Vorwort

In diesem Leitfaden wird die Programmierung einer Zweipunktregelung bzw. einer PID-Regelung mit der Micro-Control-Demostation (Siemens S7-1200 SPS) unter der Siemens Software STEP 7 Professional TIA V11.0 SP1 beschrieben.

Inbetriebnahme der Anlage:

Um die Anlage in Betrieb nehmen zu können, wird zuerst der Würfel mit dem Montageboard verbunden. Dafür wird je ein Bananenstecker in eine Buchse des Würfels gesteckt, dass dazugehörige Ende wird entsprechend der Buchsennummer in diejenige Klemme der Klemmenleiste X4 gesteckt. Es ist <u>unbedingt</u> auf die richtige Reihenfolge zu achten, ansonsten riskiert man Sachschäden an der Anlage! Danach wird der Stecker in eine Steckdose gesteckt, alle Sicherungen werden eingeschaltet und das Netzwerkkabel wird mit dem Profinetanschluss der S7-1200 verbunden. Bei korrektem Anschluss startet die SPS im RUN-Betrieb und der Temperaturmessumformer "B2" gibt keinen Fehler aus (sonst leuchtet eine rote Lampe unterhalb der weißen Plastikabdeckung). Als letztes muss man kontrollieren, ob die beiden Striche des Potentiometers auf der Platine deckungsgleich sind.

Im folgenden wird davon ausgegangen, dass ein fertig konfiguriertes Projekt nach dem Leitfaden "STEP 7 Basic V10.5" für die CPU 1214C DC/DC/DC V2.0 erstellt wurde, deshalb wird hier auf die Gerätekonfiguration, die Einbindung der SPS ins Netzwerk und die grundlegende Handhabung der Software nicht weiter eingegangen.

Vorbereitung

Schritt 1:

Als erstes fügen wir die analoge Ausgangskarte der Gerätekonfiguration hinzu. Dazu wechseln wir in die *Gerätekonfiguration* und öffnen rechts am Rand den Reiter Hardwarekatalog. Wir öffnen das Untermenü *"Signalboards -> AQ -> AQ1 x 12 Bit"* und ziehen das Signalboard per *"*Drag and Drop" in die Mitte der CPU. Nun ist die Ausgabebaugruppe korrekt eingebunden.



Schritt 2:

Als nächstes ändern wir die Ein- und Ausgabeadressen in der *Geräteübersicht* der CPU wie im Bild dargestellt (vgl. Schritt 7 im Leitfaden zu STEP 7).

祝 Baugruppe	Steck	E-Adresse	A-Adresse	Тур	Bestell-Nr.	Firmware
	103					
	102					
	101					
✓ A0	1			CPU 1214C DC/DC/DC	6ES7 214-1AE30-0XB0	V2.0
DI14/DQ10_1	11	01	01	D114/DQ10		
AI2_1	12	25		AI2		
AQ1 x12Bit_1	13		23	AQ1 Signalboard	6ES7 232-4HA30-0XB0	V1.0
HSC_1	1 16	100010		Schneller Zähler (HSC)		
HSC_2	1 17	100410		Schneller Zähler (HSC)		
HSC_3	1 18	100810		Schneller Zähler (HSC)		
HSC_4	1 19	101210		Schneller Zähler (HSC)		
HSC_5	1 20	101610		Schneller Zähler (HSC)		
HSC_6	1 21	102010		Schneller Zähler (HSC)		
Pulse_1	1 32		100010	Impulsgenerator (PTO/P		
Pulse_2	1 33		100210	Impulsgenerator (PTO/P		
PROFINET-Schnittstelle 1	1 X1			PROFINET-Schnittstelle		

Die restlichen Einstellungen belassen wir bei den Standardwerten. Es ist sinnvoll den Netzwerken Namen zu geben und beschreibende Kommentare zu benutzen.

Zweipunktregelung

Schritt 1:

Wir öffnen in der Projektnavigation unter der CPU den Reiter "*PLC-Variablen -> Alle Variablen anzeigen"* und legen folgende Variablen dem Bild entsprechend fest.

PL	C-Variablen						
	Name	Datentyp	Adresse	Rema- nenz	Sicht- bar in HMI	Er- reich- bar aus HMI	Kommentar
-	AQ	Word	%QW2	False	True	True	Ansteuerung Transistor
	AIO	Word	%IW2	False	True	True	Signal vom PT-100-Tem- peratursensor
-0	Lampe EIN	Bool	%10.0	False	True	True	Schließer Lampe EIN
-	Lampe AUS	Bool	%IO.1	False	True	True	Öffner Lampe AUS
-	к1	Bool	%Q0.0	False	True	True	Relais K1 Lampe EIN/AUS
-	Sollwert Eingabe	Real	%QD4	False	True	True	Eingabe des Sollwertes in Grad Celsius
	Umrechnungsfaktor	Real	%QD8	False	True	True	Umrechnungsfaktor für die Solltemperatur in Volt
	Sollwert Volt	Real	%QD12	False	True	True	Zwischenergebnis Sollwert in Volt
-	Zwischenergebnis 1	Real	%QD16	False	True	True	Zwischenergebnis Berech- nung Sollwert als INT-Zahl
-	Sollwert Real	Real	%QD20	False	True	True	Sollwert als REAL-Zahl
-0	Sollwert Int	Int	%QW24	False	True	True	Sollwert als INT-Zahl
	Merker Ausgang	Bool	%M1.0	False	True	True	Analoger Ausgang EIN/AUS

Schritt 2:

Wir öffnen in der Projektnavigation unter der CPU den Reiter "*Programmbausteine -> Neuen Baustein hinzufügen"*. Wir wählen *Funktion* aus, vergeben einen Namen, ändern die Sprache auf FUP und bestätigen mit OK.



Wir öffnen nun den Baustein "Main [OB1]" indem wir auf diesen Doppelklicken. Folgend ziehen wir den Baustein FC_1 per "Drag and Drop" in das Netzwerk 1 des OB1. Das Ergebnis sieht folgendermaßen aus:

-		
^		
	 Bausteintitel: "Main Program Sweep (Cycle)" 	
	Kommentar	
	Netzwerk 1:	
=	Kommentar	
	%FC1	
	"FC_1"	
	– EN ENO –	
		► HF H/F -O- 121 → -1 ■ Bausteintitel: *Main Program Sweep (Cycle)* Kommentar ■ Netzwerk 1: Kommentar FC1 FC_1* EN ENO

Als Effekt wird nun auch der FC_1 im Betrieb der SPS aufgerufen.

Schritt 3:

Nun programmieren wir das Ein- und Ausschalten der Lampe. Dazu wechseln wir in die Ansicht des FC_1 durch Doppelklick und erstellen im Netzwerk 1 eine S/R-Stufe für die Ansteuerung der Lampe (Doppelklick oder Drag and Drop unter *"Einfache Anweisungen -> Bitverknüpfungen -> SR"*). Gesetzt wird die S/R-Stufe durch den Schließer S1 (10.0), rückgesetzt durch den Öffner S0 (10.1). Die S/R-Stufe steuert das Relais K1 an, das die Lampe schaltet.

Projektnavigation	Î (nktregelung + A0 [CPU 1214C DC/DC/DC] + Programmbausteine + FC_1 [FC1] 😐 🖬 🚍 🗙	Anweisungen	🗗 🔟 🕨
Geräte			Optionen	
000		ು ನ ೫ ೫ ₦ 🗖 🖬 🖓 📲 🖓 🥙 🚱 🐂 🖓 🖼		
		Bausteinschnittstelle	> Favoriten	
▼ 🔄 Zweipunktregelung	^		✓ Einfache Anweisung	jen ,
Neues Gerät hinzufügen			Name	1
Geräte & Netze		Bausteintitel: FC_1	🕨 🛅 Allgemein	~
- A0 [CPU 1214C DC/DC/D	c]	Kommentar	🕶 🔄 Bitverknüpfung	
Gerätekonfiguration			8 🗉	(
🧏 Online & Diagnose		Netzwerk 1: Relais K1	E >=1	
👻 🙀 Programmbausteine		🗢 Der Schließer S1 setzt das S/R-Kippglied, der Öffner S0 lässt es abfallen. Die Lampe E1 wird über	E x	
Neuen Baustein hi	inzufügen	das Relais K1 ein- und ausgeschaltet.	E -[=]	
Main [OB1]			E -(/=)	
EC_1 [FC1]		%Q0.0		
🕨 🎆 Technologieobjekte		- <u>k1</u> -	E -[s]	
🕨 🐻 Externe Quellen		SR	E SET BF	1
🗢 🌄 PLC-Variablen		0.048	RESET BF	
lle Variablen anz	eigen	"Lampe EIN" — S		
Neue Variablentat	belle hinzu	%0.1	FT RS	
💥 Standard Variable	ntabelle ["Lampe AUS" — R1 Q —		
PLC-Datentypen				
📃 🕨 🥅 Beobachtungs- und F	orcetabel 🗡		FT -iP1-	
<	>	Netzwerk 2:		

Schritt 4:

4

Nun erzeugen wir, wie vorhin, einen FC_2 und binden auch diesen in den OB1 ein.

In dem folgenden Abschnitt wird behandelt, wie eine Solltemperatur eingegeben und mit einer Isttemperatur verglichen werden kann. Dementsprechend wird ein Ausgang, der den Lüftermotor steuert, ein- oder ausgeschaltet. Dafür begeben wir uns in die Theorie der Analogwertverarbeitung der Siemens S7-1200. Wir müssen außerdem noch wissen, dass unser PT-100 Signal von einem Messumformer aufbereitet wird. Dieser bildet einen Temperaturbereich von 0 C° bis 65 C° auf einen Spannungsbereich von 0 V- 10 V ab.

Das bedeutet für uns, dass wir unseren Sollwert, den wir in Grad Celsius eingeben möchten, in eine Spannung übersetzen müssen. Wir überprüfen, dass folgende Umrechnungsformel dafür in unserem Fall zum Ziel führt: $Sollwert [V] = \frac{10 [V]}{65 [C^{\circ}]} * Sollwert [C^{\circ}]$. Der <u>rot</u> markierte Teil ist unser Umrechnungsfaktor. Wir programmieren also zunächst unseren Umrechnungsfaktor durch Division von 10 durch 65 und multiplizieren das Ergebnis mit unserem Sollwert in C°. Dafür ziehen wir per Drag and Drop aus dem rechten Reiterzusammenschluss *"Anweisungen -> Einfache Anweisungen -> Mathematische Funktionen"* einmal *DIV* und einmal *MUL* in das erste Netzwerk des FC_2. Als Datentyp wählen wir beiden *"*Real" aus. Wir geben die Werte entsprechend der obigen Formel ein, dass Ergebnis sieht folgendermaßen aus:



Unseren Sollwert in C° geben wir also beim Eingang IN2 des Multiplikationsbausteins ein.

Schritt 5:

Die S7-1200 bildet nun generell eine anliegende Spannung an einer analogen Eingangskarte auf eine Integerzahl ab. Diese Abbildung erfolgt von 0 V – 10 V auf einen Integerbereich von 0 bis 27648. Deshalb bilden wir jetzt nach dieser Vorgabe unseren Sollwert in Volt auf seine zugehörige Integerzahl ab, um nachher den Sollwert mit der Isttemperatur vergleichen zu können (die Isttemperatur liegt, wie gesagt, als Integerzahl am analogen Eingangs vor). Wir führen folgende Rechnung durch und programmieren diese wie oben:

$$Sollwert [Integerzahl] = \frac{Sollwert [V]}{10 [V]} * 27648$$

Das Ergebnis sieht folgendermaßen aus:



Schritt 6:

Bis jetzt wurde nur in Real-Zahlen gerechnet, um keine Rundungsfehler zu erhalten. Da unsere Isttemperatur nachher aber als Integerzahl am Eingang vorliegt, müssen wir unseren Sollwert nun noch in eine Integerzahl umwandeln (runden). Wir wählen unter dem Reiter *"Einfache Anweisungen -> Umwandler"* den Befehl *Convert* aus und ziehen diesen per Drag and Drop in das Netzwerk 2 des FC_2. Wir wandeln von Real nach Int und definieren den Ein- bzw. Ausgang noch mit unseren richtigen Variablen.



Jetzt liegt unser eingegebener Sollwert als verwertbare Integerzahl vor.

Schritt 7:

Wir können nun mit dem Vergleichen des Sollwertes und des Istwertes fortfahren. Dafür erstellen wir eine S/R-Stufe. Am Setzeingang fügen wir ein größer als Vergleicher (*CMP* >) ein, am Rücksetzeingang einen kleiner gleich (*CMP* <=) Vergleicher. Beide sind im Reiter *"Einfache Anweisungen -> Vergleicher"* zu finden. Wir vergleichen jeweils die Solltemperatur als Integerzahl mit dem analogen Eingang, an dem unser Messsignal vorliegt.

Als Ergebnis erhalten wir eine S/R-Stufe, die gesetzt ist, falls sich der Istwert oberhalb des Sollwertes bewegt.



Schritt 8:

Nun müssen wir es schaffen, dass bei gesetzter S/R-Stufe "Merker Ausgang" aus dem Schritt 7 eine Spannung von 8,5 V an unserem analogen Ausgang anliegt. Sonst soll eine Spannung von 0V ausgegeben werden. Dies hat schaltungstechnische Gründe, die am verwendeten Transistor und seiner Ansteuerung liegen. Wir nehmen diese Tatsache hier als gegeben hin.

Dafür verwenden wir nun aus dem Reiter *"Einfache Anweisungen -> Wortverknüpfungen"* den Befehl *SEL*. Dieser ermöglicht es uns, einen boolschen Eingang auszuwerten und je nach Zustand dieses Eingangs entweder den Eingang INO oder IN1 an einem Ausgang auszugeben. Wir können also jetzt unseren *"Merker Ausgang"* am Eingang G auswerten, der Typ unserer beiden Eingänge INO und IN1 ist INT, der SEL-Befehl bekommt dementsprechend diesen Typ. Wir möchten jetzt bei nichtgesetzter S/R-Stufe 0 V ausgeben, d.h. bei unserem Eingang INO schreiben wir die 0, bei unserem Eingang IN1 schreiben wir die Zahl 23501, da nach unserer Umrechnungsbeziehung von oben gilt 23500,8 = $\frac{8.5 V}{10 V} * 27648$ (aufgerundet). Unser Ausgang ist hier logischerweise der analoge Ausgang. Wir erhalten folgendes Ergebnis:



Wir sind fertig mit programmieren und können nach dem Leitfaden "STEP 7 Basic V10.5" alles auf die S7-1200 übertragen und mit der "Brillenfunktion" das arbeitende Programm betrachten. Wir haben eine Zweipunktregelung programmiert.

PID-Regelung:

Schritt 1:

Wir öffnen in der Projektnavigation unter der CPU den Reiter "*PLC-Variablen -> Alle Variablen anzeigen"* und legen folgende Variablen dem Bild entsprechend fest.

PLC-Variablen						
Name	Datentyp	Adresse	Rema- nenz	Sicht- bar in HMI	Er- reich- bar aus HMI	Kommentar
-•■ ^{K1}	Bool	%Q0.0	False	True	True	Relais K1 zur Ansteuerung der Lampe E1
ampe AUS	Bool	%IO.1	False	True	True	Öffner SO Lampe AUS
√ ■ ^{AQ}	Word	%QW2	False	True	True	Analoger Ausgang zur An- steuerung des Transistors T2
AIO AIO	Word	%IW2	False	True	True	Analoger Eingang zur Er- fassung der Temperatur über den Messumwandler
Sollwert PID	Real	%ID4	False	True	True	Sollwert für die PID-Rege- lung

Schritt 2:

Wir öffnen in der Projektnavigation unter der CPU den Reiter *"Programmbausteine -> Neuen Baustein hinzufügen"*. Wir wählen *Funktion* aus, vergeben einen Namen, ändern die Sprache auf FUP und bestätigen mit OK.



Wir öffnen nun den Baustein "Main [OB1]" indem wir auf diesen Doppelklicken. Folgend ziehen wir den Baustein FC_1 per "Drag and Drop" in das Netzwerk 1 des OB1. Das Ergebnis sieht folgendermaßen aus:



Als Effekt wird nun auch der FC_1 im Betrieb der SPS aufgerufen.

Schritt 3:

Nun programmieren wir das Ein- und Ausschalten der Lampe. Dazu wechseln wir in die Ansicht des FC_1 durch Doppelklick und erstellen im Netzwerk 1 eine S/R-Stufe für die Ansteuerung der Lampe (Doppelklick oder Drag and Drop unter *"Einfache Anweisungen -> Bitverknüpfungen -> SR"*). Gesetzt wird die S/R-Stufe durch den Schließer S1 (I0.0), rückgesetzt durch den Öffner S0 (I0.1). Die S/R-Stufe steuert das Relais K1 an, das die Lampe schaltet.



Schritt 4:

Wir öffnen in der Projektnavigation unter der CPU den Reiter "*Programmbausteine -> Neuen Baustein hinzufügen"*. Wir wählen "*Organisationsbaustein -> Cyclic interrupt" (Weckalarm-OB)* aus, vergeben einen Namen, ändern die Sprache auf FUP, setzten die Zykluszeit auf 250 ms und bestätigen mit OK. Dies ermöglicht dem OB später eine gewisse Zeit, um die Rechnung ausführen zu können. Ansonsten könnte es passieren, dass der PID-Regler noch rechnet, zu keinem Ergebnis gekommen ist, die CPU aber schon wieder in ihrer Zykluszeit von vorne anfängt. Dieses Phänomen müssen wir hierdurch verhindern.



Schritt 5:

Wir öffnen in der Projektnavigation unter der CPU den Reiter *"Technologieobjekte -> Neuen Baustein hinzufügen"*. Wir wählen *"PID-Control -> PID_Compact"* aus und belassen den Namen.



von Nils-Erik Schäfer

Schritt 6:

Wir schließen das sich öffnende Fenster und öffnen den Baustein PID-Regelung [OB30]. Wir navigieren dann in der Projektnavigation unter der CPU zu *"Programmbausteine -> Systembausteine -> Programmressourcen"* zu PID_Compact [FB1130] und ziehen diesen per Drag and Drop in das Netzwerk 1 des OB30. Wir wählen, wie unten dargestellt, in dem auftretenden Fenster als Name des Datenbausteins den vorher erstellten PID-Compact_1 aus und bestätigen mit OK.

ne	🚱 Online & Diagnose	^	Aufrufoptionen	-1.		×
PLC- Programm	 Programmbausteine Neuen Baustein hinzufügen Main [OB1] PID-Regelung [OB30] FC 1 [FC1] Systembausteine Programmressourcen PID_Compact [FB1130] Reues Objekt hinzufügen Neues Objekt hinzufügen PID_Compact_1 [DB1] Konfiguration Inbetriebnahme Externe Quellen 		Einzel- Instanz	Datenbau Name Nummer Der aufger einem eige mehr	Ustein PID_Compact_1 Manuell Manuell Automatisch ufene Funktionsbaustein speichert se enen InstanzDatenbaustein.	eine Daten in
	PLC-Variablen Alle Variablen anzeigen				ОК	Abbrechen

Danach klicken wir mit einem Rechtsklick auf den Baustein im Netzwerk 1 des OB30 und wählen Eigenschaften aus.



Schritt 7:

In dem sich öffnenden Fenster konfigurieren wir die *Grundeinstellungen* wie gezeigt. Wir wählen als *Regelungsart Temperatur* in *Grad Celsius* aus und invertieren den Regelsinn, da bei uns eine höhere Kühlleistung eine niedrigere Temperatur verursacht. Unseren Sollwert (*"Setpoint"*) setzen wir auf 37 C°, den Eingang (*"Input"*) auf Input_PER (analog) mit unserem analogen Eingang AlO und den Ausgang (*"Output"*) auf Output_PER (analog) mit unserem analogen Ausgang AQ.

PID_Compact			Eigenschafter	n 🔄 🗓 Info 🔒 🗓 Diagno
Allgemein	nfiguration			
Grundeinstell 🤣 Istwerteinstel 🥑	Grundeinstellungen _			
	Regelungsart			
	Temperatur	∞ ≈	💌 🛃 Invertieren des	Regelsinns
	Eingangs-/Ausgan	rt letzte Betriebsar Igsparameter	: aktivieren	
	37.0	l lec		
	Input:			Output:
-	Input_PER (analog)	•		Output_PER (analog)
•	■ ★ AIO*			

Wir klicken links auf *Istwerteinstellungen* und konfigurieren auch hier wie angezeigt. Der *obere Istwert* ist hier beides mal 65 C°, der *untere Istwert* 0 C°, weil der Messumformer diese Temperaturspanne auf 0 V bis 10 V abbildet.

PID_Compact			Q	Eigenschafte	n 🛄 Info	😧 🗓 Diagnose
Allgemein	Konfiguration					
Grundeinstell Istwerteinstel	Istwerteinstel	lungen				
		Input_PER:	Aktiviert			
	ŝ	kalierter oberer Istwert:	65.0	 ~		
	-	Obergrenze Istwert:	65.0	~		
		Untergrenze Istwert:	0.0	<u> </u>		
	SI	alierter unterer Istwert:	0.0	<u> </u>		
						•
					0	27648
	•		Input_P	PER:	Unten:	Oben:
			Auto	matische Einste	ellung	

!Achtung!

Wer jetzt nochmal auf Grundeinstellungen zurückklickt, wird merken, dass der In- und Output eventuell wieder auf dem Standardwert steht. Dies ist leider so, weil das Programm eine Macke hat. Wir werden es jetzt "zwingen" unseren richtigen Ein- und Ausgang zu verwenden. Dazu schließen wir das Fenster und öffnen nun nochmal den OB30 und sehen den Baustein PID_Compact_1 im Netzwerk 1. Wir tragen jetzt hier bei *Input_Per* unseren analogen Eingang Al0 ein, genauso wie bei *Output_Per* unseren Ausgang AQ.



Schritt 8:

Wir navigieren in der Projektnavigation unter der CPU zum Reiter *"Technologieobjekte -> PID_Compact_1*" und öffnen die Konfiguration durch einen Doppelklick. Dort wechseln wir zum Reiter *"Erweiterte Einstellungen -> Istwertüberwachung*" und tragen eine *obere Warngrenze* von 60 C° ein. Diese Temperatur ist hoch genug für den Würfel, als *untere Warngrenze* wählen wir 5 C°.



Schritt 9:

Wir navigieren noch zum Reiter *Ausgangswertgrenzen* in der Konfiguration. Unser *Obere Ausganswert* beträgt 85 % (durch die automatische Skalierung entspricht dies später 8,5 V am Ausgang), der *untere Ausgangswert* 0 % (entspricht also 0 V). Diese Grenzen hängen mit der Ansteuerung des Transistors zusammen, wir nehmen es hier als gegeben hin (der Lüfter wird so mit voller Leistung betrieben).



Schritt:10

Wir schließen das Fenster der *Konfiguration* und navigieren in der Projektnavigation unter der CPU zum Reiter *"Technologieobjekte -> PID_Compact_1*". Wir öffnen durch Rechtsklick auf den PID-Compact_1 das Menü und klicken dann auf "DB-Editor öffnen". Wir sehen folgendes Bild:

ojektnavigation 🛛 🕅		PID-Regel	ung → A0 [CPU 1214C	DC/DC/DC] +	Technologieobjekte	≥ ► PID_C	ompact_1	[DB1] 🗕 🖷	1
Geräte									
400	•	1 the the							Ø
	*	PID Co	ompact 1						
D PID-Receiung		Nan	ne	Datentio	Startwert	Remanenz	Sichthari	Kommentar	
Neues Gerät hinzufügen			sh ResOld	Bool	FALSE			Intern	
Gerate & Nette		7 403 8	sh TuBeninEver	Bool	FALSE			Intern	
TIM AD TOPU 1214C DC/DC/DCI	10		sh GetOrleTime	Bool	TRUE	Ä		Start der automatischen Ermittlung der Zukluss	
Gerätekonfiguration			sh EnCyclEstimation	Bool	TRUE	ă		Freinahe der Ermittlung der Zykluszeit	
V. Online & Diagnose	NO IN	10	sh EnCyclMonitoring	Bool	TRUE	A		Freinahe der Liebenvachung der Zykluszeit	
Programmbausteine		1 40 1	sh Startun	Bool	FALSE			latera	
Neuen Baustein hi		12 -00 =	sb_RunModeBvStartup	Bool	True	- A		Freigabe des Startens aus dem letzten Zustant	
A Main (OB1)	6	13 -00 =	sby EsData 1	Byte	8#16#0		ă	intern	
PID-Regelung (OB		4	sby EsData 2	Byte	B#16#0			intern	
	-	15 -00 =	si TMCnt	Int	C			Intern	
Systembausteine		16 -611 =	si Unit	Int	0			Einheit des Istwertes	
▼ → Technologieobiekte		7 -00 =	si Type	Int	1	Ä		Tvo des Reglers	
Neves Objekt hinz.		8 📶 =	si SveModeBvRes	Int	0		ă	Intern	
PID Compact 1 ID.	5	9 -00 =	sd Warning	DWord	DW#16#0000000			Warnungstext des Reglers	
Kontiguration		20 -60 =	st TMEnd	Time	T# OMS		ă	Intern	
t Inbetriebnahme		21 - 📶 😐	sr TMDiff	Real	0.0			Intern	
Externe Quellen	~	22 - 📶 =	sr TMDiffMax	Real	0.0			Intern	
		23 -00 =	sr TMDiffMaxWed	Real	0.0			Intern	
Detailansicht		24 📶 🔹	sr TMDiffSum	Real	0.0			Intern	
	-	25 🕣 =	sBackUp	Struct				Gespeicherter Parametersatz	
24000 F		26 📶 🔹	sPid Calc	Struct				Daten fuer die Selbsteinstellung	
lame		27 🥢 🔳	sPid Cmpt	Struct				Daten fuer den Reglerteil	
Programmbausteine		28 📶 🛎	sParamCalc	Struct				Daten zur Parameterneuberechnung	
lechnologieobjekte		29 📶 🗖	- sRet	Struct				Retaindaten	
Externe Quellen		30 📶	b EnableManOld	Bool	EALSE T		Ā	Intern	
PLC-Variablen		31 📶	i Mode	Int				Betriebsartenanwahl (O=Inaktiv;1=SUT;2=TR;3=	
PLC-Datentypen		32 -50	i ModeOld	Int	0	V	ă	Intern	
Beobachtungs- und For		33 📶	i SvetvlodeByEntvlan	Int	0		ā	Intern	
Programminformationen		34 📶	i StateOld	Int	0			Intern	
lextlisten		25 400	a Ctd Cain	Real	8 060224			Aktive Proportionalizerstaerkupp	

Unter *"sRet -> i_Mode"* setzen wir den Startwert auf 3 (Automatikmodus), damit die SPS immer sofort in diesem Modus startet.

Schritt 11:

Wir schließen das Fenster des *DB-Editors* und navigieren in der Projektnavigation unter der CPU zum Reiter *"Technologieobjekte -> PID_Compact_1"* und öffnen die *Inbetriebnahme* durch einen Doppelklick. Wir sehen folgendes Bild:

- 25 - 25	Projektnavigation	PID-Reg	elung 🔸	AO [CPU	1214C.DC/DC/DC] + Technologieobjekte + PID_Compact_1 [DB1]	_ # = ×
		0	ptimierun	Ig		
	Online & Diagnose Programmbausteine Neuen Baustein hin Nain [OB1] PIO-Receiving [OB30]		Mess	ung:	Abtastæit: 0.3 💌 s 🕨 Start	Start
	 FC_1 [FC1] Systembausteine Programmessou Kenfauration Meus Variablen Alle Variablen anzei Neue Variablentabe Detailansicht Name 	inbeztiebnahmeedtor New Thema	🛤 Setpoint/input ["C] 📕 🦉	 ● t ● t 50,000 40,000 30,000 20,000 10,000 0,000 	Static • • Setpoint: Input: • Output: •	
					11:39:30 11:40:00 11:40:30 11:41:00 13:12:11:41:00 13:12:11-11:41:00 13:11-11:41:00 13:11-11:41:00 13:11-11:41:100 13:11-11:41:11-11:41:100 13:11-11:41:100 13:11-11:41:100 13:11-11:41:100 13:11-10:11:41:100 13:11-10:11-10:11:11:41:100 13:11-10:11:100 13:11-10:11:100 13:11-10:11:100 13:11-10:11:11:100 13:11-10:11:11:100 13:11-10:11:11:100 13:11-10:11:100 13:11-10:11:11:100 13:11-10:11:100 13:11-10:11-10:11:100 13:11-10:11:100 13:11-10:11:100 13:11-10:11	i1:19 □ ▼
			Status Fo	Optimie ortschritt: Status:	Ung Online-Zustand des Reglers Setpoint:	×

Schritt 12:

Jetzt laden wir das gesamte Projekt in die SPS und schalten danach die Lampe durch SO ein. Wir drücken im *Inbetriebnahmefenster* bei der *Erstoptimierung* auf *Start*. Es erscheint ein Fehler, weil unsere Isttemperatur nicht 50% größer als die Solltemperatur ist. Deswegen warten wir nun bis die Isttemperatur bei ca. 47 C° angelangt ist. Wir können den Verlauf mit dem Graphen verfolgen.

100	Optimierung	9			
Neuen Baustein hi… ∧ Nain (OB1) PID:Regelung (… FC_1 [FC1]	Messu	ing:	Abtastzeit: 0.3 v e Stop	Optimierungsart: Erstoptimierung]
▼ Systembaustei ● ▼ Septembaustei ●	€, €,	@‡ ⊝;	Strip •		
Technologieobjekte	<u></u>	65.000	• Setpoint: 13.12.2011 11:59:02.547 : 34 Input: 13.12.2011 11:59:02,547 : 37,249		85,000
✓ ₩ PID_Compact ●	5	50,000	Output: 13.12.2011 11:59:02,547 : 0		-70,000
inbetriebnahme	E				60,000
Externe Quellen DLC-Variablen	put [3	40,000 -			50,000
Alle Variablen anz	int/In	30,000			40,000 trid
Standard-Variable	Setpo	20.000			30,000
		20,000			20,000
Detailansicht	1	10,000			10,000
Name	ŝ	0.000			0,000
			11:57:30 11:58:00	11:58:30 11:59:0	0
			13.12.11 - 11:57:02	13.12.11 - 11:59:02	
	Status	Optimie	rung C	Online-Zustand des Reglers	
	For	rtschritt:		Setpoint:	
< 1		Status:	I Fehler beim Starten der Erstöptimierung. I	34.0	>

Schritt 12:

Auch beim *Online Zustand des Reglers* können wir Ist- und Solltemperatur beobachten. Bei einer Isttemperatur von ca. 47 C° wählen wir nun oben die *Optimierungsart "Nachoptimierung"* aus und drücken auf *Start*. Wir überspringen die Erstoptimierung, da die Nachoptimierung nachher viel genauer und besser arbeitet (die Erstoptimierung ist sozusagen eine schnelle Inbetriebnahme). Der Automatikmodus fängt nun an die Regelstrecke auszutesten und berechnet sich daraus selbstständig optimale Werte für den P-, D- und I-Anteil der Regelung. Im optimalen Fall steht nach einiger Zeit im Statusfenster "System ist optimiert." Ansonsten wiederholen wir die Nachoptimierung.

War die Nachoptimierung erfolgreich klicken wir noch *"PID-Parameter laden"* an, um die berechneten Werte aus der CPU in unser Projekt auf dem Rechner zu laden. Wird dies nicht gemacht, sind die P-, I- und D-Parameter nicht im Projekt gesichert!



Wir haben erfolgreich eine PID-Regelung programmiert, konfiguriert und in Betrieb genommen. Beobachten sie den Graphen, wie der Ausgang ständig an die Differenz zwischen Soll- und Istwert angepasst wird und wie genau der PID-Regler die Temperatur ausregelt. Es sind nur Differenzen von $\pm 0,2$ C° zu beobachten.