paar ausgewählte Einstellungen dargestellt werden. Für weitere Informationen lesen Sie bitte in der Hilfe von Codesys nach (https://help.codesys.com/). Das Konfigurationsmenü sieht nicht für alle Visualisierungselemente gleich aus. Dies ist abhängig von den Eigenschaften des jeweiligen Elementes. Um den vollen Umfang an Konfigurationsmöglichkeiten zur Verfügung zu haben stellen sie sicher, dass ein Haken bei *Experte* gesetzt ist. Die am häufigsten verwendeten Konfigurationselemente werden im Folgenden kurz erläutert und sind teilweise auch in Abbildung 16 dargestellt.

- *Position*:
  Über der weißen Visualisierungsfläche ist ein Koordinatensystem gelegt, welches seinen Ursprung in der oberen linken Ecke hat. Die x-Achse zeigt waagerecht nach links und die y-Achse senkrecht nach unten.
  - X, Y: Positionsangaben für die Visualisierungselemente.
  - Breite, Höhe: Größe des Visualisierungselemente.
- *Variable:*
  Hier kann das Visualisierungselement mit einer Programmvariable verknüpft werden.
- *Zustandsvariablen*:
  - Unsichtbarkeit: Mit diesem Tool lassen sich Visualisierungselemente bei Verknüpfung mit einer Variable unsichtbar machen.
  - Eingaben deaktivieren: Hiermit bleiben Visualisierungselemente sichtbar reagieren jedoch nicht auf eine Eingabe.
- *Textvariablen - Textvariable:*
  Hier können Variablen in Text (in eine sichtbare Zahl) umgewandelt werden um sich diese anzeigen zu lassen.
- *Eingabekonfiguration - Tasten - Variable*:
  Hier kann das Visualisierungselemente mit einer Programmiervariable verknüpft werden

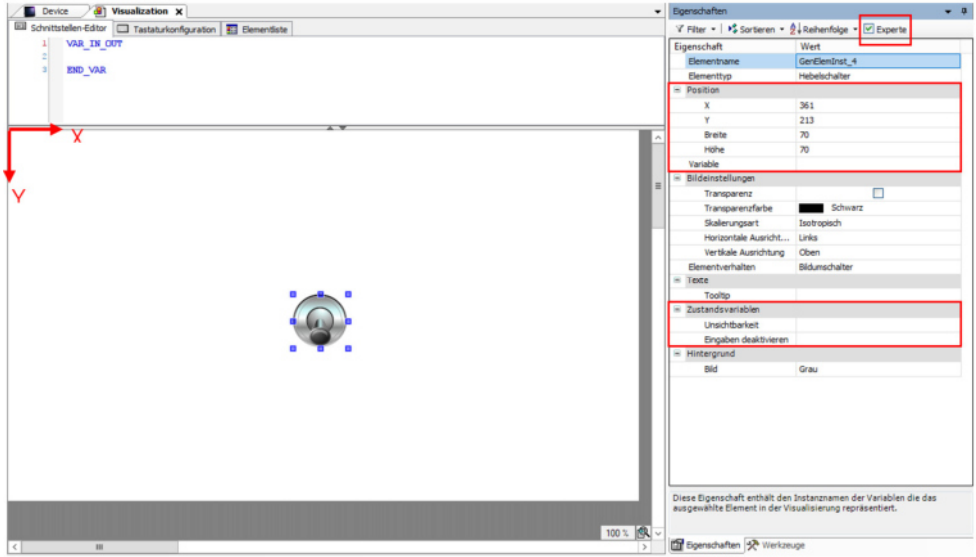


Abbildung 16: Konfiguration des Visualisierungselements Hebelschalter

3.6 Deklaration von Variablen in Codesys

In Codesys werden Variablen in Abhängigkeit des Anwendungsbereiches deklariert. Dabei ist es wichtig zu berücksichtigen, dass das MVisio HMI sowohl als Master als auch als Slave arbeiten kann. Im Masterbetrieb können auf dem MVisio HMI Programme geschrieben werden, wobei die ZX20 SPS dabei als Slave betrieben wird. Zunächst soll kurz auf eine Auswahl an Variablen und deren Wirkungsbereich eingegangen werden:

Typ 1: SPS Variablen in einem Codesys Programm:

Diese Variablen werden in einem Programm in Codesys (siehe Gerätebaum: PLC_PRG) und damit auf dem HMI deklariert. Der Wirkungsbereich der Variablen beschränkt sich hierbei auf das Codesysprogramm und auf die Visualisierungen. Diese Variablen können nicht für die Kommunikation mit der ZX20 SPS verwendet werden, siehe Abbildung 17.

Typ 2: Variablen zur Kommunikation mit der ZX20 SPS:

Diese Variablen werden an den Ein- und Ausgängen der Profinet Schnittelle des HMI deklariert. Sie dienen zur Kommunikation mit der ZX20 SPS. Die Variablen können in Visualisierungen genutzt werden. Eine Nutzung dieser Variablen in einem Codesysprogramm ist jedoch nicht möglich, siehe Abbildung 17.

Typ 3: Globale Variablen

Eine direkte Nutzung der Ein- und Ausgangsvariablen in einem Codesysprogramm ist nicht möglich. Hierfür müssen die Variablen über welche die Kommunikation stattfindet, erneut deklariert werden. Dies geschieht im Gerätebaum unter dem Reiter GLV. Hier werden die Variablen aus den Ein/Ausgängen erneut eingetragen um dies dann in einem Codesysprogramm nutzen zu können, siehe Abbildung 17.

In der folgenden Abbildung 17 sind die einzelnen Variablentypen und deren Nutzungsbereich nochmals graphisch veranschaulicht.

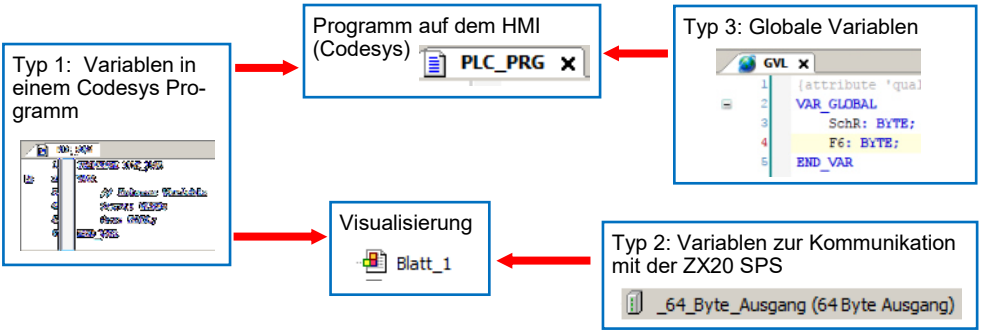


Abbildung 17: Variablen und deren Nutzungsbereich

3.6.1 Anlegen von Variablen in einem Codesys Programm (Typ 1)

Im Gerätebaum einen Doppelklick auf PLC_PRG (PRG) ausführen. Dann im oberen Feld die gewünschten Variablen deklarieren siehe Abbildung 18. Hier wurden die Variablen Start und Box deklariert welche im der Trace-Visualisierung (siehe Absatz 4.6) genutzt werden.

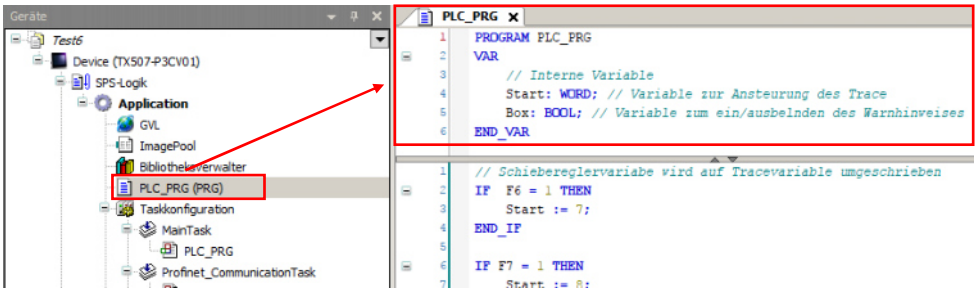


Abbildung 18: Anlegen von Variablen des Type 1

3.6.2 Anlegen von Variablen zur Kommunikation mit der ZX20 SPS (Typ 2)

Sowohl eingehende als auch ausgehende Variablen zur Kommunikation mit der ZX20 SPS müssen in Codesys deklariert werden. Eine Variable, die in einem Ausgang deklariert ist, wird von dem HMI an die SPS geschickt. Bei einem Eingang ist dies genau umgekehrt. Das Anlegen einer neuen Variable zur Kommunikation mit der ZX20 SPS erfolgt durch einen Doppelklick auf den gewünschten Ein- oder Ausgang im Gerätebaum (in diesem Fall Ausgang 1), siehe Abbildung 19. Dann den Reiter PNIO Module E/A-Abbild anklicken und unter Variablen das kleine + auswählen. Nun werden alle Variablen angezeigt die übergeben oder empfangen werden. In diesem Fall sind dies die Variablen EinAus, Up, CheckR, CheckT und SchR, welche an die SPS übergeben werden.

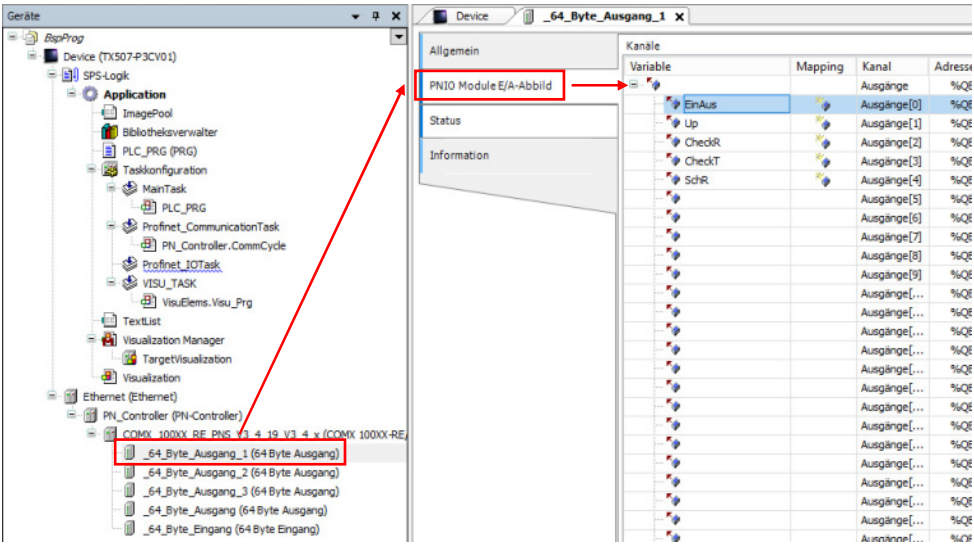


Abbildung 19: Auswahl der Variablen des Types 2

Um eine neue Variable anzulegen einen Doppelklick unter der Spalte Variable ausführen und der Variablen einen beliebigen Namen zuweisen (hier: Counter), siehe Abbildung 20 Dann in der Spalte Mapping klicken und =Neue Variable oder =Auf bestehende Variable mappen konfigurieren.

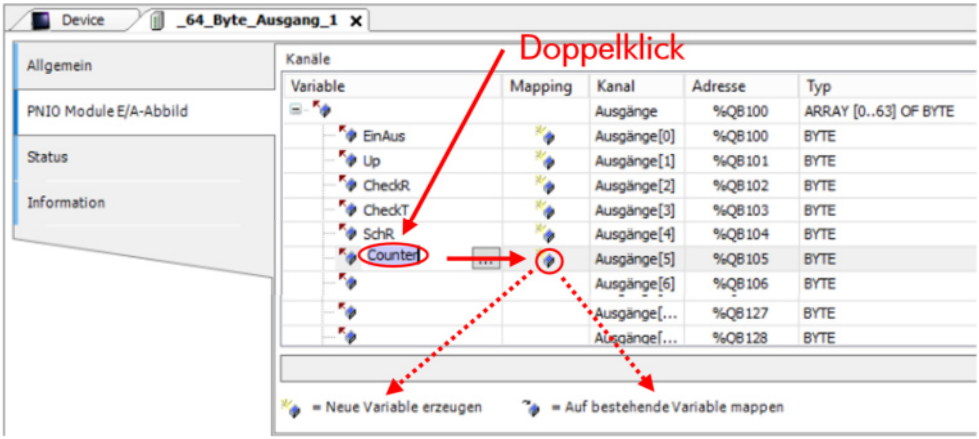


Abbildung 20: Deklaration der Variablen „Counter“ vom Type 2

Für das Programmierbeispiel in Absatz 5 sind z.B. folgende Variablen zu deklarieren:

Ausgang: EinAus, Up CheckR, CheckT und SchR, siehe Abbildung 21.

Eingang: Ein, Aus, F6, F7, F8, F9, und Count, siehe Abbildung 22 c).

Wichtig: Die Reihenfolge der Variablen ist zu beachten! Diese muss gleich der Reihenfolge der Variablen in dem Programm der SPS sein! Namensgleichheit ist hingegen nicht notwendig (siehe auch Absatz 5.2.1)!

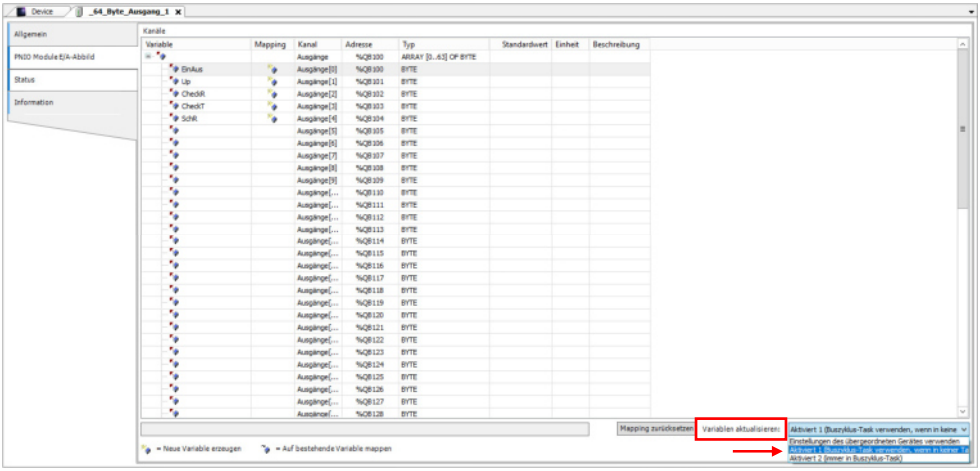


Abbildung 21: Variablen Aktualisierung

Damit die Variablen laufend aktualisiert werden bei *Variablen aktualisieren – Aktivieren 1 setzen (für Ein- und Ausgänge),* siehe Abbildung 21.In Codesys lassen sich Variablen des Typs 2 nur mit dem Datentyp Byte erzeugen. D.h. das ausschließlich über diesen Datentyp mit der SPS kommuniziert werden kann. Dies gilt für beide Kommunikationsrichtungen (HMI ◄►► SPS). Mit dem Datentyp Byte lassen sich beispielsweise die Zahlen von 0 bis 255 darstellen.

3.6.3. Anlegen einer Globalen Variablen (Typ 3)

Um eine Globale Variable zu deklarieren, auf *Bearbeiten - Variable deklarieren* klicken, siehe Abbildung 22 a). In dem aufgehenden Fenster unter *Gültigkeitsbereich - VAR_GLOBAL* und unter *Objekt - GVL [Application]* einstellen, siehe Abbildung 22 b). Unter Name den Variablennamen des gewünschten Variablentypes 2 eintragen und bei *Datentyp - Byte* einstellen, siehe Abbildung 22 c). Um die Variablen des Type 2 auf den Type 3 umzuschreiben muss Namensgleichheit gegeben sein. Mit diesem Verfahren sind folgende Variablen des Datentyps BYTE zu deklarieren: *SchR, F6, F7, F8, F9, CheckR, CheckT* und *Ein.* Dieser werden nach der Initialisierung in der GVL-Datei angezeigt, siehe Abbildung 23.

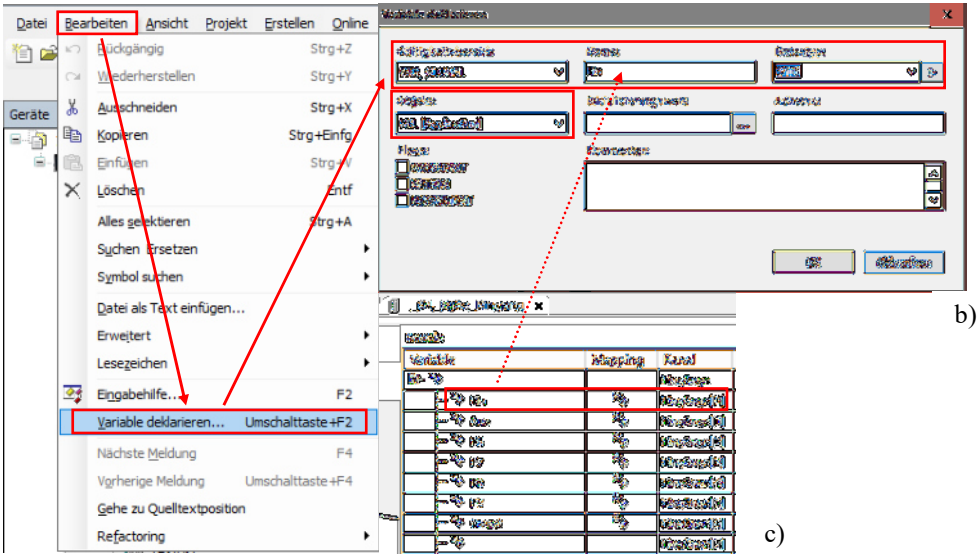


Abbildung 22: Deklaration einer Variable des Types 3

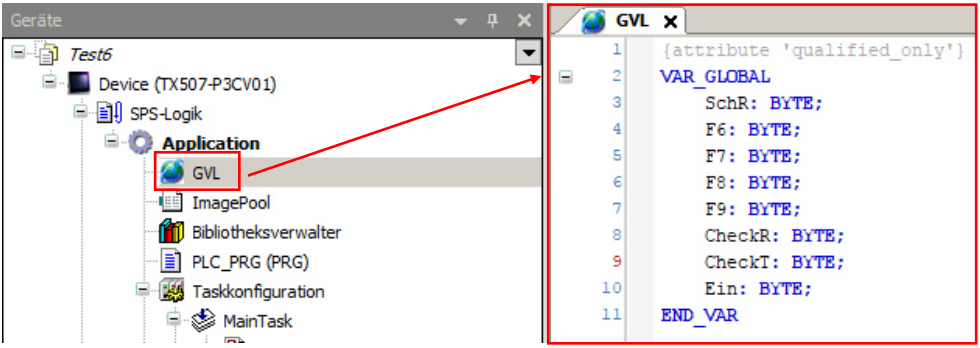


Abbildung 23: Deklarierung der Variablen des Types 3

3.7 Verknüpfen einer Variablen mit einem Visualisierungselement

Um beispielsweise einen Schalter mit einer zuvor angelegten Variablen des Typs 1 oder 2 zu versehen, muss zunächst das Konfigurationsmenü des Schalters geöffnet werden. Dann einen Doppelklick auf das Feld neben der *Variable* ausführen und dann auf *(…) klicken, siehe* Abbildung 24. Im Folgenden öffnet sich ein Fenster wo alle zur Verfügung stehenden Variablen zur Verknüpfung mit dem Schalter ausgewählt werden können. Unter dem Reiter *IoConfig_Globals_Mapping* lassen sich die zuvor angelegten Variablen des Typs 2 auswählen, siehe Abbildung 24. Unter *Application - PLC_PRG* lassen sich die Variablen des Typs 1 einbinden. Beim Einschalten des Schalters wird die ihm zugewiesene Variable von Low auf High gesetzt. Beim Ausschalten umgekehrt.

Des Weiteren ist es wichtig zu verstehen, dass nicht jedes Visualisierungselement den Datentype Byte der Variableninitialisierung akzeptiert. Ein Schalter beispielsweise mit einer Variablen des Types Byte zu verknüpfen ist wenig sinnvoll, da ein Schalter nur zwei Zustände kennt (TRUE oder FALSE bzw. 1 oder 0 bzw. High oder Low). Ein Schalter kann eine ihm zugewiesene Zahl z.B. 42 nicht umsetzen. Um einen Schalter mit einer Variablen des Typs Byte zu verknüpfen wird nur das erste Bit der Byte Variable betrachtet. Dies wird erreicht, in dem an den Namen der Variable ein „.0“ bei der Verknüpfung mit dem Visualisierungselement angehängt wird. Im Gegensatz dazu steht beispielsweise das Visualisierungselement Textfeld, welches Zahlen darstellen kann. Dies kann mit einer Variablen des Datentyps Byte verknüpft werden.

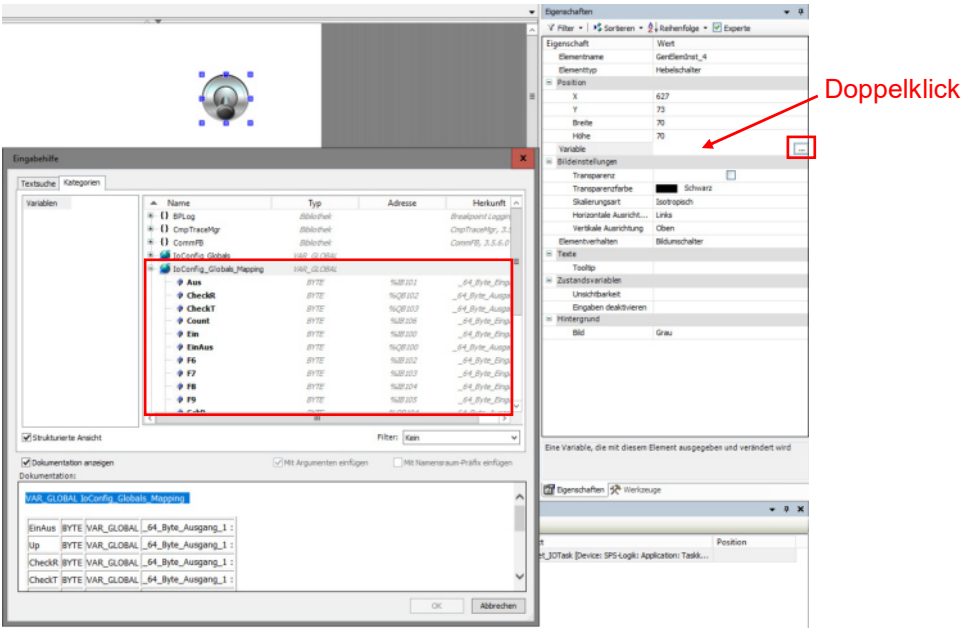


Abbildung 24: Verknüpfung eines Schalters mit einer Variablen

3.8 Starten und Beenden einer Visualisierung

Um die Visualisierung zu starten zunächst auf übersetzen klicken. Sollten keine Fehler angegeben werden dann auf *Einloggen - Start* klicken, siehe Abbildung 25. Nun sollte die erstellte Visualisierung auf dem HMI angezeigt werden. Zum Beenden der Visualisierung auf *Stopp - Ausloggen* klicken. Die Visualisierung kann auch getestet werden, ohne dass das HMI mit dem PC verbunden ist. Dafür unter Online die Schaltfläche Simulation auswählen. Danach die Simulation wie oben beschrieben starten. Im Simulationsmodus ist zu beachten, dass keine Variablen des Types 2 abgefragt werden. Somit kann die uneingeschränkte Funktion der Visualisierung nicht garantiert werden.

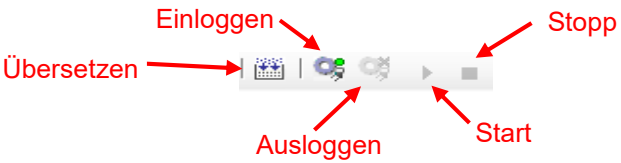
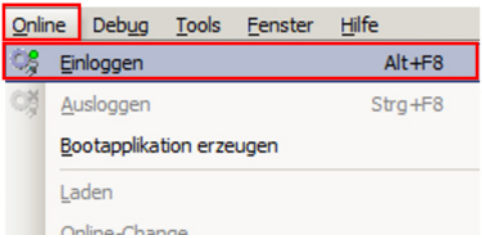
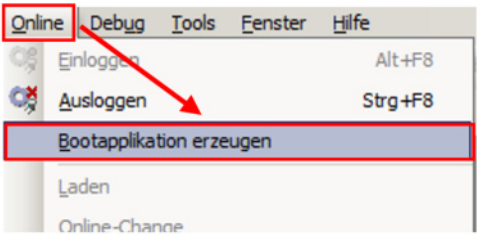


Abbildung 25: Symbolleiste zum Starten/Stoppen der Visualisierung

3.9 Eine Bootapplikation erzeugen

Das Erzeugen einer Bootapplikation ist notwendig wenn das HMI losgelöst vom PC seinen Betrieb aufnehmen soll. Dazu zunächst auf *Online - Einloggen* klicken, siehe Abbildung 26 a). Nach dem Download auf *Online - Bootapplikation erzeugen* klicken, siehe Abbildung 26 b). Nun wird ein Bootapplikation erzeugt und auf dem HMI gespeichert. Um die Bootapplikation zu testen Codesys schließen und das HMI neu starten. Dieses zeigt nach dem Neustart die zuvor erstellte Visualisierung an.

a) 



b)

Abbildung 26: Erzeugen einer Bootabpplikation

3.10 Simulation

Um eine Simulation des Codesysprogramms durch zu führen auf *Online - Simulation* klicken. Hierbei ist zu beachten, dass ausschließlich Simulationen für Programme durchgeführt werden können, welche auf dem HMI und somit in Codesys programmiert werden. Eine Simulation von Variablen die extern dem HMI zugesendet werden ist nicht möglich. Um das Simulationstool zu deaktivieren auf *Online - Simulation* klicken.

4. Beispielprogramm: MVisio HMI mit Codesys

Das Beispielprogramm dient zunächst dazu die in Kapitel 2 und 3 dargestellte Theorie zu veranschaulichen. Es soll die bisher erläuterten Praktiken aufgreifen und einen Gegensatz zu dem bisherigen Theorieteil darstellen. Dabei werden nicht ausschließlich die Visualisierungsmöglichkeiten des MVisio HMI dargestellt. Ein wichtiger Punkt stellt die Visualisierung der Kommunikation zwischen der SPS und dem HMI dar.

4.1 Zieldefinition

Das Beispielprogramm in Codesys soll in Zusammenarbeit mit dem Strukturierten Textprogramm in Absatz 5, welches auf ZX20 SPS programmiert wird, die Kommunikation zwischen HMI und SPS verdeutlichen. Dazu soll der Benutzer über das HMI die SPS Ein- und Ausschalten können, siehe Abbildung 28. Des Weiteren soll ein Ausgang der SPS

über einen Taster oder Schieberegler ausgewählt werden können, siehe Abbildung 31. Der ausgewählte Ausgang wird auf dem HMI als Graphik und auf der SPS über eine LED angezeigt. Zuletzt gibt es auf dem HMI eine Graphen welcher den Verlauf der ausgewählten Ausgänge in den letzten 10 Sekunden zeigt, siehe Abbildung 33

4.2 Projektinitialisierung

Legen Sie ein neues Projekt an, siehe Abschnitt 11.3. Erstellen Sie drei Visualisierungsdateien (hier mit den Namen *Blatt_1, Blatt_2, Blatt_3*) wie in Abschnitt 12.1 erläutert. Der Gerätebaum sollte danach wie in Abbildung 27 aussehen.

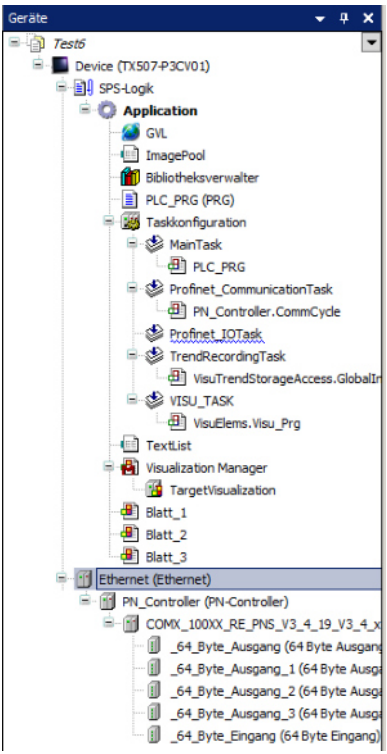


Abbildung 27: Gerätebaum bei Projektstart

4.3 Variablendeklaration

Zunächst werden alle benötigten Variablen deklariert. Dabei ist auf die verschiedenen

Variablen Typen (siehe Absatz 3.6) zu achten.

Type 1 SPS Variablen im Codesys Programm:

Wie in Absatz 3.6.1 erläutert und in Abbildung 18 dargestellt werden in der *PLC_PRG* Umgebung die Variablen *Start (WORD)* und *Box* (*BOOL)* initialisiert.

Type 2 Variablen zur Kommunikation mit der ZX20 SPS:

Wie in Absatz 3.6.2 erläutert werden die Variablen an dem jeweiligen Ein bzw. Ausgang deklariert. Die Eingangsvariablen lauten: Ein, Aus, F6, F7, F8, F9, und Count. Die Variablen am Ausgang lauten: *EinAus, Up CheckR, CheckT* und *SchR*. Bitte dabei die Reihenfolge der Variablen beachten.

Type 3 Globale Variablen:

Wie in Absatz 3.6.3 erläutert und in Abbildung 23 dargestellt werden in der *GLV* Umgebung die Variablen *SchR (BYTE), F6 (BYTE), F7 (BYTE), F8 (BYTE), F9 (BYTE), CheckR (BYTE), CheckT (BYTE)* und *Ein (BYTE)* deklariert.

4.4 Visualisierung 1

Das erste Visualisierungsblatt soll dem Nutzer ermöglichen die SPS Ein bzw. Aus zuschalten. Dabei soll sichergestellt werden, dass die Kommunikation zwischen der ZX20 SPS und dem MVisio HMI einwandfrei funktioniert.

4.4.1 Oberfläche

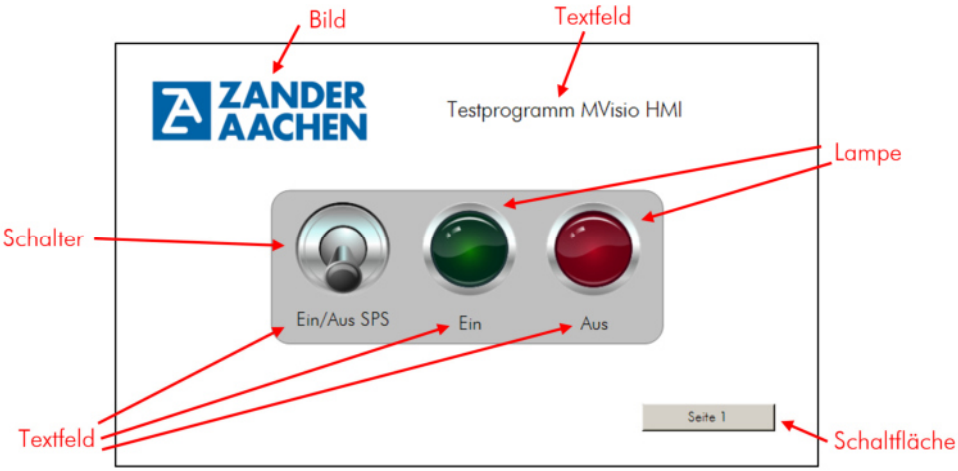


Abbildung 28: Oberfläche 1

Für die Oberfläche der Visualisierung werden ein Schalter, zwei Lampen, eine Schaltfläche, zwei Rechtecke, ein Bild und 4 Textfelder verwendet, siehe Abbildung 28. Im Konfigurationsmenü (siehe Abbildung 16) lassen sich unter Position die Visualisierungselemente genau ausrichten. Des Weitern kann es hilfreich sein die Visualisierungselemente zu gruppieren (siehe *Visualisierung - Gruppieren*). Um ein Bild einzufügen zunächst im Gerätebaum auf *ImagePool* klicken, siehe Abbildung 29 a). Dann einen Doppelklick auf das Feld unter *Dateiname* ausführen. Die daraufhin erscheinenden drei Punkte auswählen und in dem aufgehenden Fenster unter *Bilddatei* den Dateipfad eines Bildes angeben, siehe Abbildung 29 b). Danach auf *OK* klicken. Im *Werkzeuge*-Feld *Bild* per Drag & Drop in die Visualisierung einfügen und das gewünschte Bild auswählen, Abbildung 29 c).

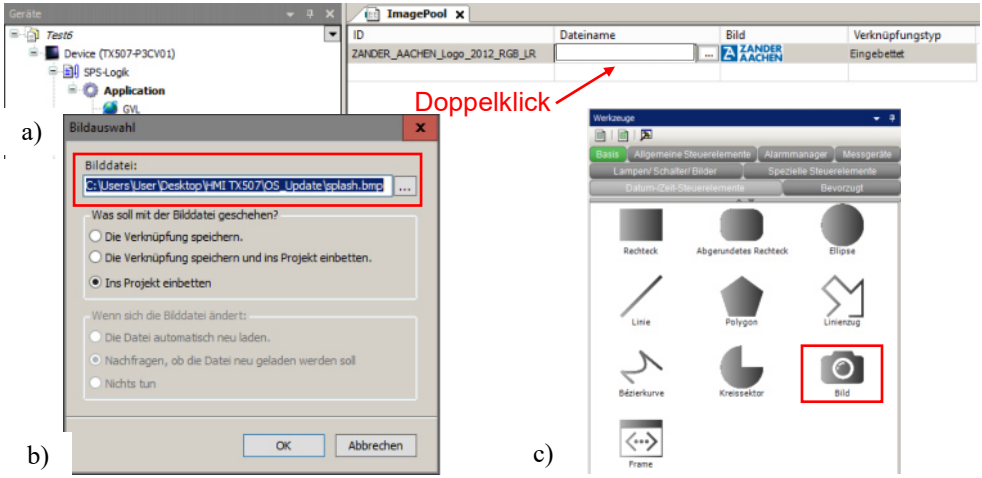


Abbildung 29: Einfügen einer Bilddatei

4.4.2 Variableneinbindung

Schalter und Lampen:

Der Schalter wurde mit der Ausgansvariable „EinAus.0“ deklariert. Die Lampen jeweils mit den passenden Eingangsvariablen „Ein.0“ und „Aus.0“. Bei beiden Visualisierungselemente ist es wichtig, dass nur die erste Bit-Stelle betrachtet wird (deshalb „.0“). Eine Zahl kann durch ein Boolsches Visualisierungselement nicht ausgedrückt werden, siehe Abschnitt 12.7.

Schaltfläche:

Die Schaltfläche dient dazu zwischen den beiden Visualisierungen *Blatt 1 und Blatt 2* zu wechseln. Dazu wird zunächst die Schaltfläche ausgewählt und das Konfigurationsmenü

geöffnet, siehe Abbildung 30. Dann unter *Eingabekonfiguration - OnMouseDown - Konfgurieren* anklicken, siehe Abbildung 30 a). Nun öffnet sich das Eingabekonfiguration - Fenster. In der linken Spalte Visualisierungswechsel auswählen und auf den Pfeil nach rechts *(>)* klicken, siehe Abbildung 30 b). In der mittleren Spalte Visualisierungswechsel auswählen und unter Zuweisen - *(…)* - Blatt_2 auswählen, siehe Abbildung 30 c). Damit sich die Schaltfläche in beiden Visualisierungen (Blatt 1 und Blatt 2) an der gleichen Stelle befindet empfiehlt es sich eine Positionierung mit den X, Y Koordinaten vorzunehmen.

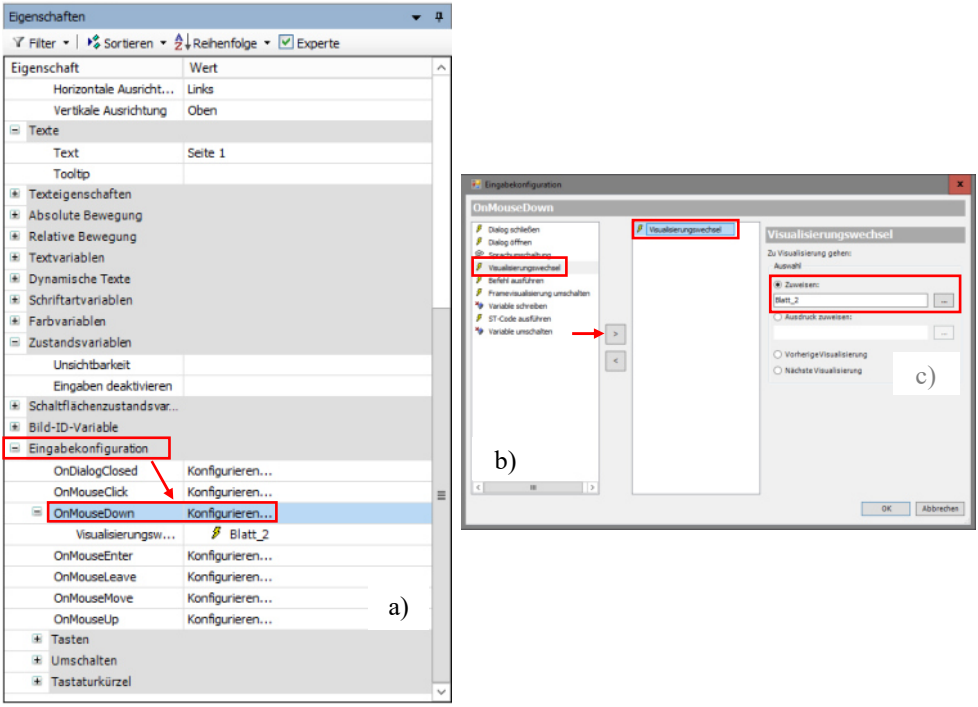


Abbildung 30: Konfiguration eines Visualisierungswechsels

4.5 Visualisierung 2

Die zweite Visualisierungsoberfläche ermöglicht dem Nutzer einen Ausgang der SPS auszuwählen. Dies kann über einen Schieberegler oder einen Taster geschehen. Gleichzeitig wird der Zustand der Ausgänge der SPS auf dem HMI nochmals angezeigt.

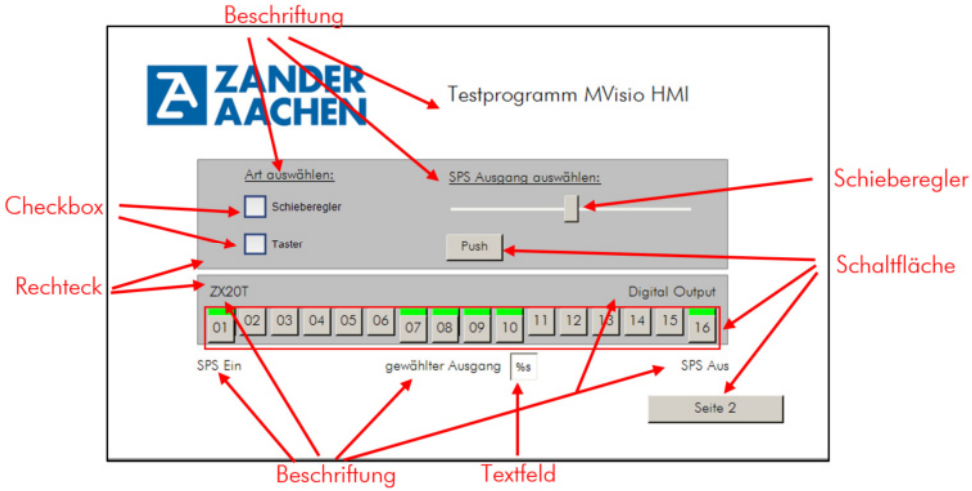


Abbildung 31: Oberfläche 2

4.5.1 Oberfläche

Hierzu wurden acht Beschriftungen, ein Textfeld, zwei Checkboxen, ein Schieberegler, 24 Schaltflächen und zwei Rechtecke verwendet, siehe Abbildung 31. Im Konfigurationsmenü (siehe Abbildung 16) lassen sich unter Position die Visualisierungselemente genau ausrichten. Des Weitern kann es hilfreich sein die Visualisierungselemente zu gruppieren (siehe *Visualisierung – Gruppieren*).

4.5.2 Variableneinbindung

Checkbox:

Die Checkboxen sind mit zwei verschieden Variablen gekoppelt. Unter Position - Variable ist je nach Box „CheckT.0“ oder „CheckR.0“ zu finden. In dieser Variable wird das gesetzte / nicht gesetzte Häkchen gespeichert. Da dies eine boolsche Eigenschaft ist wird an die Variable „.0“ angefügt, siehe auch Absatz 3.7. Unter Zustandsvariablen - Eingaben deaktivieren ist die Variable „Aus.0“ vermerkt. Dies bewirkt, dass falls die SPS ausgeschaltet ist (Aus.0 = 1) nicht die Möglichkeit besteht ein Häkchen zu setzen.

Schieberegler:

Der Schieberegler ist unter Position – Variable mit der Variablen „SchR“ gekoppelt. Da der Schieberegler die Werte von 0 bis 3 annehmen kann (siehe Skala - Skalenanfang bzw. Skalenende) wird hier auf den Zusatz „.0“ verzichtet, siehe auch Absatz 3.7. Der Schieberegler stellt kein boolsches Visualisierungselement dar, sondern kann ein beliebige

Anzahl von Zuständen annehmen. Unter Zustandsvariablen - Eingaben deaktivieren ist die Variable „Aus.0“ vermerkt. Dies bewirkt, dass falls die SPS ausgeschaltet ist (Aus.0 = 1) nicht die Möglichkeit besteht den Schieberegler zu betätigen.

Schalfläche Push:

Die Schaltfläche Push ist unter Eingabekonfiguration - Tasten - Variable mit der Variable „Up.0“ verknüpft. Diese gibt dem SPS ZX20-Programm einen Zählimpuls. Dieser Zählimpuls wird im dem ZX20 Programm so umgesetzt das immer der nächst höhere Ausgang ausgewählt wird. Beim Erreichen des höchsten Zählstandes beginnt das Programm von vorne. Unter Zustandsvariablen - Eingaben deaktivieren ist die Variable „Aus.0“ vermerkt. Dies bewirkt, dass falls die SPS ausgeschaltet ist (Aus.0 = 1) nicht die Möglichkeit besteht den Taster zu betätigen.

Schalflächen SPS Ausgänge:

Die Schalflächen 02 bis 06 und 11 bis 15 dienen lediglich der Visualisierung und sind daher mit keiner Variablen verknüpft. Unter Zustandsvariablen - Unsichtbarkeit ist die Schaltfläche 01 mit der Variable „Ein.0“ verknüpft. Die Schaltfläche wird somit unsichtbar bei „Ein.0 = 1“. Das hat die Folge, dass die exakt darunter liegende Schalfläche 01 (grün), welche ohne eine Variable versehen ist, in Erscheinung tritt. Beim Ausführen der Visualisierung hat dies den Effekt, dass die Schaltfläche 01 scheinbar von grau auf grün wechselt. Nach dem gleichen Verfahren wird bei den Schaltflächen 07, 08, 09, 10 und 16 vorgegangen. Dabei werden Variablen „F6“ bis „F9“ mit den Schaltflächen 07 bis 10 verknüpft. Die Schalfläche 16 wird mit der Variable „Aus.0“ verknüpft.

Textfeld:

Das Textfeld ist unter der Textvariablen - Textvariable mit der Variablen „Count“ verknüpft. Da das Textfeld verschiedene Werte anzeigen soll wird hier auf den Zusatz „.0“ verzichtet, siehe auch Absatz 3.7. Das Textfeld soll nun den Wert der Variable „Count“ anzeigen. Dazu wird unter Texte - Text %s (String) eingetragen, siehe Abbildung 32. Dies hat zur Folge, dass die Variable „Count“ in das Textfeld als String hineingeschrieben wird. Während der Programmierung wird auf dem HMI weiter %s angezeigt. Wird jedoch die Simulation gestartet (siehe Absatz 3.10) erscheint der Wert der Variablen „Count“ auf dem HMI.

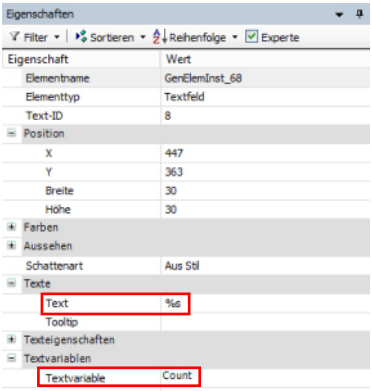


Abbildung 32: Konfiguration des Textfelds

4.6 Visualisierung 3

Bei der dritten Visualisierung steht die Echtzeitapplikation *Trace* im Vordergrund. Dabei wird der gewählte Ausgang, der Zeit in einem Graphen gegenübergestellt. Die Zeitachse visualisiert immer die letzten 10 Sekunden und wird laufend aktualisiert. Um ein überschneiden der Eingabe von Taster und Schiebe-Regler zu vermeiden erscheint beim gleichzeitigen auswählen ein Warnhinweis.

4.6.1 Oberfläche

Bei der Oberfläche wird lediglich auf die neuen Visualisierungselemente eingegangen. Hierzu gehört der Elementtyp Trace. Mit diesem lässt sich ein Graph anzeigen welcher eine zeitliche Abhängigkeit aufweist. Des Weiteren wurde ein Warnhinweis mit einem Textfeld eingefügt, siehe Abbildung 33.

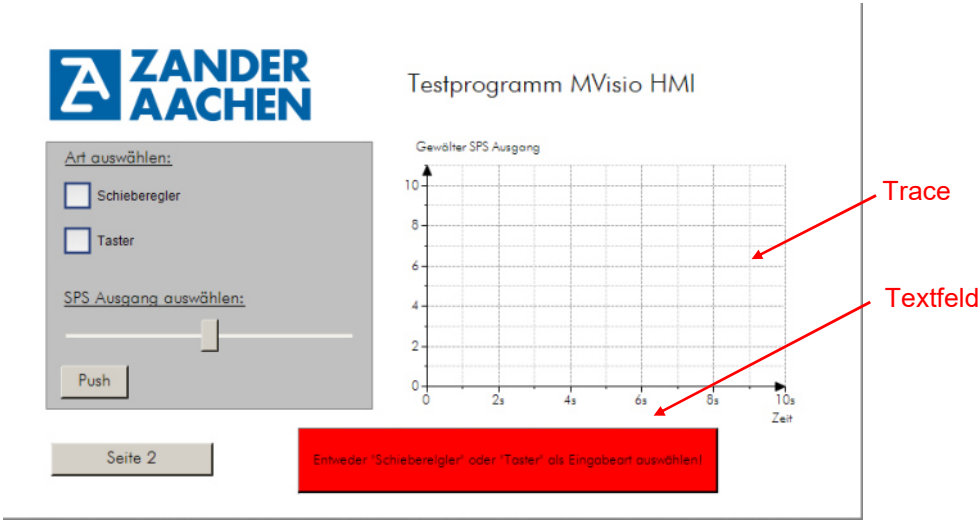


Abbildung 33: Oberfläche 3

4.6.2 Codesysprogrammierung

Auf dem MVisio HMI kann ähnlich wie auf einer SPS mit strukturiertem Text programmiert werden. Das Schreiben eines Programms in Codesys ist der Tatsache geschuldet, dass sich die Variablen zur Kommunikation zwischen HMI und SPS (Type 2) nicht direkt in die Tracevisualisierung einbinden lassen, siehe auch Absatz 3.6 und Abbildung 17. Somit werden die benötigten Variablen des Type 2 zunächst in den Globalen Variablen Type 3 umgeschrieben (siehe Abbildung 23) um diese dann in einem Codesysprogramm zu verwenden. Des Weiteren werden Variablen benötigt, welche im Codesysprogramm selber generiert werden, siehe auch Abschnitt 3.6.1. Zur Programmierung des HMI im Gerätebaum auf *PLC_PRG (PRG)* klicken, siehe Abbildung 34. Dies stellt die interne Programmierplattform in Codesys dar. Im oberen Feld werden die benötigten Variablen (Type 1) deklariert die für die interne Programmierung benötigt werden, siehe Abbildung 34 a). Im unteren Feld kann mit strukturierten Text programmiert werden, siehe Abbildung 34 b).

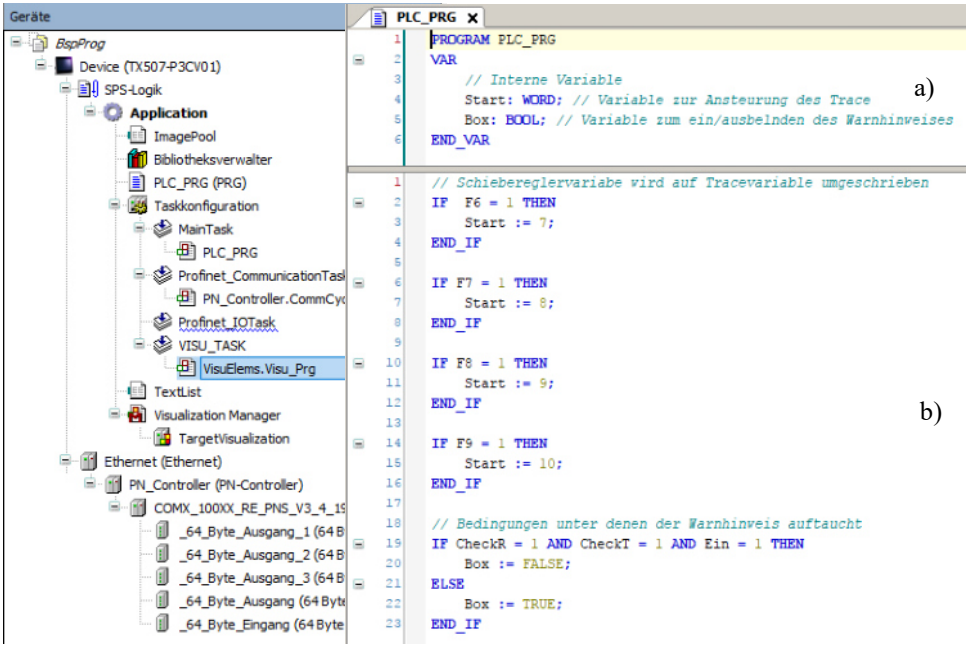


Abbildung 34: Programmierumgebung in Codesys mit Beispielprogrammcode

4.6.3 Variableneinbindung

Trace:

Eine Besonderheit stellt die X-Achse der Trace-Graphik dar. Diese zeigt die letzten 10 Sekunden an und wird laufend erneuert. Um die Tracevisualisierung zu konfigurieren einen Doppelklick im Konfigurationsmenü auf *Trace* ausüben, siehe Abbildung 35 a). Dann auf *Variable hinzufügen* klicken, siehe Abbildung 35 b). Unter *Variable* die gewünschte Variable, welche auf der Y-Achse aufgetragen werden soll auswählen, siehe Abbildung 35 c). Bei der Variablenauswahl ist zu beachten, dass sich nur Variablen des Typs 1 (siehe Absatz 3.6.1) auswählen lassen.

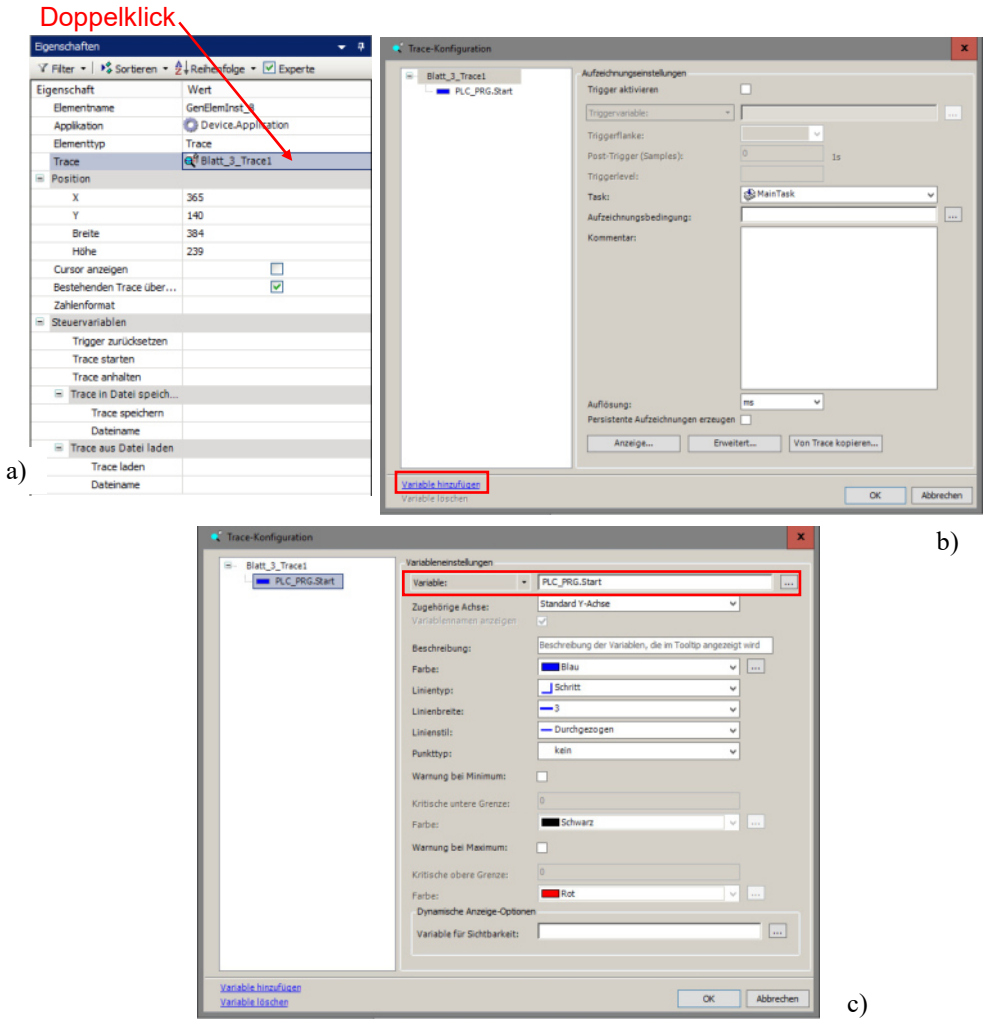


Abbildung 35: Konfiguration der Trace Visualisierung

Textfeld:

Der Warnhinweis soll nur angezeigt werden wenn die Checkbox Schieberegler und Taster gleichzeitig ausgewählt sind. Des Weiteren muss die SPS eingeschaltet sein. Diese Bedingungen werden im Codesysprogramm (Abbildung 34 ab Zeile 18) definiert und in der

Variablen Box hinterlegt. Für das Ein- und Ausblenden des Textfeldes wird im Konfigurationsmenü die Option Unsichtbarkeit genutzt.

5. Beispielprogramm: ZX20 SPS mit EX_PRESS 5

Die ausführlichen Erläuterungen in Absatz 4 betrafen bis jetzt das Beispielprogramm hinsichtlich der Codesys Programmierung für das HMI. Das eigentliche Hauptprogramm, welches die Eingaben des HMI verarbeitet und umsetzt, liegt jedoch auf der ZX20 SPS. Daher soll im Folgenden das Programm der ZX20 SPS genauer erläutert werden.

Wichtig: Die in diesem Absatz dargestellt Erklärungen setzten einen grundlegenden Wissenstand in der Programmierung der ZX20 mit Strukturiertem Text voraus. Eine ausführlichen Einstieg in die Programmierung der ZX20 finden den sie in dem dazu gehörigen Handbuch (https://www.zander-aachen.de/files/zander/betriebsanleitung/automation/steuerungen/benutzerhandbuch-ex_press-5.pdf).

5.1 Zieldefinition

Das Beispielprogramm in Codesys aus Absatz 4 soll in Zusammenarbeit mit dem Strukturierten Textprogramm auf der ZX20 die Kommunikation zwischen HMI und SPS verdeutlichen. Aufgabe des ZX20-Programms ist es dabei die ankommenden Variablen des HMI zu empfangen, zu verarbeiten und zuletzt Variablen an das HMI zurück zu senden. Das HMI nutzt dabei die von der ZX20 ausgewerteten Daten zur Visualisierung.

5.2 Variablen Initialisierung (Zeile 14 bis 45)

Zunächst werden alle benötigten Variablen in den Zeilen 14 - 46 deklariert. Dabei wird gesondert auf die Kommunikationsvariablen zwischen SPS und HMI eingegangen. Danach folgen die Variablen welche für die interne Programmverarbeitung notwendig sind.

5.2.1 Kommunikationsvariablen zwischen ZX20 SPS und dem MVisio HMI (Zeile 14 bis 30)

Zunächst soll auf die Kommunikation via PROFINET zwischen der ZX20 SPS und dem MVisio HMI eingegangen werden. In Codesys wurden an den Ein- und Ausgängen der PROFINET-Schnittstelle alle Variablen welche vom HMI auf die SPS und von der SPS auf das HMI geschickt werden deklariert. Ähnliches wird in dem ZX20 Programm umgesetzt, siehe Abbildung 36. Die Variablen welche von der SPS auf das HMI gesendet werden, sind dabei mit dem Typ VAR_GLOBAL zu initialisieren. Variablen die von dem HMI auf die SPS gesendet werden, sind mit dem Typ VAR_EXTERN zu initialisieren

```
14 VAR_GLOBAL (* Von SPS auf HMI *)
15            Ein : USINT;            (* SPS ist Eingeschaltet *)
16            Aus : USINT;            (* SPS ist Ausgeschaltet *)
17            F6 : USINT := 0;        (* Variable für Feld 7 *)
18            F7 : USINT := 0;        (* Variable für Feld 8 *)
19            F8 : USINT := 0;        (* Varialbe für Feld 9 *)
20            F9 : USINT := 0;        (* Variable für Feld 10 *)
21            Count : USINT := 0;     (* Wert des Textfeldes *)
22 END_VAR;
23
24 VAR_EXTERN (* Von HMI auf SPS *)
25            EinAus : USINT;         (* SPS Ein- oder Ausschalten *)
26            Up : USINT;             (* Signal des Push-Tasters *)
27            CheckR : USINT;         (* Checkbox Variable Schieberegler *)
28            CheckT : USINT;         (* Checkbox Variable Taster *)
29            SchR : USINT;           (* Wert des Schiebereglers *)
30 END_VAR;
```

Abbildung 36: Variablen Deklaration zur PROFINET Kommunikation

Wichtig: Es muss zwischen den Variablen in Codesys und den Variablen in dem ZX20 Programm keine Namensgleichheit gegeben sein. Es kann jedoch der Übersichtlichkeit wegen sinnvoll sein die Variablen gleich zu benennen. Vielmehr kommt es auf die Reihenfolge, in der die Variablen deklariert werden, an. Die Reihenfolge der Variablen muss in Codesys und EX_PRESS 5 immer gleich sein, siehe Abbildung 37.

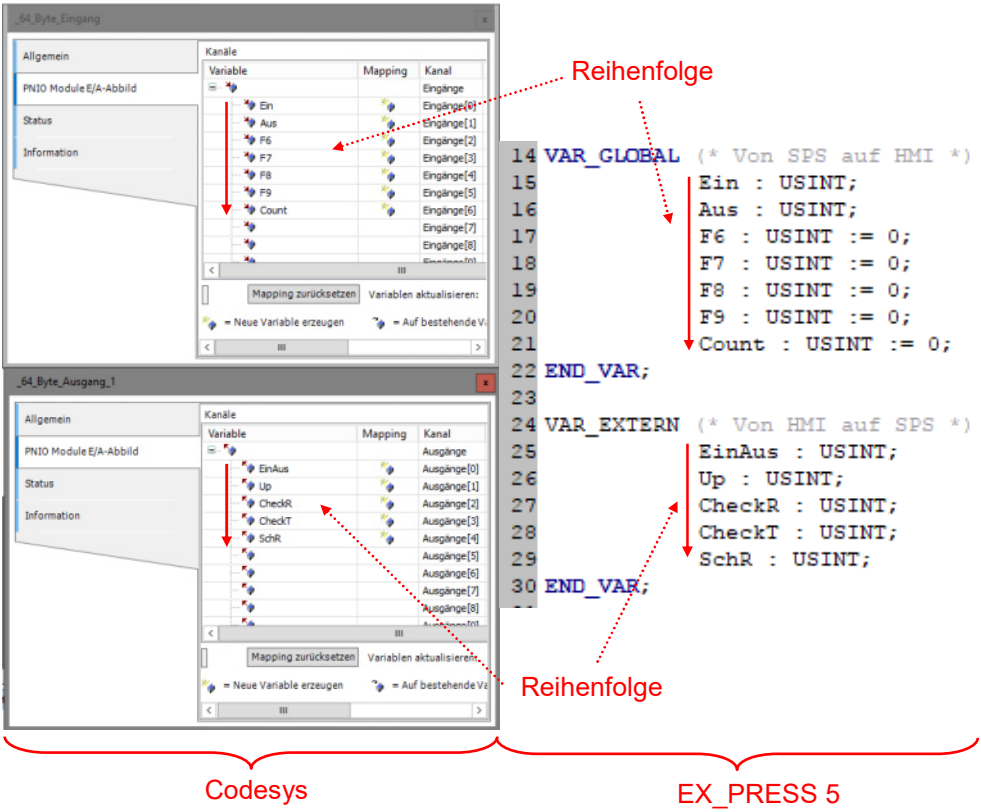


Abbildung 37: Gleiche Reihenfolge der Variablen in Codesys und EX_PRESS 5

5.2.2 Interne Variablen im ZX20 Programm (Zeile 31 bis 45)

Für die Verarbeitung der Daten die vom HMI an die ZX20 gesendet werden ist das Anlegen von Variablen zur internen Verarbeitung unumgänglich. Dazu gehören zunächst die Initialisierung der Ausgangsvariablen der SPS, siehe Abbildung 38 Zeile 32 bis 35. Des Weiteren kommen Variablen für die Taster Erfassung und die Berechnung der Schieberegler Einstellung hinzu, siehe Abbildung 38 Zeile 37 bis 41. Zuletzt werden die Tastervariablen in dem VAR_ALIAS Type zusammengefasst, siehe Abbildung 38 Zeile 43 bis 45.

```
32 VAR_OUTPUT (* Ausgaenge der SPS *)
33            Q0; Q1; Q2; Q3; Q4; Q5; Q6; Q7;
34            Q8; Q9; Q10; Q11; Q12; Q13; Q14; Q15;
35 END_VAR;
36
37 VAR
38            Kom : BIT ;            (* Clockvarible für den Taster *)
39            M9, M8, M7, M6 : BIT;  (* Interne Tastervariablen *)
40            N9, N8, N7, N6 : BIT;  (* Interne Schiebereglervariablen *)
41 END_VAR;
42
43 VAR_ALIAS
44            M9_6[M9, M8, M7, M6];  (* Zusammenfassen der internen Tastervariablen *)
45 END_VAR;
```

Abbildung 38: Interne Variablen im ZX20 Programm

5.3 Programmkern Zeile (Zeile 46 bis 147)

Im Programmkern werden die zuvor initialisierten Variablen verarbeitet und dann die berechneten Ergebnisse an das HMI übergeben bzw. an die Ausgänge der SPS gelegt.

5.3.1 Ein und Ausschalten der SPS (Zeile 46 bis 65)

Zunächst soll erfasst werden, ob die SPS über das HMI ein- oder ausgeschaltet ist. Dieser Zustand wird in der Variablen „EinAus“ von dem HMI an die SPS übergeben. Der Zustand der Variable „EinAus“ wird dem HMI über die Variablen „Ein“ und „Aus“ zurückgesendet, siehe Abbildung 39. Je nach Zustand der Variablen „EinAus“ wird der erste („Q0“) bzw. letzte Ausgang („Q1“) der SPS angesteuert.

```
54 IF (EinAus = 1) THEN
55            Ein := 1;
56            Aus := 0;
57            Q0 := 1;
58            Q15 := 0;
59 ELSE
60            Ein := 0;
61            Aus := 1;
62            Q0 := 0;
63            Q15 := 1;
64 END_IF;
```

Abbildung 39: Reaktion auf den Status der „EinAus“ Variabel

5.3.2 Programmierung des Tasters (Zeile 66 bis 85)

In dem Moment wo der Taster (Push) gedrückt wird, wechselt die Variable „Up“ von 0 auf 1. Dieses Signal wird als Clocksignal verwendet. Dazu wird die Variable „Up“ auf die Clockvariable „Kom“ umgeschrieben unter der Bedingung das die SPS eingeschaltet ist, die richtige Checkbox ausgewählt ist und das der Taster gedrückt ist, siehe Abbildung 40 Zeile 72 bis 78. Die Clockvariable „Kom“ ist mit dem Clockeingang der Variablen „M6“ bis „M9“ verbunden, siehe Abbildung 40. Somit wird über die Variable „Up“ das in Zeile 81 bis 84 programmierte Schieberegister angesteuert, siehe Abbildung 40 und Abbildung 41. Das Schieberegister verwenden dabei die internen Variablen „M6“ bis „M9“.

```
72 IF ( CheckT = 1 AND CheckR = 0 AND UP = 1 AND EinAus
73             Kom := 1;
74 ELSE
75             Kom := 0;
76 END_IF;
77
78 M9_6.CLK := Kom;
79
80 (* Schieberegister *)
81 M6 := (NOT M6 AND NOT M7 AND NOT M8 AND NOT M9) OR M9
82 M7 := M6;
83 M8 := M7;
84 M9 := M8;
```

Abbildung 40: Programmierung des Taster

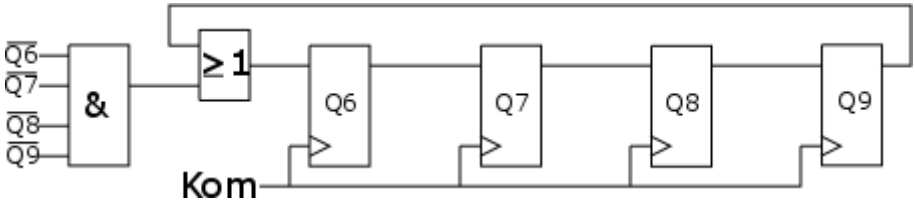


Abbildung 41: Logikskizze des Schieberegisters

5.3.3 Programmierung des Schiebereglers (Zeile 86 bis 98)

In der Schieberegler Variable „SchR“ ist der Status des Schiebereglers gespeichert, siehe Abbildung 42. Je nach Position des Reglers kann die Variable „SchR“ die Werte 0 bis 3 annehmen. Dies wird in CASE-Abfrage analysiert, siehe (Zeile 91). Je nach Status werden die internen Variablen „N6“ bis „N9“ aktiviert.

```
91 CASE SchR OF
92              0: N6 := 1; N7 := 0; N8 := 0; N9 := 0;
93              1: N6 := 0; N7 := 1; N8 := 0; N9 := 0;
94              2: N6 := 0; N7 := 0; N8 := 1; N9 := 0;
95 ELSE
96               N6 := 0; N7 := 0; N8 := 0; N9 := 1;
97 END_CASE;
```

Abbildung 42: Statuserfassung des Schiebereglers

5.3.4 Programmierung des Textfeldes und der HMI-Felder (Zeile 99 bis 123)

Die Felder die auf dem HMI aktiviert werden, um den ausgewählten Ausgang der SPS anzuzeigen sind in den Variablen „F6“ bis „F9“ hinterlegt. Das Aktivieren der Felder kann sowohl durch den Schieberegler als auch durch den Taster erfolgen. Dies ist davon abhängig welche Checkbox ausgewählt wurde und ob die SPS eingeschaltet ist, siehe Abbildung 43. Die internen Variablen des Tasters und des Schiebereglers werden abgefragt und je nach Satus die jeweilige Feldvariable aktiviert.

Das Textfeld wird über die Variable „Count“ angesteuert. Je nach dem welches Feld auf dem HMI aktiviert ist nimmt die Variable unterschiedliche Zahlen an. Bei ausgeschalteter SPS werden keine Ausgänge gesetzt und die Variable „Count“ mit Null initialisiert, siehe Zeile 117 bis 121.

```
104 IF (((M6 = 1 AND CheckT = 1) XOR (N6 = 1 AND CheckR = 1)) AND EinAus = 1) THEN
105             F6 := 1; F7 := 0; F8 := 0; F9 := 0;
106             Count := 7;
107 ELSIF (((M7 = 1 AND CheckT = 1) XOR (N7 = 1 AND CheckR = 1)) AND EinAus = 1) THEN
108             F6 := 0; F7 := 1; F8 := 0; F9 := 0;
109             Count := 8;
110 ELSIF (((M8 = 1 AND CheckT = 1) XOR (N8 = 1 AND CheckR = 1)) AND EinAus = 1) THEN
111             F6 := 0; F7 := 0; F8 := 1; F9 := 0;
112             Count := 9;
113 ELSIF (((M9 = 1 AND CheckT = 1) XOR (N9 = 1 AND CheckR = 1)) AND EinAus = 1) THEN
114             F6 := 0; F7 := 0; F8 := 0; F9 := 1;
115             Count := 10;
116 ELSE
117             F6 := 0;
118             F7 := 0;
119             F8 := 0;
120             F9 := 0;
121             Count := 0;
122 END_IF;
```

Abbildung 43: Programmierung des Textfeldes und der HMI Felder

5.3.5 Programmierung der SPS-Ausgänge (Zeile 124 bis 147)

Zuletzt werden die SPS-Ausgänge konfiguriert. Dazu werden die internen Variablen des Tasters „M6“ bis „M9“ und des Schiebereglers „N6“ bis „N9“ auf die SPS-Ausgänge „Q6“ bis „Q9“ gesetzt, siehe Abbildung 44. Dies ist abhängig davon welche Checkbox ausgewählt ist und ob die SPS eingeschaltet wurde.

```
128 IF (CheckR = 0 AND CheckT = 1 AND EinAus = 1) THEN        (* Taster *)
129             Q6 := M6;
130             Q7 := M7;
131             Q8 := M8;
132             Q9 := M9;
133 ELSIF (CheckR = 1 AND CheckT = 0 AND EinAus = 1) THEN     (* Schieberegler *)
134             Q6 := N6;
135             Q7 := N7;
136             Q8 := N8;
137             Q9 := N9;
138 ELSE
139             Q6 := 0;
140             Q7 := 0;
141             Q8 := 0;
142             Q9 := 0;
143 END_IF;
```

Abbildung 44: Schreiben der SPS-Ausgänge

6. Notizen