

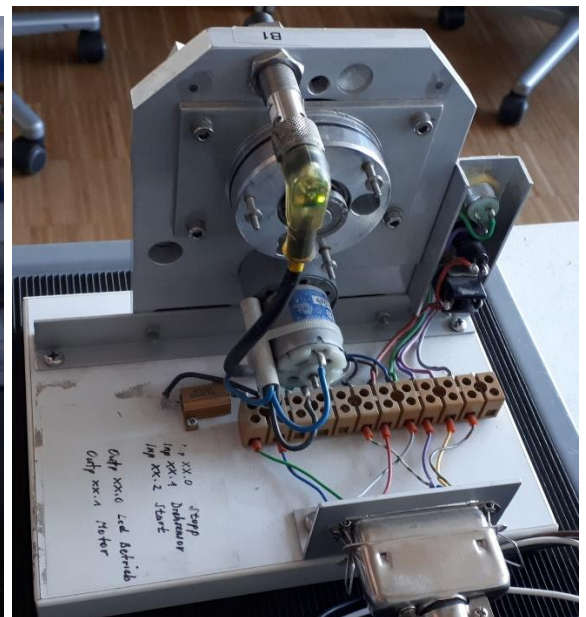
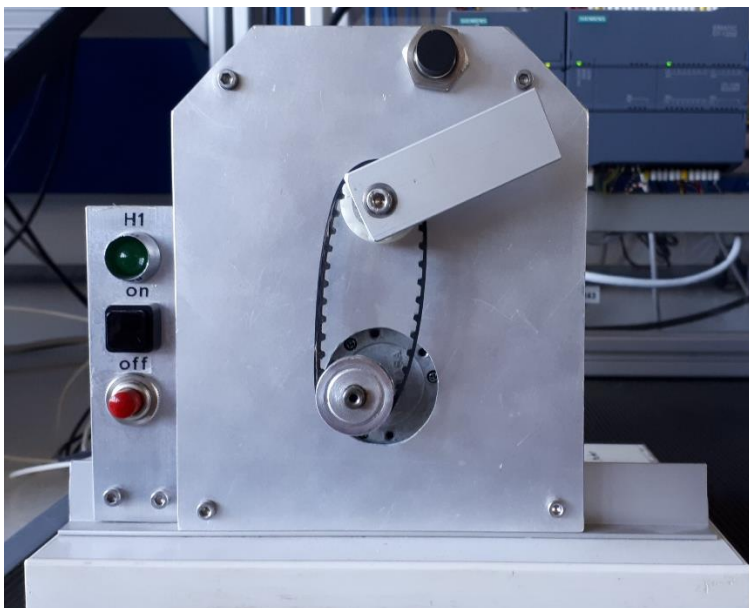
Einführung in die SPS-Programmierung am Beispiel eines Rührwerkmodells

Rührwerkmodell

Das Modell bildet die Funktion eines Rührwerkes nach und dient als Übungsmodell zum Kennenlernen der Simatic S7-1200 Steuerung und dem Programmentwicklungssystem Simatic STEP 7 TIA-Portal.

Komponenten des Rührwerkmodells:

- Antriebsmotor für eine Drehrichtung
- Drehsensor für 1 Puls/Umdrehung
- Bedienung über Taste für Start und Stopp und Anzeigelampe
- Die Ein-, und Ausgänge sind mit einer S7-1200 Steuerung verdrahtet
- Für eine zusätzliche Visualisierung steht ein Simatic KTP700 Basic HMI-Gerät zur Verfügung



Für das Übungsmodell Rührwerk soll schrittweise eine Steuerungssoftware mit folgenden Funktionen erstellt werden:

- Rührwerkmotor ein- und ausschalten über die Tasten am Modell
- Der Betrieb soll mit der LED beim Rührwerksmodell angezeigt werden.
- Über den Sensor am Rührwerk soll die Stopp-Position bei jedem Stoppen gleich sein.
- Drehüberwachung mit Abschaltung, Fehleranzeige, Fehlerquittierung
- Drehzahlmessung
- HMI, KTP600: Parametervorgabe, Istwerte, Bedienung

Aufgabe 1: erstes Programm

Zum ersten Kennenlernen des Simatic S7-1200 Steuerungsgerätes und der Programmierumgebung Simatic STEP 7 TIA-Portal soll ein einfaches Programm erstellt und an der Zielhardware getestet werden.

Arbeitsschritte

Der Platz zwischen den Arbeitsschritten ist für Notizen vorgesehen!

1. TIA starten, Projektansicht
2. Neues Projekt anlegen: Menu Projekt-Neu, oder Icon
Projektname und Speicherpfad angeben
Projektname: *RW <Nachname> JJJJMMTT*
3. Neues Gerät hinzufügen (Projektbaum)
S7-1200 CPU, muss mit der Zielhardware genau übereinstimmen
IP-Adresse der CPU einstellen, 10.100.82.xxx (Gerätekonfiguration)
4. Variablen definieren (PLC-Variablen)
1 Eingang: Taste für Start des Rührwerkmodells
1 Ausgang: Motor des Rührwerkmodells
(Adressen beim Rührwerkmodell herausfinden)
5. Programm erstellen (Programmbausteine), OB1 / Main verwenden.
Funktion: solange die Starttaste gedrückt wird, soll der Motor laufen
6. Projektierung in die CPU laden: Menu Online-Laden in Gerät, oder Icon
7. Programm testen, Programmbausteine beobachten, Variablen beobachten über Beobachtungstabellen
8. Projekt archivieren: Menu Projekt-Archivieren...
Speicherpfad und Dateiname angeben.
Archivname: *RW <Nachname> JJJJMMTT*
> das ganze Projekt wird in einer Datei gespeichert.
(um ein archiviertes Projekt zu bearbeiten, muss es vorher dearchiviert werden)

Aufgabe 2: Bedienfunktionen

Das Simatic Projekt von Aufgabe 1 soll erweitert werden, um folgende Bedienfunktionen zu erhalten:

- Durch kurzes Drücken auf die Starttaste soll das Rührwerk laufen.
- Durch kurzes Drücken auf die Stopptaste soll das Rührwerk bis zum Drehsensor weiterlaufen und dann stoppen.
- Solange das Rührwerk läuft, soll die Betriebsanzeige leuchten.

Wie soll die Anlage reagieren, wenn beide Tasten gleichzeitig gedrückt werden?

- Es soll eine sinnvolle Lösung gefunden, begründet und umgesetzt werden.

Informationen

Drahtbruchssichere Einbindung von Sensoren

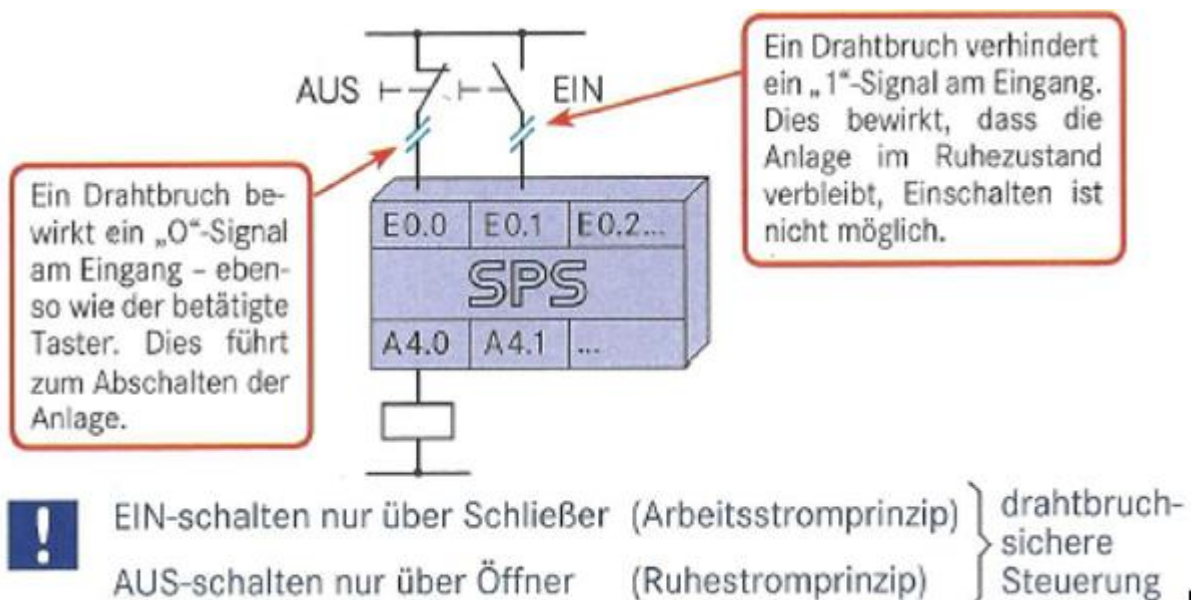
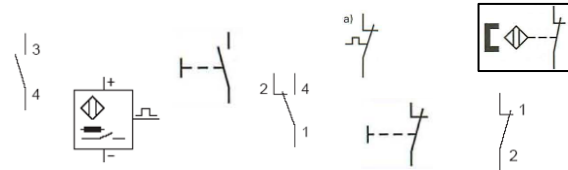
- Bei sicherheitsrelevanten Signalen erforderlich
- Drahtbruch bewirkt Gleiches wie die Sensorbetätigung.
- Realisierung durch Sensoren mit Öffner.

Schließer, NO

nicht betätigt ----- offen
betätigt ----- geschlossen

Öffner, NC


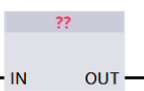
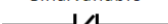
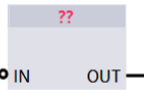
nicht betätigt -geschlossen
betätigt ----- offen



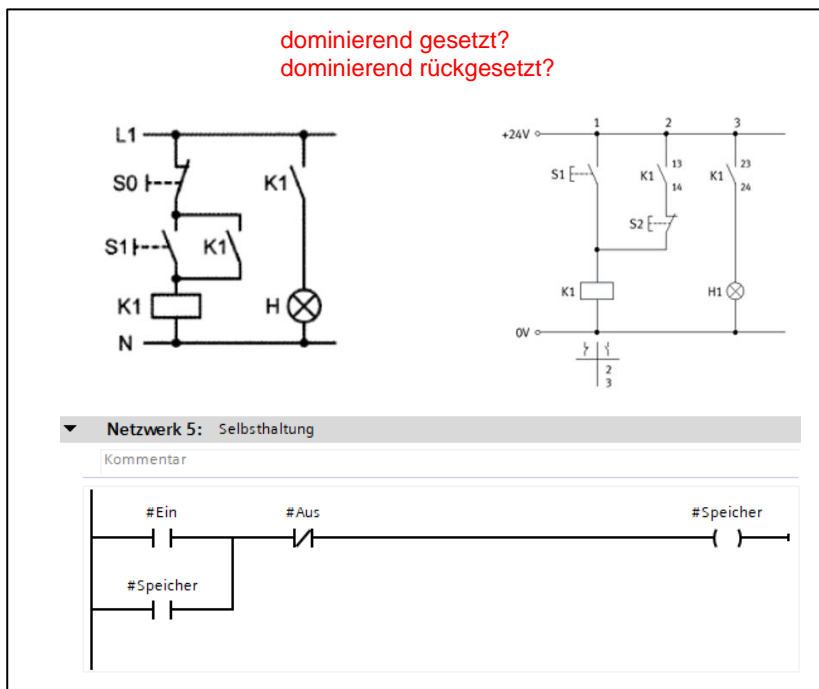
Beschreibung in der Variablenliste:

- Die Schaltlogik des Sensors muss in der Variablenliste eindeutig sein.
- Üblicherweise erfolgt die Beschreibung für '1'- Signal
z.B.: Taste Stopp ist nicht gedrückt, od. Taste Stopp (NC)

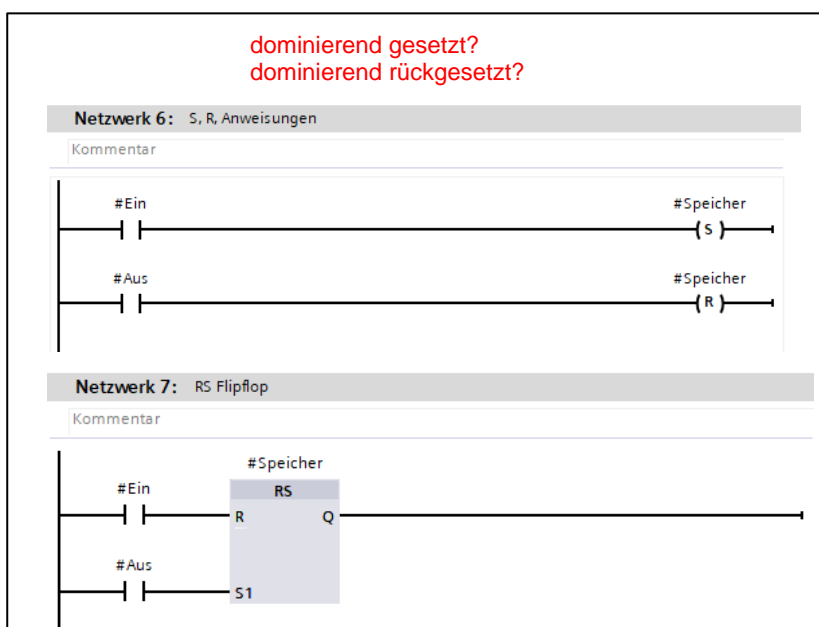
Signalabfragelogik im Programm

Signalzustand abfragen	KOP	FUP												
Abfrage auf „1“	<p>%E0.0 "Binärvariable"</p> 	 <p>%E0.0 "Binärvariable"</p>												
Abfrage auf „0“	<p>%E0.0 "Binärvariable"</p> 	 <p>%E0.0 "Binärvariable"</p>												
<table border="1"> <thead> <tr> <th></th><th colspan="2">Signalzustand</th></tr> <tr> <th>Binärvariable</th><th>0</th><th>1</th></tr> </thead> <tbody> <tr> <td>Abfrageergebnis Abfrage auf „1“</td><td>0</td><td>1</td></tr> <tr> <td>Abfrageergebnis Abfrage auf „0“</td><td>1</td><td>0</td></tr> </tbody> </table>				Signalzustand		Binärvariable	0	1	Abfrageergebnis Abfrage auf „1“	0	1	Abfrageergebnis Abfrage auf „0“	1	0
	Signalzustand													
Binärvariable	0	1												
Abfrageergebnis Abfrage auf „1“	0	1												
Abfrageergebnis Abfrage auf „0“	1	0												

Möglichkeiten zum Programmieren von speicherndem Verhalten



Selbsthaltung



S, R Anweisungen
Flipflop Anweisung

Aufgabe 3: Drehüberwachung

Das Simatic Projekt von Aufgabe 2 soll um eine Drehüberwachung erweitert werden:

- Wenn bei eingeschaltetem Rührwerk der Drehsensor längere Zeit nicht anspricht, soll die Anlage wie folgt reagieren:
 - Der Motor wird abgeschaltet
 - Eine Störungsinformation wird angezeigt
- Um das Rührwerk wieder starten zu können, muss vorher die Störung quitiert werden.
- Der Störungsspeicher soll remanent ausgeführt werden. Damit ist gemeint, dass der Störungsspeicher auch erhalten bleibt, wenn die Stromversorgung der Steuerung unterbrochen wird.

Informationen

Zeitfunktionen

Parameter	Deklaration	Datentyp	Beschreibung
IN	Input	BOOL	Starteingang
PT	Input	TIME	Zeitdauer der Einschaltverzögerung. Der Wert am Parameter PT muss positiv sein.
Q	Output	BOOL	Ausgang, der nach dem Ablauf der Zeit PT gesetzt wird.
ET	Output	TIME	Aktueller Zeitwert

Parameter	Operand	Wert
IN	Tag_Start	Signalwechsel "0" => "1"
PT	Tag_PresetTime	T#10s
Q	Tag_Status	FALSE; nach 10s => TRUE
ET	Tag_ElapsedTime	von T#0s => T#10s

Zeiten

- TP: Impuls erzeugen
- TON: Einschaltverzögerung erzeugen
- TOF: Ausschaltverzögerung erzeugen
- TONR: Zeit akkumulieren

Taktsignal, Möglichkeiten

Netzwerk 4: Blinktakt

Gerätedaten

Eigenschaften

Systemkonstanten

Immer 0 (low):

Taktmerkerbits

☒ Verwendung des Taktmerkerbytes aktivieren

Adresse des Taktmerkerbytes (MBx): 1000

Wertebereich: [0..8191]

Takt 10 Hz: %M1000.1 (Clock_5Hz)

Takt 5 Hz: %M1000.2 (Clock_2.5Hz)

Takt 2.5 Hz: %M1000.3 (Clock_2Hz)

Takt 1.25 Hz: %M1000.4 (Clock_1.25Hz)

Takt 1 Hz: %M1000.5 (Clock_1Hz)

Anlauf

Zyklus

Kommunikationslast

System- und Taktmerker

Webserver

Konfigurationssteuerung

Einstellung des remanenten Speicherverhalten für den Merkerbereich

The screenshot shows the Siemens STEP 7 interface. On the left is the 'Geräte' (Devices) tree with 'PLC_1 [CPU 1215C AC/DC/Rly]' expanded. The main window displays the 'PLC-Variablen' (PLC Variables) table. A red arrow points to the 'Remanenter Speicher' (Remanent Memory) icon in the toolbar. The 'Remanenter Speicher' dialog box is open, showing settings for remanent memory. The 'Anzahl der remanenten Bytes, beginnend bei MBO:' is set to 10. The 'Remanente SIMATIC-Zeiten, beginnend bei T0:' and 'Remanente SIMATIC-Zähler, beginnend bei Z0:' are both set to 0. The 'Aktuell verfügbarer remanenter Speicher (Bytes):' is 10230. The background table lists variables with their names, data types, and addresses.

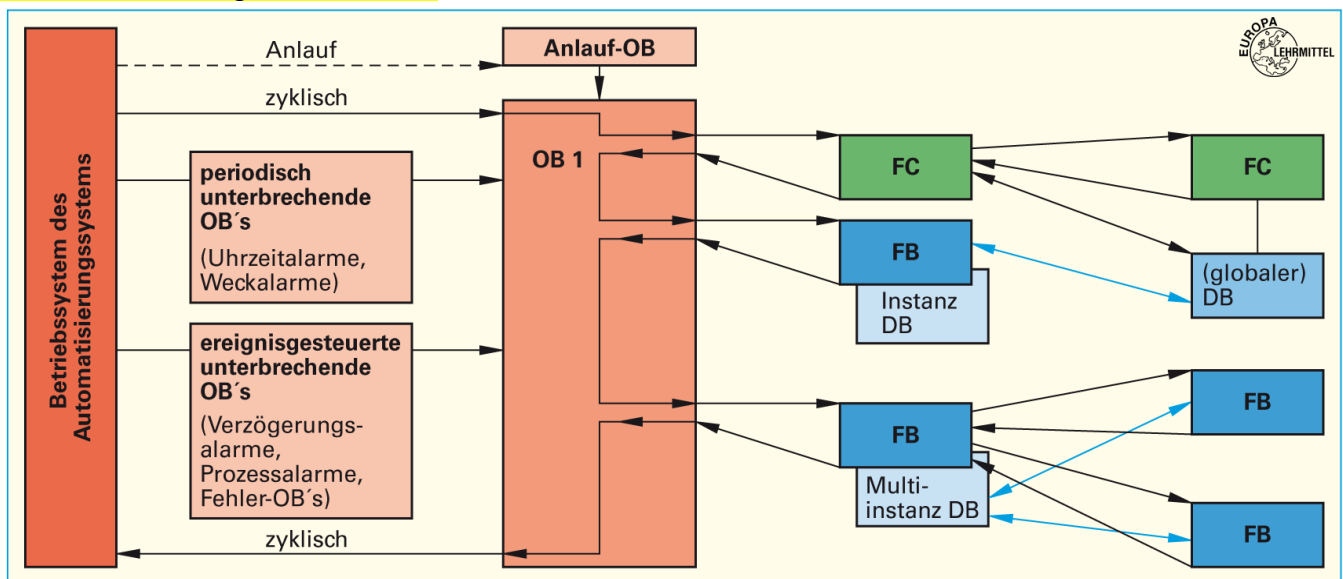
Name	Variablentabelle	Datentyp	Adresse	Rema...
AnlageInGrundstellung...	Standard-Variablen...	Bool	%M0.6	<input type="checkbox"/>
AnlageInGrundstellung...	Standard-Variablen...	Bool	%M0.7	<input type="checkbox"/>
AnlageInGrundstellung...	Standard-Variab...	Bool	%M9.4	<input checked="" type="checkbox"/>
AnlageInGrundstellung...	Standard-Variablen...	Bool	%M10.5	<input type="checkbox"/>

Aufgabe 4: Funktionsbaustein

Das Simatic Projekt von Aufgabe 3 soll derart umgeändert werden, dass die ganze Rührwerkfunktion in einem Funktionsbaustein (FB) programmiert ist. Der FB soll so erstellt werden, dass er im Programm mehrmals aufgerufen werden kann, wenn beispielsweise bei einer Anlage mehrere Rührwerke gesteuert werden sollen.

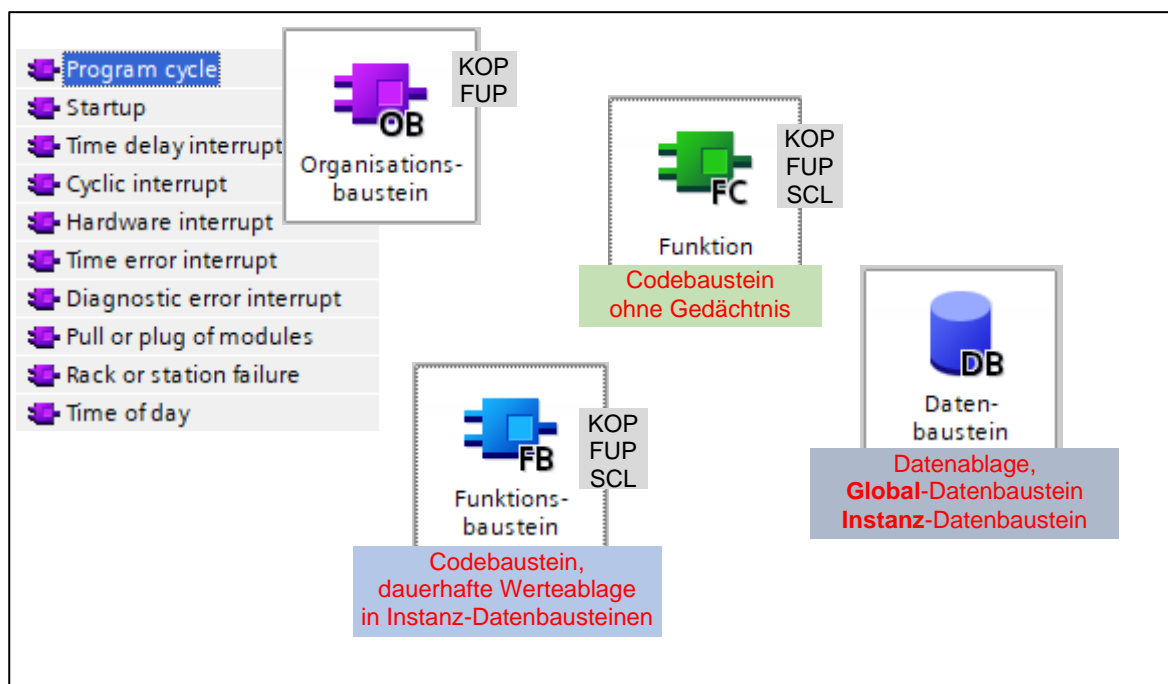
Informationen

Strukturierte Programmstruktur



Quelle: Fachkunde Elektrotechnik, 2017, S. 550

Bausteintypen bei Simatic

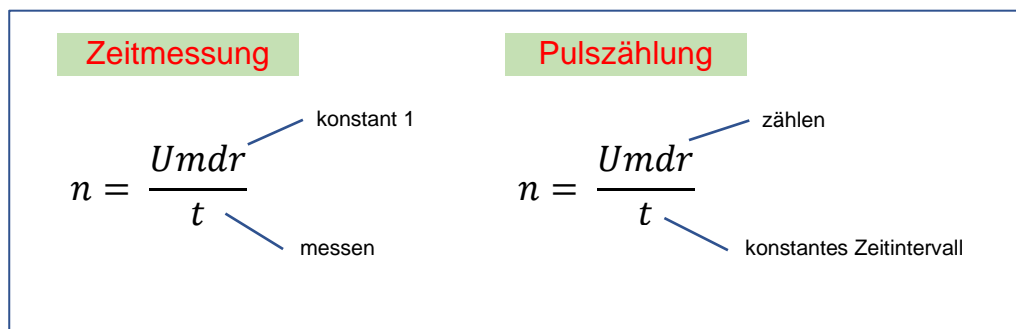


Aufgabe 5: Drehzahlmessung

Das Simatic-Projekt von Aufgabe 4 soll eine Drehzahlmessung bekommen. Der aktuelle Drehzahlwert soll mit der Einheit von 1/100 U/min in einer INT Variable gespeichert werden. Wenn das Rührwerk nicht eingeschaltet ist, soll auch der Drehzahlwert 0 sein. Die Drehzahlmessung soll in einem eigenen FB programmiert werden und vom Rührwerk-FB aufgerufen werden.

Informationen

Prinzipielle Möglichkeiten der Drehzahlmessung über Impulse



Datentypen für Wertedarstellung

INT	(16 bit)	Ganzzahl mit Vorzeichen, 16 Bit	Dezimal (Zweierkompl.)	-32_768 bis +32_767 INT#x, INT#10#x
DINT	(32 bit)	Ganzzahl mit Vorzeichen, 32 Bit	Dezimal (Zweierkompl.)	-2_147_483_648 bis +2_147_483_647 DINT#x, DINT#10#x, L#x
REAL	(32 bit)	Gleitpunktzahl	Exponentialdarstellung	±3.402823E+38 bis ±1.175494E-38 REAL#x
			Dezimaldarstellung	123.4567 REAL#x

Mathematische Funktionen

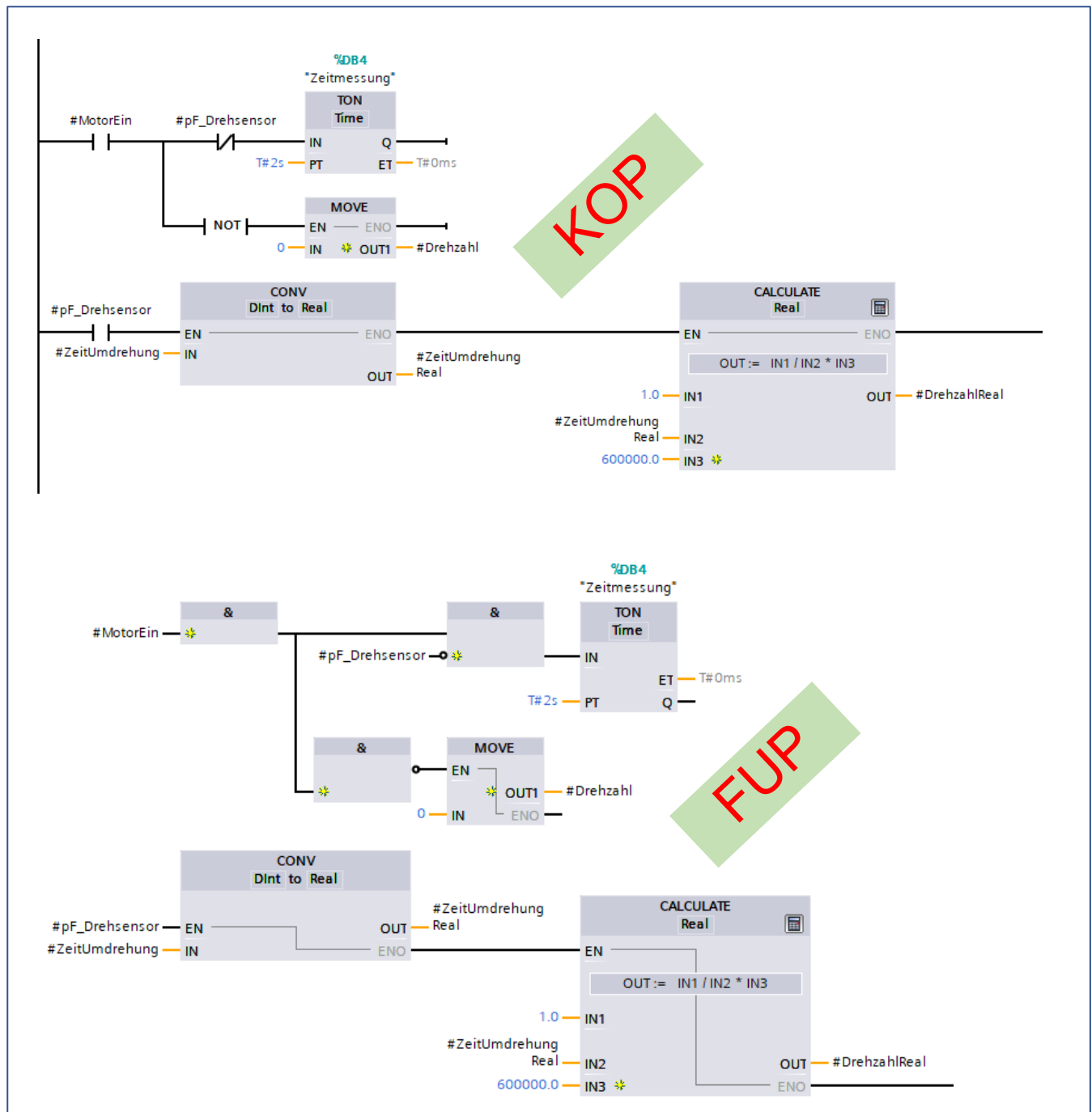
Mathematische Funktionen	
CALCULATE	Berechnen
ADD	Addieren
SUB	Subtrahieren
MUL	Multiplizieren
DIV	Dividieren

Aufgabe 6, Programmiersprache SCL

Die Drehzahlmessung von Aufgabe 5 soll zusätzlich auch in der Programmiersprache SCL programmiert werden. Dafür soll ein eigener FB erstellt werden.

Informationen

Programmiersprachen KOP, FUP



Programmiersprache SCL

IF...	CASE... OF...	FOR... TO DO...	WHILE... DO...	(*...*)	REGION
1 IF "StoppNicht" = TRUE AND "Start" = true THEN					"StoppNicht" %E0.0 Öffner (NC)
2 // Statement section IF					"Start" %E0.2
3 "Motor Ein" := true ;					"Motor Ein" %A0.1
4 END_IF;					
5 IF "StoppNicht" = FALSE AND "Motor Ein" = true THEN					"StoppNicht" %E0.0 Öffner (NC)
6 // Statement section IF					"Motor Ein" %A0.1
7 #AnwahlStopp := true;					
8 END_IF;					
9 IF #AnwahlStopp = TRUE AND "Drehsensor" = true THEN					"Drehsensor" %E0.1 1 Puls pro Umdrehung
10 // Statement section IF					
11 "Motor Ein" := FALSE;					"Motor Ein" %A0.1
12 END_IF;					
13 IF "Motor Ein" = FALSE THEN					"Motor Ein" %A0.1
14 // Statement section IF					
15 #AnwahlStopp := FALSE;					
16 END_IF;					
17 "Lampe Betrieb" := "Motor Ein";					"Lampe Be..." %A0.0
18					"Motor Ein" %A0.1

▼ Programmsteuerung	
SCL IF ... THEN ...	Bedingt ausführen
SCL IF ... THEN ... ELSE ...	Bedingt verzweigen
SCL IF ... THEN ... ELSIF ...	Mehrfach bedingt verzweigen
SCL CASE ... OF ...	Mehrfach verzweigen
SCL FOR ... TO ... DO ...	In Zählschleife ausführen
SCL FOR ... TO ... BY ... DO ...	In Zählschleife mit Schrittweite ausfü...
SCL WHILE ... DO ...	Bei erfüllter Bedingung ausführen
SCL REPEAT ... UNTIL ...	Bei nicht erfüllter Bedingung ausfüh...
SCL CONTINUE	Schleifenbedingung erneut prüfen
SCL EXIT	Schleife sofort verlassen
SCL GOTO ...	Springen
SCL RETURN	Baustein verlassen
SCL (* *)	Kommentarabschnitt einfügen
SCL REGION	Programmcode strukturieren

Die Programmiersprache SCL (Hochsprache) wird üblicherweise für komplexe Berechnungen, Datenverarbeitung, Stringverarbeitung, usw. verwendet.