# Assignment REG23-AS-002

## Steuerungs- und Regelungstechnik

Thema „Bohreinrichtung“


| | | | |
|---|---|---|---|
| Name, Vorname: | Papa Schlumpf | Studiengang: | Mechatronik - Bachelor of Engineering (B. Eng.) |
| Immatrikulationsnr.: | 123456789 | | |
| Adresse: | Am Pilz 01<br>1234 Schlumpfhausen | Modul: | REG23 |
| | | Abgabe am: | 1. Januar 2022 |
| E-Mail: | test@test.com | Dozent: | Heimerdinger |


AKAD

# Inhaltsverzeichnis

**Abbildungsverzeichnis** III

**Tabellenverzeichnis** IV

**Abkürzungsverzeichnis** V

**1. Einleitung** 1

**2. Grundlagen der Steuerungstechnik** 2
2.1. Sensoren . . . 2
2.2. Aktuatoren . . . 3
2.3. Speicherprogrammierbare Steuerung . . . 4
2.4. CoDeSys . . . 5
2.5. ISO/OSI-Referenzmodell . . . 6

**3. Bohreinrichtung** 8
3.1. Lastenheft . . . 8
3.2. Programmablaufplan . . . 9
3.3. Programmierung . . . 11
3.4. Simulation . . . 12

**4. Zusammenfassung** 14

**Anhang** VI
A. Programmablaufplan . . . VI
B. PLC_PRG . . . VII

**Literaturverzeichnis** X

# Abbildungsverzeichnis

1. magnetischer Sensor . . . . . 2
2. doppeltwirkender Pneumatikzylinder . . . . . 3
3. Aufbau einer SPS . . . . . 4
4. Benutzeroberfläche CoDeSys . . . . . 6
5. ISO/OSI-Referenzmodell . . . . . 7
6. Bohreinrichtung . . . . . 8
7. Ausschnitt Programmablaufplan . . . . . 10
8. Design der Simulation . . . . . 12
9. gesamter Programmablaufplan . . . . . VI

# Tabellenverzeichnis

1. Adressen der Sensoren und Aktoren . . . . . 9
2. Globale Variablendeklaration . . . . . 11

# Abkürzungsverzeichnis

| | |
|---|---|
| **SPS** | speicherprogrammierbare Steuerung |
| **VPS** | verbindungsprogrammierbare Steuerung |
| **CoDeSys** | Controller Development System |
| **ST** | strukturierter Text |
| **AS** | Ablaufsprache |
| **FUP** | Funktionsplan (auch Funktionsbausteinsprache) |
| **KOP** | Kontaktplan |
| **AWL** | Anweisungsliste |
| **CFC** | Continous Function Chart |
| **OSI** | Open Systems Interconnect |
| **ISO** | International Standards Organization |
| **LED** | engl. light emitting diode, dt. Lichtemittierende Diode, auch Leuchtdiode |

# 1. Einleitung

Im Zeitalter der Industrie 4.0 spielen Steuerungs- und Automatisierungssysteme eine zentrale Rolle. Bewegungsvorgänge und Zustand der Anlage werden in digital verwertbare Daten umgewandelt und online ausgewertet. Erst diese Informationen machen es möglich, dass Maschinen autonom arbeiten können. Für die Erfassung und Verarbeitung der Daten gibt es verschiedene technische Möglichkeiten.[1] Das Assignment legt den Fokus auf die Verarbeitung der erfassten Informationen mithilfe einer speicherprogramierbaren Steuerung (SPS) und dessen Programmierung.
Ziel der Arbeit ist es, einen vorher definierten Bewegungsablauf einer Bohreinrichtung in der Programmierumgebung Controller Development System (CoDeSys) zu programmieren. Anschließend soll der Ablauf in einer Simulation dargestellt werden.
Im ersten Schritt werden Grundlagen der Steuerungstechnik dargestellt. Hierzu beschreibt der erste Teilabschnitt die für die Aufgabenstellung benötigten Sensoren und Aktoren und die grundlegenden Eigenschaften einer SPS. Zusätzlich wird die Programmierumgebung CoDeSys vorgestellt. Außerdem wird auf das OSI-Schichtmodell eingegangen und welche Ebenen bei einer SPS Anwendung finden.
Anschließend wird der Bewegungsablauf der Bohreinrichtung realisiert. Hierfür ist zuerst das Lastenheft entsprechend der Aufgabenstellung beschrieben. Im Anschluss wird ein Programmablaufplan entworfen. Darauf aufbauend kann die eigentliche Programmierung erstellt werden. Zur Überprüfung der Funktionsfähigkeit des Programms wird der Bewegungsablauf in einer Simulation überprüft.
Das Assignment endet mit der Zusammenfassung, in der die Erkenntnisse dieser Arbeit reflektiert werden und eine kritische Auseinandersetzung mit den Ergebnissen erfolgt.

[1] vgl. Seitz, 2021, S. 24

# 2. Grundlagen der Steuerungstechnik

## 2.1. Sensoren

Sensoren erfassen Messgrößen der Umwelt und wandeln diese in elektrische, hydraulische oder pneumatische Ausgangsgrößen um. Hierzu werden physikalische, chemische oder biologische Effekte genutzt. Das Ausgangssignal kann analog (z.B. PTC-Temperatursensor) oder digital (z.B. Hallsensor) ausgegeben werden. Moderne Sensoren besitzen meistens zusätzlich Wandler, Verstärker und Signalvorverarbeitungselemente. Sie werden auch als integrierte Sensoren bezeichnet.[2]

Es gibt eine Vielzahl verschiedener Sensoren. In diesem Assignment wird nur auf die für die Bohrvorrichtung benötigten Sensoren eingegangen. Hierbei müssen die Anfangs- und Endpositionen der Zylinder erfasst werden. Das kann mechanisch, kapazitiv, induktiv, über Widerstandsänderungen, mithilfe des Hall-Effekts, akustisch, optisch oder magnetisch erfolgen.[3]

Magnetische Zylindersensoren stellen die einfachste und effektivste Variante dar, um die Endanschläge des Zylinders zu erfassen.

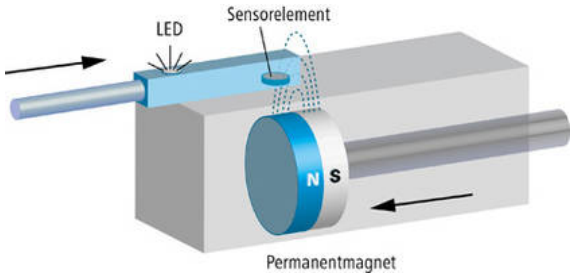



Abbildung 1: magnetischer Sensor [4]

In Abbildung 1 ist die Funktionsweise eines solchen Sensors dargestellt. Beim Erreichen der jeweiligen Position erfolgt durch den magnetischen Kolben eine Magnetfeldänderung. Diese wird vom Sensorelement registriert. Je nach Schaltlogik gibt der integrierte Sensor ein High oder Low als Ausgangsgröße aus, welches von einer Steuerungsanlage verarbeitet werden kann.[5]

[2] vgl. Hering; Endres; Gutekunst, 2021, S. 389 ff.
[3] vgl. ebd., S. 394 ff.
[4] o.V., 2021
[5] vgl. ebd.

## 2.2. Aktuatoren

Aktuatoren (mittellateinisch actuare = sich betätigen, auch Aktor genannt[6]) bestehen im Allgemeinen aus einem Energiesteller, der aus der Stellgröße zusammen mit einer Hilfsenergie einen Energiefluss erzeugt. Dieser wird durch einen Energiewandler in eine andere physikalische Größe, meist eine mechanische Bewegung, überführt.[7]
Wie bei den Sensoren gibt es auch bei den Aktuatoren die unterschiedlichsten Arten, welche auf verschiedenen Wirkprinzipien beruhen. Hierauf kann im Rahmen des Assignments nicht näher eingegangen werden.
Für die Bohreinrichtung wird zum einen ein Elektromotor benötigt. Dieser wandelt einen elektrischen Strom in eine rotatorische Bewegung um. Zum anderen werden zwei fluidische Aktoren benötigt, die die Bewegung einer Flüssigkeit oder eines Gases in eine mechanisch translatorische Bewegung umwandeln. Zum Einsatz kommen laut Aufgabenstellung zwei doppelt wirkende Pneumatikzylinder mit einseitiger Kolbenstange und elektromagnetischer Betätigung.

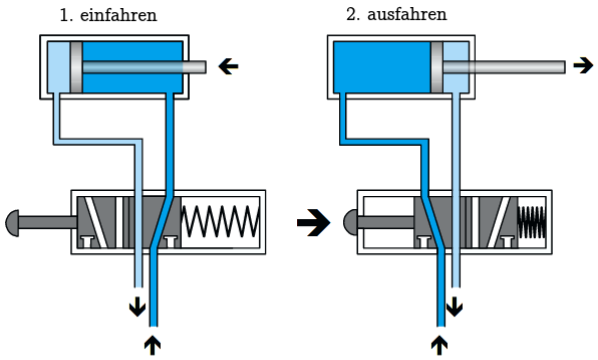



Abbildung 2: Funktionsweise eines doppeltwirkenden Pneumatikzylinders [8]

In Abbildung 2 wird die Ansteuerung der Zylinder mit einem mechanisch betätigtem 5/2-Wegeventil dargestellt. Links wird der Einfahrprozess ersichtlich, die rechte Darstellung zeigt den Ausfahrvorgang. Die dunkelblau eingefärbten Leitungen sind jeweils mit Druck beaufschlagt, das bewirkt eine Bewegung des Kolbens aufgrund eines Differenzdruckes. Eine messtechnische Erfassung des Druckes ist ebenso

[6] o.V., Aktuator
[7] vgl. Heimann; Albert; Ortmaier; Rissing, 2016, S. 30 ff.
[8] in Anlehnung an: Ebel, 2000

möglich. Pneumatisch gesteuerte Aktuatoren weisen eine schlechte Regelbarkeit der (Ausfahr-)Geschwindigkeit auf. Deshalb sind die Zylinder in der Aufgabenstellung zusätzlich mit einer Ölbremseinheit ausgestattet.[9]

## 2.3. Speicherprogrammierbare Steuerung

Eine SPS ist eine programmierbare elektronische Einheit, welche Maschinen und Anlagen steuern und regeln kann. Ihr Vorgänger ist die verbindungsprogrammierbare Steuerung (VPS). Im Unterschied zur SPS wurde der Programmablauf durch eine feste Verdrahtung der Logikbausteine realisiert. Dadurch ist die VPS bei komplexen Systemen sehr aufwändig zu realisieren. Zusätzlich ist die Anpassung an veränderte Bedingungen der Steuerung nicht oder nur schwer möglich.[10]

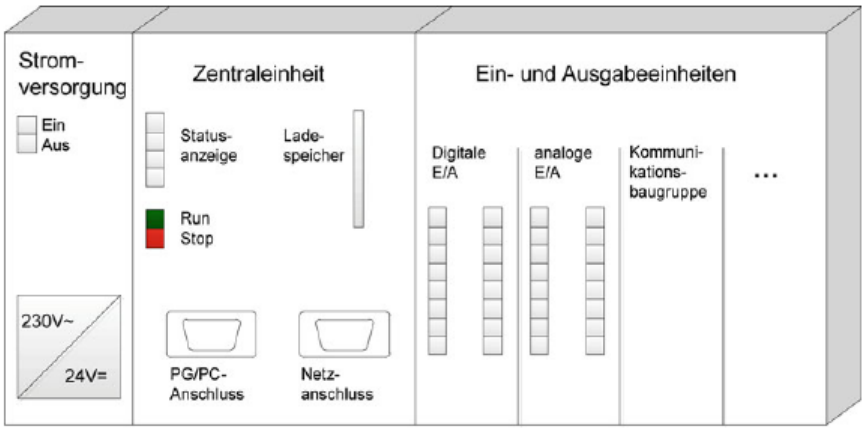



Abbildung 3: Aufbau einer SPS [11]

Der grundlegende Aufbau einer SPS besteht in der Minimalausführung aus einer Stromversorgung, einer Zentraleinheit und einer Ein- und Ausgabeeinheit (siehe Abbildung 3). Die Stromversorgungseinheit wandelt die Netzspannung, meist 230 V Wechselspannung, in 24V Gleichspannung um. Die Zentraleinheit ist für die Verarbeitung und Speicherung des Ablaufprogrammes zuständig. Zusätzlich kann ein PC angeschlossen werden, um Anwenderprogramme einzulesen oder Fehler zu suchen. An der Ein- und Ausgabeeinheit werden die Sensoren und Aktuatoren angeschlossen. Zusätzlich

[9] vgl. Heimann; Albert; Ortmaier; Rissing, 2016, S. 54 f.
[10] vgl. Hering; Endres; Gutekunst, 2021, S. 755
[11] A. Böge; W. Böge, 2021

kann die Einheit mit Kommunikationsmodulen, z.B. einem Datenbussystem oder einer WLAN-Einheit erweitert werden.[12] Die Programmbearbeitung erfolgt zyklisch. „Jeder Zyklus beginnt mit dem Einlesen der aktuellen Signalzustände der Eingänge [...] und endet mit der Ausgabe der Signale an die Ausgänge [...]“.[13] Die benötigte Zeit für einen Durchlauf wird auch als Zykluszeit bezeichnet. Diese muss den Echtzeitbedingungen der SPS entsprechend ausreichend klein sein.[14]

## 2.4. CoDeSys

CoDeSys ist eine Entwicklungsumgebung zur Programmierung und Simulation von SPS-Steuerungen.[15] Für das Assignment wird die Version V3.5 SP17 Patch 2, 64-bit in einer virtuellen Maschine (Host: Fedora 33, KDE Plasma, Gast: Windows 10 Pro, Version 20H2) verwendet.

CoDeSys bietet die Möglichkeit, die Programmierung in unterschiedlichen Sprachen umzusetzen. Strukturierter Text (ST), Ablaufsprache (AS), Funktionsplan (auch Funktionsbausteinsprache) (FUP), Kontaktplan (KOP), Anweisungsliste (AWL) und Continous Function Chart (CFC) sind die zur Auswahl stehenden Programmiersprachen. ST ist eine textbasierte Implementierungssprache, alle anderen Sprachen verwenden einen grafischen Editor. Zusätzlich kann die Syntax und der Quellcode auf Fehler überprüft werden.[16]

In Abbildung 4, Seite 6 ist die Benutzeroberfläche von CoDeSys abgebildet. Unter (1) ist der Gerätebaum und -editor zu sehen. Die Datei „PLC_PRG (PRG)“ ist das Hauptprogramm. Bereich (2) und (3) stellen den eigentlichen Editorbereich dar. Der obere Abschnitt ist ein einfacher Texteditor (2). Hier sind unter anderem die lokalen Variablen der Funktionsbausteine sowie Ein- und Ausgänge der im unteren Bereich zu sehenden grafischen Darstellung der Schrittkette deklariert. Der grafische Editor (3) besteht aus einzelnen Netzwerken, die jeweils einen einzelnen Schritt darstellen. Im Bereich (4) können die einzelnen Operatoren und Funktionsbausteine ausgewählt

[12] vgl. A. Böge; W. Böge, 2021, S. 1497 f.
[13] Wellenreuther; Zastrow, 2005, S. 13
[14] vgl. ebd., S. 13
[15] vgl. o.V., 2004, S. 1
[16] vgl. o.V., Benutzerhandbuch CoDeSys, Kapitel „Programmiersprachen und ihre Editoren“ und „Befehl ’Code erzeugen’“

werden. Bei der Erstellung einer Simulation befinden sich in diesem Bereich auch die einzelnen Steuerungselemente wie z.B. Lampen oder Schalter.[17]

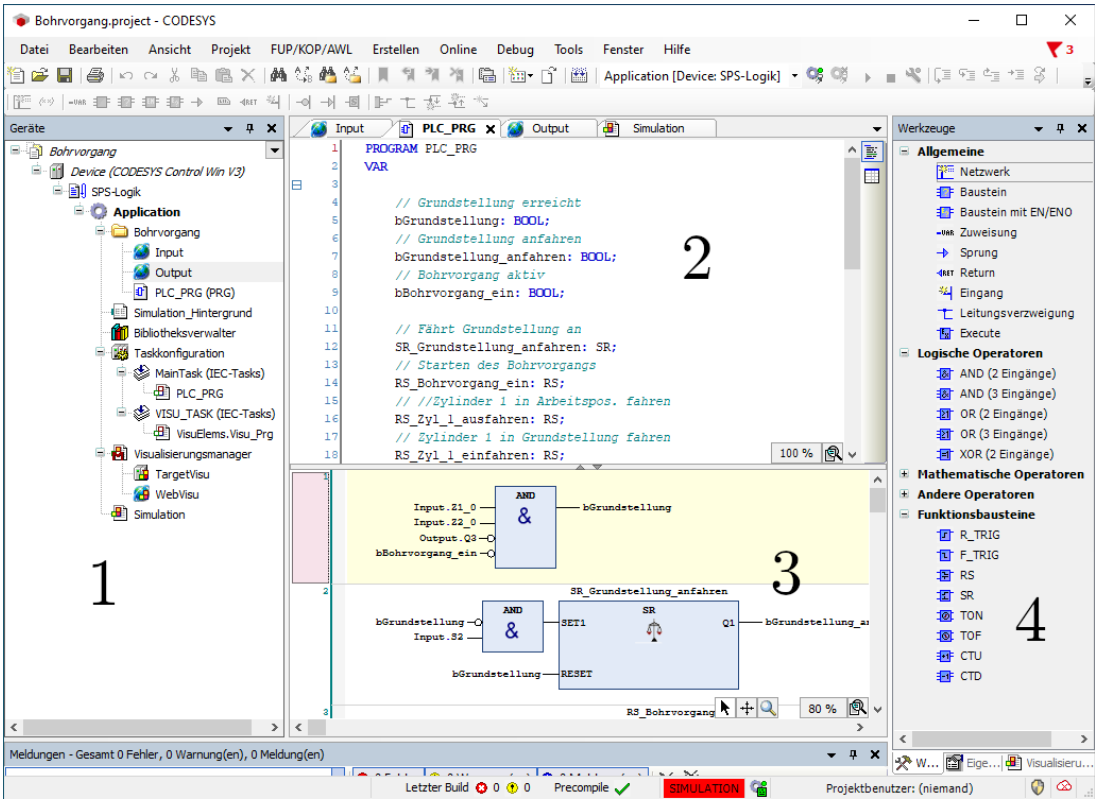



Abbildung 4: Benutzeroberfläche CoDeSys mit geöffnetem Projekt (eigener Entwurf)

## 2.5. ISO/OSI-Referenzmodell

Das Open Systems Interconnect (OSI)-Referenzmodell wurde von der International Standards Organization (ISO) für den einheitlichen Aufbau von Netzwerken und deren Schnittstellendefinition entworfen. Es umfasst sieben Schichten, diese sind in Abbildung 5, Seite 7 dargestellt.[18] Die Daten gelangen vom Endsystem A, angefangen in Schicht sieben, schrittweise abwärts bis zur Schicht eins, wo die Daten physikalisch übertragen werden. Im Endsystem B gelangen die Daten schrittweise von Schicht eins aufwärts bis zur Anwendungsschicht.[19]

[17] vgl. o.V., Benutzerhandbuch CoDeSys, Kapitel „Ihr erstes CODESYS-Programm“ sowie „Gerätebaum und Geräteeditor“
[18] vgl. Heimann; Albert; Ortmaier; Rissing, 2016, S. 209
[19] vgl. o.V., OSI-Schichtenmodell

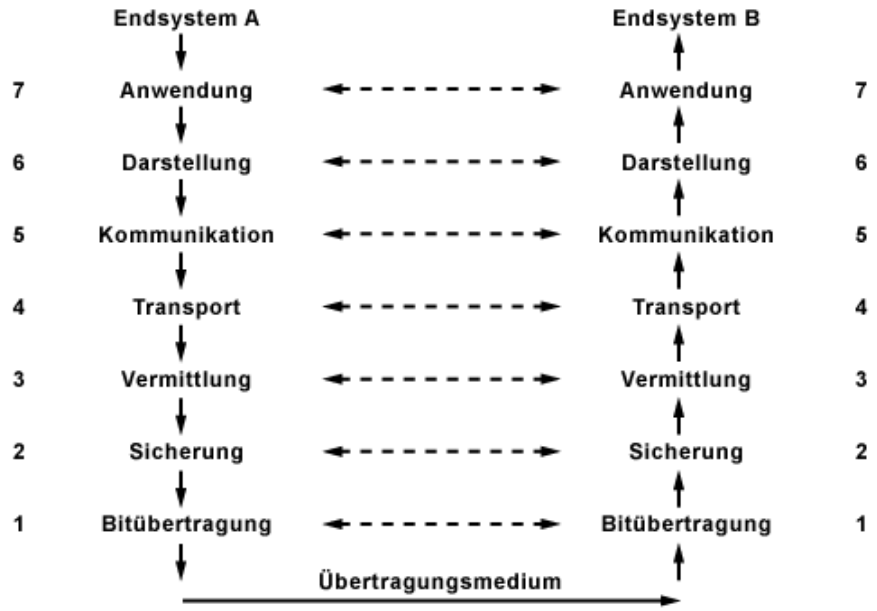



Abbildung 5: ISO/OSI-Referenzmodell nach ISO/IEC 7498 1:1994 [20]

Nicht alle Schichten müssen zwangsläufig verwendet werden. So sind in den Kommunikationssystemen der Automatisierung meistens nur die Schichten eins, zwei und sieben vorhanden. In der Bitübertragungsschicht werden die elektrischen, mechanischen und funktionalen Anforderungen festgelegt. So werden z.B. die Taktrate, Leitungscodierungen oder Steckerform festgelegt. In der Sicherungsschicht werden die Adressierung, Fehlererkennungsmechanismen und Maßnahmen zur sicheren Übertragung von Datenpaketen festgelegt. Die siebte Schicht stellt das Mensch-Maschine-Interface dar. Hier wird dem Anwender erst ermöglicht, die Maschine oder Anlage zu bedienen. Eine Dateneingabe und -ausgabe kann ebenso erfolgen.[21]

[20] o.V., OSI-Schichtenmodell
[21] vgl. Heimann; Albert; Ortmaier; Rissing, 2016, S. 209; vgl. o.V., OSI-Schichtenmodell

# 3. Bohreinrichtung

## 3.1. Lastenheft

Die in Abbildung 6 dargestellte Bohreinrichtung verfügt über zwei Schalter S1 und S2. Diese stellen die Bedienung dar. Wird der Schalter eins betätigt, soll der automatische Ablauf aus der Grundstellung heraus beginnen. Zuerst bewegt sich Zylinder eins in Arbeitsstellung und spannt das Werkstück ein. Anschließend startet der Bohrer und Zylinder zwei fährt aus und ermöglicht den eigentlichen Bohrvorgang. Am tiefsten Punkt soll der Bohrkopf für zwei Sekunden im Werkstück verbleiben und anschließend wieder einfahren. Zum Schluss fährt Zylinder eins wieder in die Grundstellung und gibt damit das Werkstück frei. Schalter S2 dient dazu, die Anlage aus jeder beliebigen Position in die Grundstellung zu bewegen.

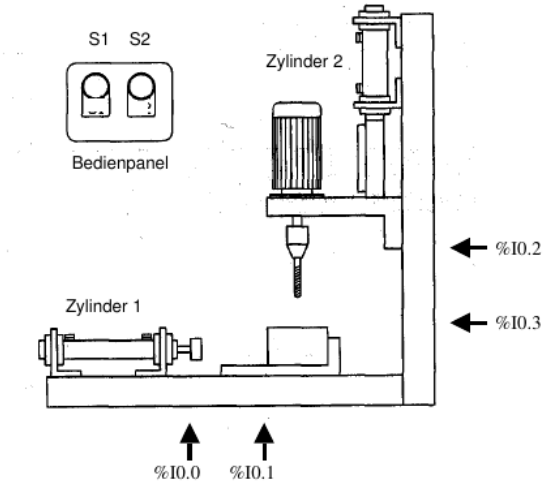



Abbildung 6: Modell der Bohreinrichtung [22]

In Tabelle 1, Seite 9 sind die Adressen der Ein- und Ausgänge festgelegt. Hierbei steht %Qx.X für Aktuatoren (Ausgang) und %Ix.X für Sensoren (Eingang). In der Aufgabenstellung wurde der Bohrantrieb nicht als Aktuator erwähnt, eine Adresszuweisung erfolgt ebenso nicht. Deshalb wird die Adresse auf %Q1.4 festgelegt. Zudem würde der Bohrer laut Aufgabenstellung ohne Drehbewegung in das Werkstück fahren. Das hätte zur Folge, dass Bohrer und Werkstück beschädigt bzw. zerstört werden. Aus diesem

[22] Riege, Bohreinrichtung

Grund wurde im oberen Abschnitt festgelegt, dass vor dem Betätigen des Zylinders zwei der Bohrantrieb bereits aktiv sein soll.

| Benennung | | Adresse |
|---|---|---|
| Schalter | S1 | %I0.4 |
| | S2 | %I0.5 |
| Zylinder 1 | Anfangspos. | %I0.0 |
| | Endpos. | %I0.1 |
| | ausfahren | %Q1.0 |
| | einfahren | %Q1.1 |
| Zylinder 2 | Anfangspos. | %I0.2 |
| | Endpos. | %I0.3 |
| | ausfahren | %Q1.2 |
| | einfahren | %Q1.3 |
| Bohrantrieb | | %Q1.4 |

Tabelle 1: Adressen der Sensoren und Aktoren

Die beschriebene Schrittabfolge soll mithilfe der Entwicklungsumgebung CoDeSys in FUP erstellt werden. Anschließend soll das erstellte Programm in einer Simulation auf Funktion überprüft werden. Auf weitere technische und wirtschaftliche Aspekte, die für ein Lastenheft notwendig sind, wird nicht weiter eingegangen, da es für das Assignment nicht gefordert wird.

## 3.2. Programmablaufplan

Steuerungs- und Regelabläufe lassen sich in unterschiedlichen Modellen darstellen. Schalt- oder Ablaufpläne, Diagramme und Graphen sind Beispiele zur unterschiedlichen Visualisierung von technischen Abläufen. Je nach Umfang des zu realisierenden Systems sind in der Regel mehrere verschiedene Modelle notwendig.[23]

Der im Lastenheft beschriebene Bohrvorgang wird in einem Programmablaufplan grafisch beschrieben. Ein Ausschnitt ist in Abbildung 7, Seite 10 dargestellt. Für die Erläuterung der Verschaltung der Aktoren, Sensoren und SPS-Einheit ist ein Stromlaufplan notwendig. Das ist in diesem Assignment jedoch nicht gefordert.

[23] vgl. A. Böge; W. Böge, 2021, S. 1446

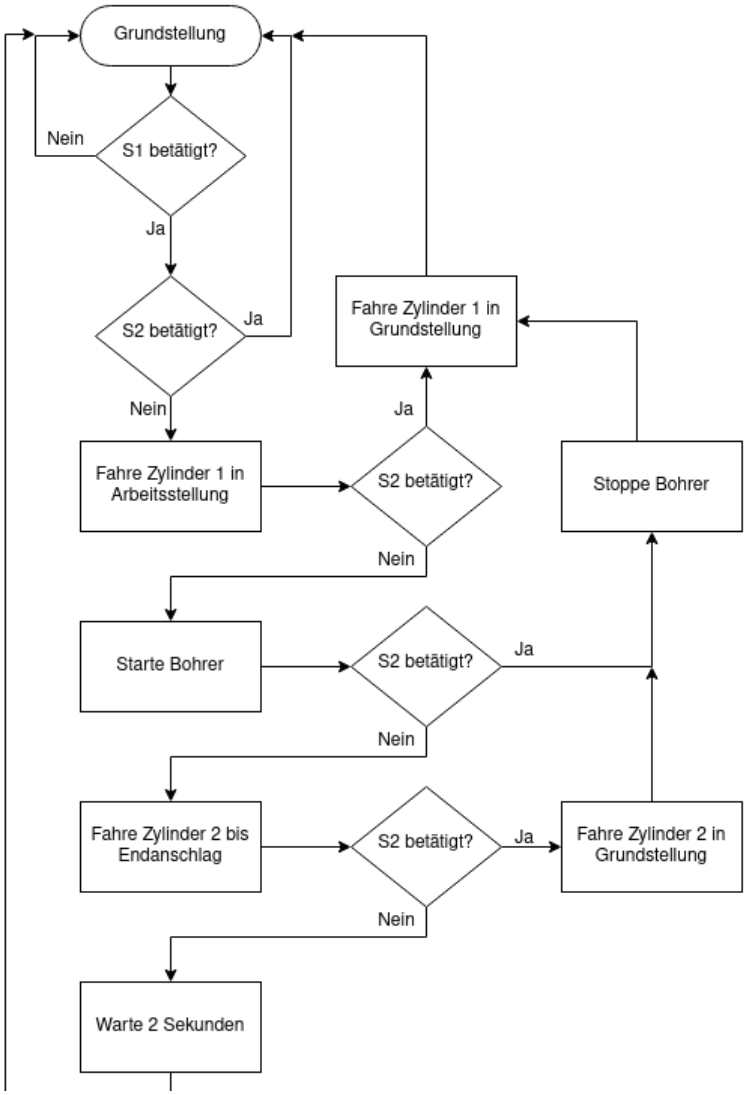



Abbildung 7: Ausschnitt vom Programmablaufplan (eigener Entwurf)

Die Startbedingung für das System ist die Grundstellung aller Aktoren. Durch das Drücken von S2 kann die Grundstellung angefahren werden. Nach dem Auslösen von S1 kann der Ablauf aus der Grundstellung heraus gestartet werden. Während jeder Anweisung wird überprüft, ob S2 betätigt wird. Ist das der Fall, wird die Schrittkette rückwärts durchlaufen, bis die Grundstellung erreicht ist. Nach der Anweisung *„Warte 2 Sekunden“* ist die Abfrage von S2 nicht mehr notwendig, da das System bereits in die Grundstellung fährt.

Der gesamte Programmablaufplan ist im Anhang A auf Seite VI, Abbildung 9 hinterlegt.

## 3.3. Programmierung

Die Erstellung des Codes erfolgt gemäß Lastenheft in der Entwicklungsumgebung CoDeSys. Zuerst werden die Eingangs- und Ausgangsvariablen in den globalen Variablenlisten *Input* und *Output* deklariert. Diese sind in folgender Tabelle 2 als Übersicht dargestellt.

| **Input** | | | **Output** | | |
|---|---|---|---|---|---|
| Bezeichnung | VAR | Typ | Bezeichnung | VAR | Typ |
| Anfangspos. Zylinder 1 | Z1_0 | BOOL | Zylinder 1 ausfahren | Q1_A | BOOL |
| Endpos. Zylinder 1 | Z1_1 | BOOL | Zylinder 1 einfahren | Q1_E | BOOL |
| Anfangspos. Zylinder 2 | Z2_0 | BOOL | Zylinder 2 ausfahren | Q2_A | BOOL |
| Endpos. Zylinder 2 | Z2_1 | BOOL | Zylinder 2 einfahren | Q2_E | BOOL |
| Schalter S1 | S1 | BOOL | Bohren | Q3 | BOOL |
| Schalter S2 | S2 | BOOL | | | |

Tabelle 2: Globale Variablendeklaration

Der Aufruf im Hauptprogramm *PLC-PRG* erfolgt z.B. durch den Befehl:

```
Input.Z1_0
```

Die eigentliche Programmierung wird in der *PLC-PRG-Datei* mithilfe der Programmiersprache FUP erstellt. Im jeweiligen Netzwerk werden Funktionsbausteine platziert, die mit lokalen Variablen versehen werden. Im oberen Texteditor werden die Bausteinbezeichnungen automatisch erstellt. Der gesamte Schrittkettenablauf ist im Anhang B, Seite VII dargestellt.

Netzwerk ein bis drei stellt sicher, dass das System in Grundstellung ist und dass der Bohrvorgang nur aus der Grundstellung heraus gestartet werden kann. Netzwerk vier bis acht enthält die Anweisungen und deren Bedingungen für die einzelnen Teilschritte. Für die folgende Simulation ist es erforderlich, dass die Sensoren der Endanschläge Signaländerungen an die SPS zurückgeben. Das ist in den Netzwerken neun bis dreizehn umgesetzt. Wird das Programm auf eine Hardware geladen, muss dieser Bereich auskommentiert werden, ansonsten kommt es zu Fehlfunktionen.

## 3.4. Simulation

Nach erfolgreich erstellter Programmierung muss diese auf Ihre Funktion überprüft werden. Da Fehlerläufe an echten Maschinen finanzielle Belastungen zur Folge haben können, wird eine Simulation erstellt. In der folgenden Abbildung 8 ist die Benutzeroberfläche der Simulationsumgebung dargestellt.

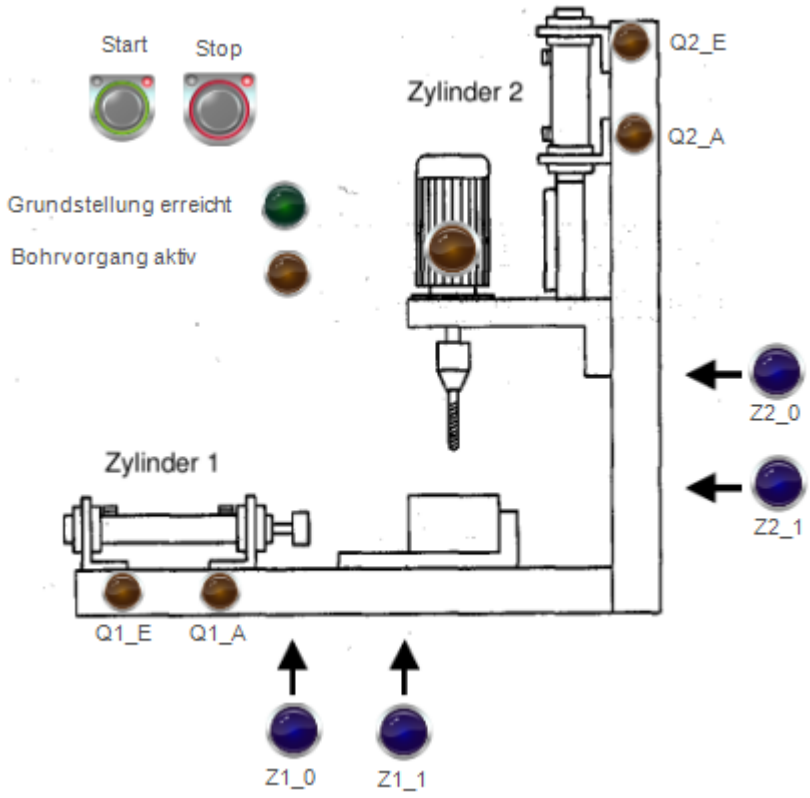



Abbildung 8: Design der Simulation [24]

Im oberen linken Bereich sind die Taster zur Steuerung der Bohrvorrichtung angebracht. Start entspricht dem Schalter *S1*, Stop entspricht *S2*. Darunter sind zwei Leuchtdioden (LED) zu sehen, die entsprechend der Betriebszustände aktiv sind. An den Zylindern sind jeweils zwei gelb leuchtende Kontroll-LED zu sehen. Diese sind aktiv, wenn der jeweilige Zylinder ein oder aus fährt. An der Antriebseinheit ist ebenso eine gelbe LED angebracht, die bei drehendem Bohrer leuchtet. Die Endanschläge der Zylinder, welche durch Sensoren erfasst werden, sind in der Simulation als blaue Leuchtmittel

[24] in Anlehnung an Riege, Bohreinrichtung

visuell dargestellt. Bei Erreichen der entsprechenden Stellung wird die dazugehörige LED eingeschaltet. Durch die Wahl ist zu jedem Zeitpunkt nachvollziehbar, in welchem Betriebszustand die Maschine ist und ob die Schrittkette korrekt abläuft.
Damit die Simulation lauffähig ist, müssen in den globalen Variablenlisten die Adressen der einzelnen Variablen auskommentiert werden. Treten bei der Simulation Fehler im Ablauf ab, muss das Hauptprogramm entsprechend verbessert werden. In der hier erstellten Steuerung ist das nicht der Fall. Der Programmablauf entspricht der vorher festgelegten Ablaufstruktur. Somit kann in einem möglichen nächsten Schritt das Programm auf eine reale SPS übertragen werden. Hierbei ist zu beachten, dass die deklarierten Adressen der einzelnen Sensoren und Aktoren auch mit der real verkabelten Bohreinrichtung übereinstimmt.

# 4. Zusammenfassung

Ziel der Arbeit ist es, ein Schrittkettensimulationsprogramm unter CoDeSys in FUP zu erstellen und zu simulieren. Hierzu sind im ersten Teil erforderliche Grundlagen dargestellt. Die in der Aufgabenstellung geforderte Einordnung der verwendeten Schichten des OSI-Modells bei einem Regelungssystem wird ebenso erläutert. Im Kapitel *Bohreinrichtung* wird im ersten Schritt anhand der Aufgabenstellung ein Lastenheft erstellt. Darauf aufbauend wird der Programmablauf grafisch in einem Ablaufplan festgehalten. Dieser stellt die Grundlage für die eigentliche Programmierung dar. Zur Überprüfung der Funktionsfähigkeit des Codes wird anschließend eine Simulation der Schrittkette erstellt.

Die Grundlagen zu den Sensoren und Aktoren konnte aufgrund des Umfangs des Assignments nur sehr knapp beschrieben werden. Genauso wurde die Entwicklungsumgebung CoDeSys nur in seiner Grundform erläutert. Die gesamten Funktionen, die für die Programmierung erforderlich sind, konnten nicht beschrieben werden. An dieser Stelle ist auf das Benutzerhandbuch[25] zu verweisen. Die Simulation zeigt, dass die Programmierung den geforderten Schrittkettenablauf erfüllt. Das Ziel des Assignments ist somit erreicht.

Aufgrund von fehlenden Spezifikationen entsprechend der Maschinenrichtlinien und nicht vorhandenen Sicherheitseinrichtungen für den Maschinenführer wird die Bohreinrichtung jedoch in der Industrie keine Anwendung finden. Dies sollte bei einer möglichen Weiterentwicklung umgesetzt werden. Zusätzlich ist eine Vorschub- und Drehzahlregelung empfehlenswert, um die Bohreinrichtung für verschiedene Werkstoffe und Bohrergrößen nutzen zu können.

[25] o.V., Benutzerhandbuch CoDeSys

# Anhang

## A. Programmablaufplan

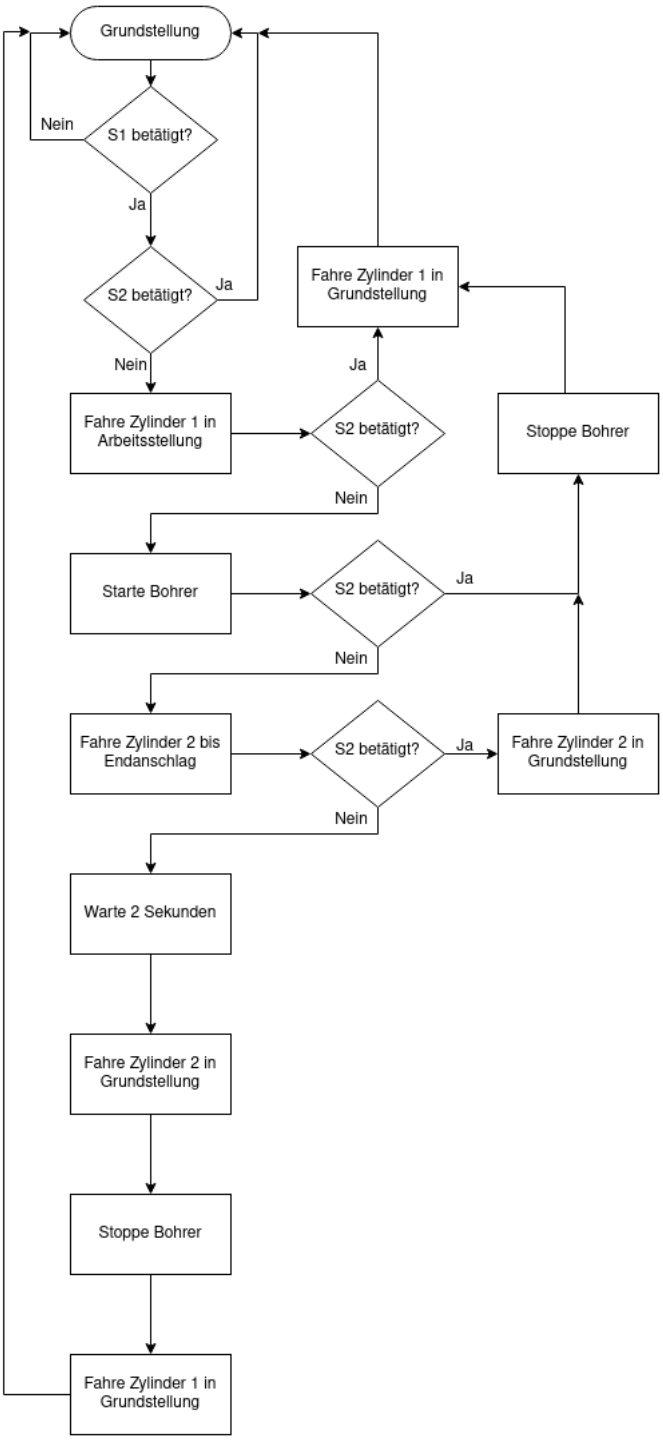
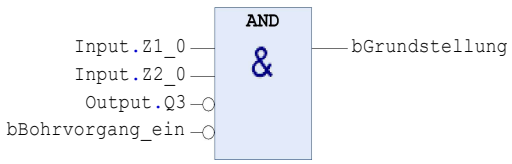



Abbildung 9: Programmablaufplan des automatischen Bohrvorgangs (eigener Entwurf)

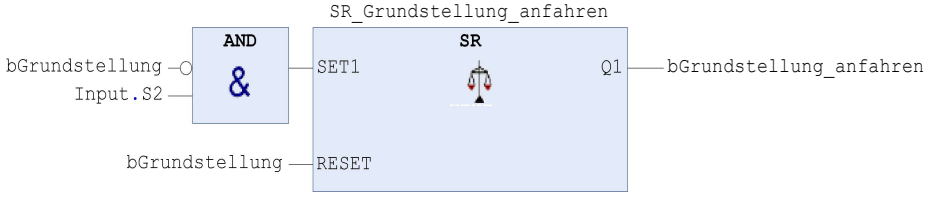
# B. PLC_PRG

```
1   PROGRAM PLC_PRG
2   VAR
3
4       // Grundstellung erreicht
5       bGrundstellung : BOOL ;
6       // Grundstellung anfahren
7       bGrundstellung_anfahren : BOOL ;
8       // Bohrvorgang aktiv
9       bBohrvorgang_ein : BOOL ;
10
11      // Fährt Grundstellung an
12      SR_Grundstellung_anfahren : SR ;
13      // Starten des Bohrvorgangs
14      RS_Bohrvorgang_ein : RS ;
15      // //Zylinder 1 in Arbeitspos. fahren
16      RS_Zyl_1_ausfahren : RS ;
17      // Zylinder 1 in Grundstellung fahren
18      RS_Zyl_1_einfahren : RS ;
19      // Zylinder 2 in Arbeitspos. fahren
20      RS_Zyl_2_ausfahren : RS ;
21      // Zylinder 2 in Grundstellung fahren
22      RS_Zyl_2_einfahren : RS ;
23      // Bohrer eingeschaltet
24      RS_Bohrer_ein : RS ;
25
26      // Wahrheitswert für Bohren
27      bBohren : BOOL ;
28
29      // Wartezeit für Ausfahren/Einfahrem der Zylinder, kann als Vorschub genutzt werden
30      tWartezeit : TIME := T#3S ;
31      // Bohrzeit
32      tBohrzeit : TIME := T#2S ;
33
34      //Laufzeit für Bohrer im ausgefahrenen Zustand
35      TON_Bohrzeit : TON ;
36      //Einschaltverzögerungen für Simulation
37      TON_0 : TON ;
38      TON_1 : TON ;
39      TON_2 : TON ;
40      TON_3 : TON ;
41
42
43  END_VAR
44
```

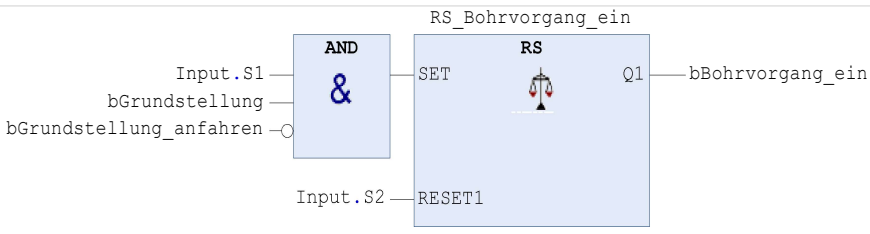
1



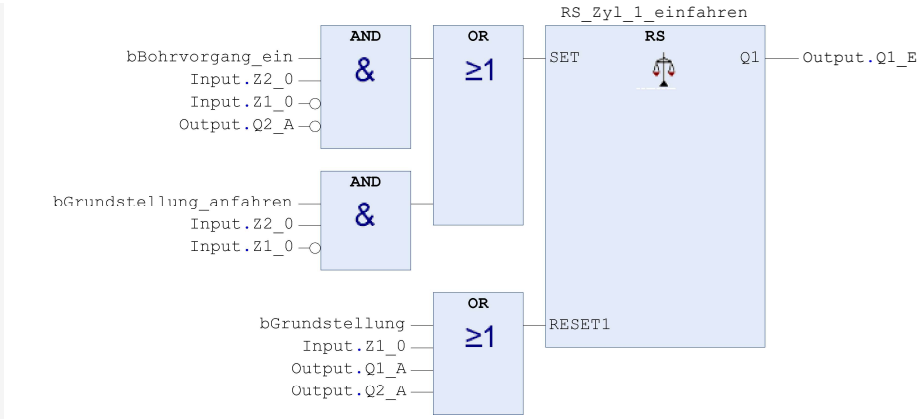

2



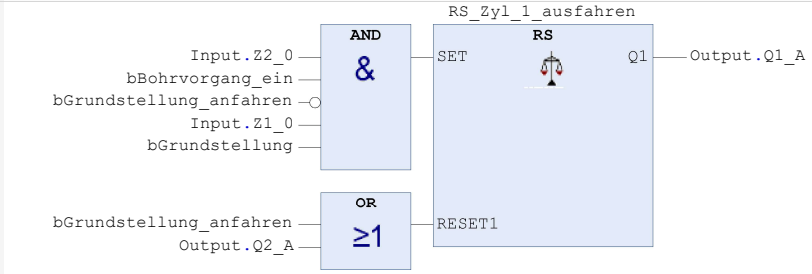

3



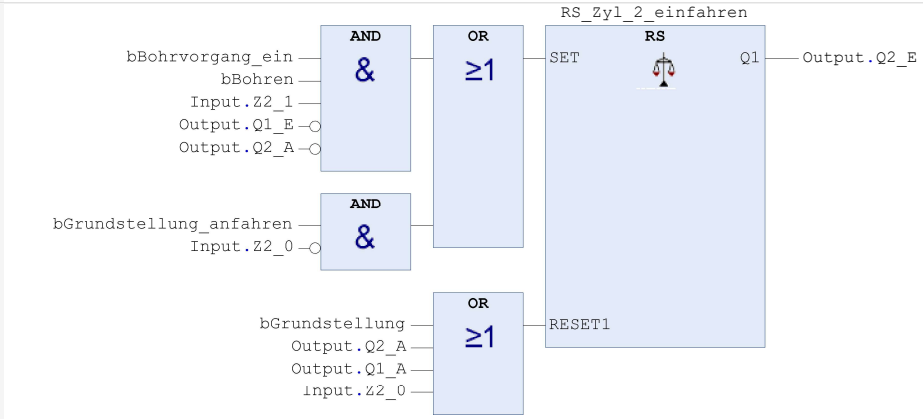
4


RS_Zyl_1_einfahren
AND
&
bBohrvorgang_ein
Input.Z2_0
Input.Z1_0
Output.Q2_A
AND
&
bGrundstellung_anfahren
Input.Z2_0
Input.Z1_0
OR
≥1
OR
≥1
bGrundstellung
Input.Z1_0
Output.Q1_A
Output.Q2_A
RS
SET
Q1
RESET1
Output.Q1_E

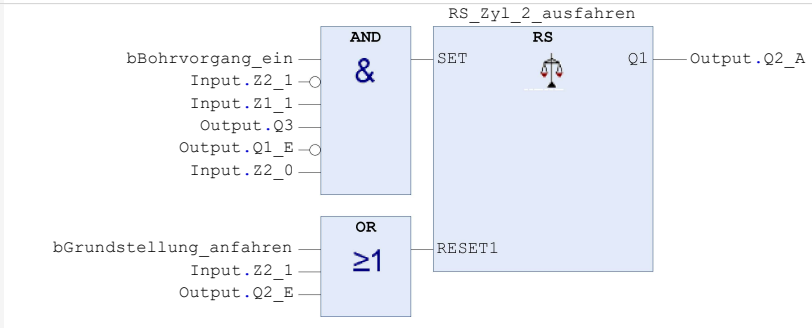
5


RS_Zyl_1_ausfahren
AND
&
Input.Z2_0
bBohrvorgang_ein
bGrundstellung_anfahren
Input.Z1_0
bGrundstellung
OR
≥1
bGrundstellung_anfahren
Output.Q2_A
RS
SET
Q1
RESET1
Output.Q1_A

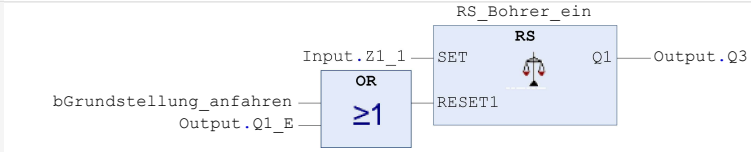
6


RS_Zyl_2_einfahren
AND
&
bBohrvorgang_ein
bBohren
Input.Z2_1
Output.Q1_E
Output.Q2_A
AND
&
bGrundstellung_anfahren
Input.Z2_0
OR
≥1
OR
≥1
bGrundstellung
Output.Q2_A
Output.Q1_A
Input.Z2_0
RS
SET
Q1
RESET1
Output.Q2_E

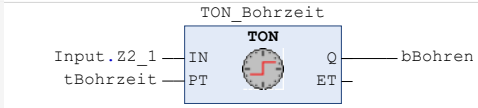
7


RS_Zyl_2_ausfahren
AND
&
bBohrvorgang_ein
Input.Z2_1
Input.Z1_1
Output.Q3
Output.Q1_E
Input.Z2_0
OR
≥1
bGrundstellung_anfahren
Input.Z2_1
Output.Q2_E
RS
SET
Q1
RESET1
Output.Q2_A

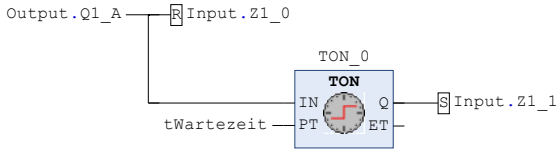
8


RS_Bohrer_ein
Input.Z1_1
OR
≥1
bGrundstellung_anfahren
Output.Q1_E
RS
SET
Q1
RESET1
Output.Q3

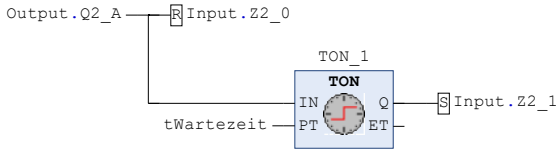
9


TON_Bohrzeit
TON
Input.Z2_1
tBohrzeit
IN
PT
Q
ET
bBohren

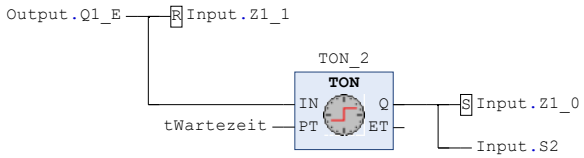
10


Output.Q1_A
R
Input.Z1_0
TON_0
TON
IN
Q
PT
ET
tWartezeit
S
Input.Z1_1

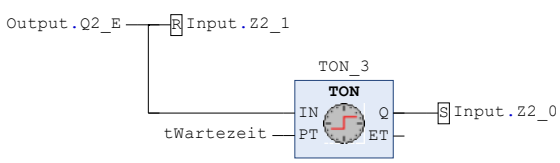
11


Output.Q2_A
R
Input.Z2_0
TON_1
TON
IN
Q
PT
ET
tWartezeit
S
Input.Z2_1

12


Output.Q1_E
R
Input.Z1_1
TON_2
TON
IN
Q
PT
ET
tWartezeit
S
Input.Z1_0
Input.S2

13


Output.Q2_E
R
Input.Z2_1
TON_3
TON
IN
Q
PT
ET
tWartezeit
S
Input.Z2_0

# Literaturverzeichnis

**Böge, Alfred; Böge, Wolfgang (2021)** in: *Handbuch Maschinenbau, Grundlagen und Anwendungen der Maschinenbau-Technik*, 24., überarbeitete und erweiterte Auflage, Springer Vieweg, Wiesbaden

**Ebel, Frank (2000)** in: *Grundlagen der Pneumatik, Foliensammlung TP101*, FESTO Didactic GmbH & CO, Denkendorf

**Heimann, Bodo; Albert, Amos; Ortmaier, Tobias; Rissing, Lutz (2016)** in: *Mechatronik, Komponenten - Methoden - Beispiele*, 4., überarbeitete und ergänzte Auflage, Carl Hanser Verlag, München

**Hering, Ekbert; Endres, Julian; Gutekunst, Jürgen (2021)** in: *Elektronik für Ingenieure und Naturwissenschaftler*, 8. Auflage, Springer Vieweg, Berlin

**o.V. (2004)** in: *Handbuch für SPS Programmierung mit CoDeSys 2.3*, Version 3.0, 3S-Smart Software Solutions GmbH, Kempten

**o.V. (2021)**, https://www.baumer.com/ch/de/service-support/funktionsweise/funktionsweise-und-technologie-von-magnet--und-zylindersensoren/a/Know-how_Function_Magnet-cylinder-sensors, Zugriff am 06.12.2021

**o.V. (Benutzerhandbuch CoDeSys)** in: *CODESYS Benutzerhandbuch*, https://help.codesys.com/, Zugriff am 12.12.2021

**o.V. (Aktuator)** in: *Duden*, https://www.duden.de/rechtschreibung/Aktuator, Zugriff am 08.12.2021

**o.V. (OSI-Schichtenmodell)** in: *Elektronik Kompendium*, https://www.elektronik-kompendium.de/sites/kom/0301201.htm, Zugriff am 13.12.2021

**Riege, Matthias (Bohreinrichtung)** in: *Thema zur Bearbeitung des Assignment im Modul REG23*, pdf-Datei, Zugriff am 13.12.2021

**Seitz, Matthias (2021)** in: *Speicherprogrammierbare Steuerungen in der Industrie 4.0, Objektorientierter System- und Programmentwurf, Motion Control, Sicherheit, Industrial IoT*, 15., aktualisierte und erweiterte Auflage, Carl Hanser Verlag, München

**Wellenreuther, Günter; Zastrow, Dieter (2005)** in: *Automatisieren mit SPS, Theorie und Praxis*, 3., überarbeitete und ergänzte Auflage, Vieweg, Wiesbaden