

Automatisierungstechnik nach internationaler Norm programmieren (11)

Autor: Dr. Ulrich Becker
Fachzentrum Automatisierungstechnik und vernetzte Systeme im BTZ Rohr-Kloster
Mail: Ulrich.Becker@BTZ-Rohr.de

Analogwertverarbeitung, Sprungbefehle und komplexe Datentypen

Mit den Folgen 8 bis 10 haben wir das erforderliche Wissen über Variablen, Funktionen und Funktionsbausteine erworben, um uns nunmehr einigen speziellen Problemen der Automatisierungstechnik zuwenden zu können. Eines davon ist die Verarbeitung analoger Signale. Mit wenigen neuen Anweisungen und Datentypen können nun auch komplexere Aufgaben bearbeitet werden.

Busklemmen für die Verarbeitung analoger Signale

Die Automatisierungstechnik verarbeitet neben binären und digitalen Signalen auch zahlreiche analoge Signale wie Temperaturen, Drücke, Feuchten, Drehzahlen oder Widerstände. Sensortechnik und Messwertumformer liefern für diese physikalischen Größen genormte analoge Werte wie beispielsweise $-1V \dots +1V$, $0 \dots 10V$ oder $4 \dots 20 \text{ mA}$. In immer mehr Technikbereichen wird traditionelle analoge Signalverarbeitung durch digitale Verarbeitung abgelöst. Die erforderliche Signalwandlung übernehmen Analog-Digital-Wandler bzw. umgekehrt – wenn analoge Signale ausgegeben werden müssen - Digital-Analog-Wandler. Diese können heute auch in Busklemmen eingebaut werden, die sich in ihrer Größe nicht von digitalen Klemmen unterscheiden (**Bild 59**). Aus dem vielfältigen Sortiment von analogen zwei- und vierkanaligen Eingangsklemmen für Spannung, Strom, Widerstand oder Thermoelemente trägt das Trainingsrack die Klemme WAGO-750-463 (www.wago.com). Sie beinhaltet zwei Kanäle $0 \dots 10V$ und wurde gewählt, weil dieses Signal leicht mit einem Labornetzgerät vorgegeben werden kann. Jeder Kanal beansprucht ein Eingangswort. Der erste Kanal schreibt in das Eingangswort mit der Adresse %IW0.

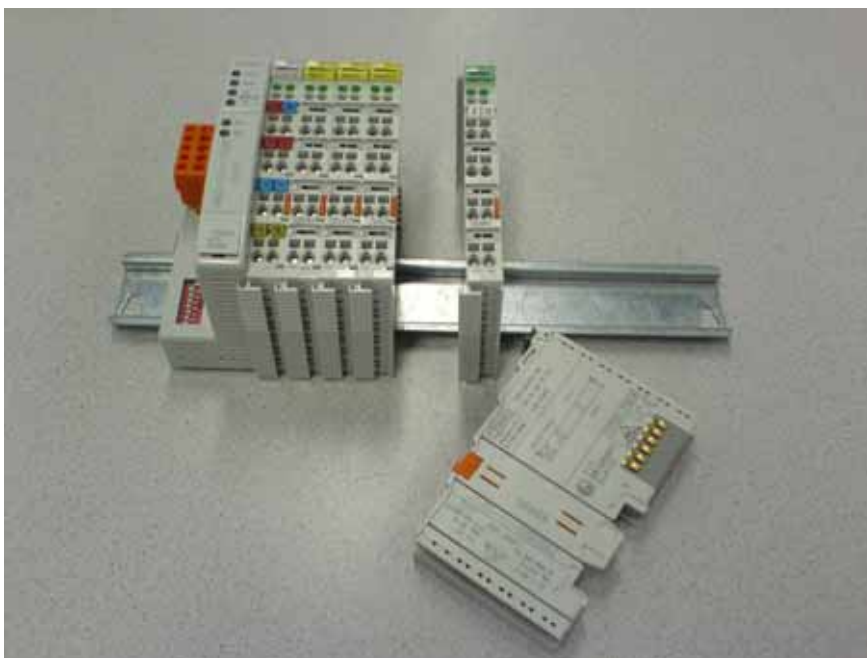


Bild 59: Analoge Klemmen (rechts) des Systems WAGO-I/O-750 (www.wago.com)

Das Ausgangssignal eines Analog-Digital-Wandlers ist zumeist ein Wort von 16 Bit Breite. Die Zahl der darin genutzten Bits wird durch die Auflösung des Wandlers bestimmt. So hat die analoge Eingangsklemme WAGO 750-467 für Signale 0..10V eine Auflösung von 12 Bit. Beobachtet man online das Eingangswort im Format „*Dezimal*“, so liefern 10V Eingangsspannung 32 761 Einheiten. Das bedeutet, dass im 16 Bit breiten Wort 12 Bit gemäss **Bild 60** belegt werden. Kleinstmögliche Änderungen des Digitalwertes bewirken deshalb Sprünge mit der Wertigkeit $2^3 = 8$, wodurch sich der Analogwert nur in Stufen von 2,44 mV verarbeiten lässt. Nachfolgendes Beispiel demonstriert Möglichkeiten der Einbindung von Analogwerten in CoDeSys-Programme.

WORD im PAE / PAA															
X	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	X	X	X
X	I	I	0	I	I	I	I	0	0	I	0	I	X	X	X
28 456 Einheiten entsprechen 8,68 V															

Bild 60: Beispiel eines digitalisierten Analogwertes mit 12 Bit Auslösung. Die mit X gekennzeichneten Bit sind nicht relevant.

Aufgabenstellung 4: Waage mit analogem Ausgangssignal

Am Platz1, von dem aus Teile abtransportiert werden, sei eine Waage mit analogem Ausgangssignal angeordnet (**Bild 61**). Diese sei so justiert, dass sie für Massen von 0 ... 200 Gramm ein Ausgangssignal von 0 ... +10V liefert. Nur Teile mit Massen ≥ 175 g sollen in das Lager transportiert werden. Teile unter 175 g sind dagegen auszusondern und nur bis zum Platz 3 zu transportieren. Die aktuellen Massen sind in eine Tabelle einzutragen, und nach 10 Einträgen jeweils ist ein Mittelwert zu ermitteln .

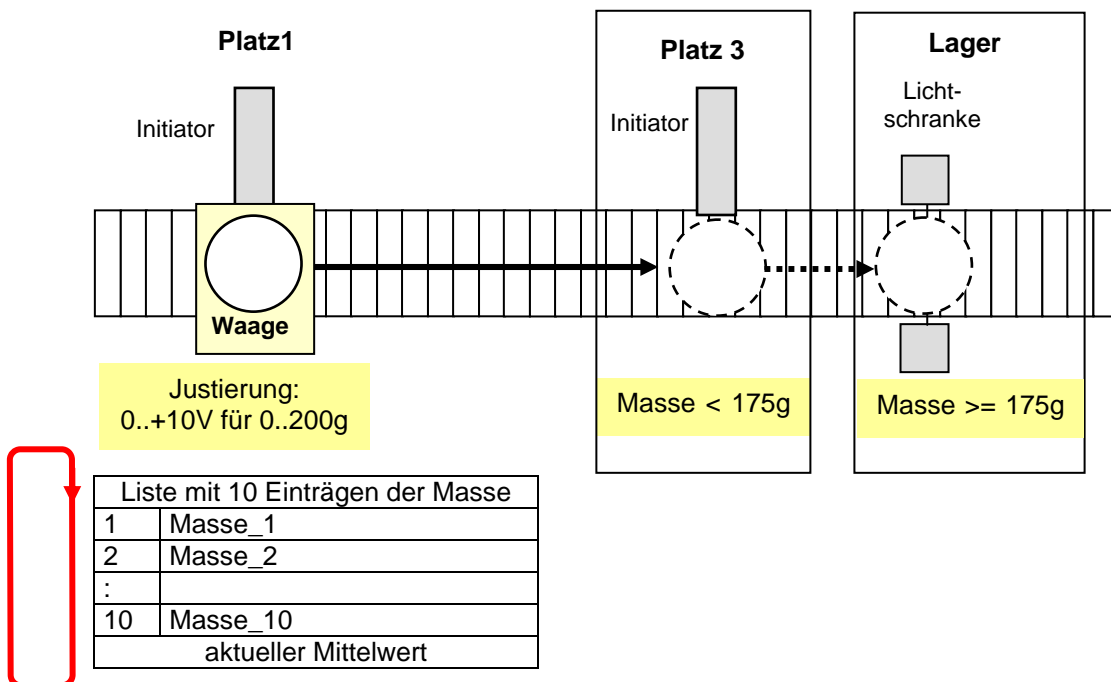


Bild 61 :Aufgabenstellung 4: Einbindung einer Waage mit analogem Ausgangssignal

Eine erste Methode der Auswertung von Analogwerten ist die arithmetische Umrechnung der im Eingangswort des Analogkanals abgelegten Einheiten: Die aktuelle Masse in Gramm im Bereich 0 .. 200g erhält man durch die Operationen „*Analogwert in Einheiten / 32 761 Einheiten * 200 g*“. Mit einem Vergleicher kann dann ein Steuerbit für die Sortierung jeder gewünschten Masse in den Grenzen 0..200g erzeugt werden.

Neben Funktionen und Funktionsbausteinen, die wir in den vergangenen Folgen ausführlich behandelt haben, kennt IEC 61131-3 weiter auch POE's vom Typ PROGRAM. Wir nutzen hier diesen Typ und lösen die Aufgabe mit folgender Anweisungsliste:

```

PROGRAM Waage
VAR
Analogwert_Masse AT %IW0 :INT;      (*Ausgangssignal der Waage*)
Masse:INT;                          (*Masse der Teile in Gramm*)
Untergewicht:BOOL;                  (Steuerbit für Steuerung des Teiltransportes*)
END_VAR

LD Analogwert_Masse
DIV 32 761.0
MUL 200
REAL_TO_INT
ST Masse                             (* Masse in den Grenzen 0..200g*)

LD Masse
LT 175                               (*Vergleich auf kleiner oder gleich 175g*)
ST Untergewicht

```

In diesem Programm erscheint mit der Vergleichsoperation LT ein neuer IEC Operator. Die Liste der Operatoren Tabelle 11 in Folge 9 wird deshalb in **Tabelle 12** ergänzt.

Operation	Erklärung	In Step 7:
GT	Vergleichsoperator grösser als	> I, > DI, >R
GE	Vergleichsoperator grösser oder gleich	>=I, >=D, >=R
LT	Vergleichsoperator kleiner als	<I, <D, <R
LE	Vergleichsoperator kleiner oder gleich	<=I, <=D, <=R
EQ	Vergleichsoperator Gleichheit	==I, ==D, ==R
NE	Vergleichsoperator Ungleichheit	<>I, <>D, <>R
Hinweis:	Die Operanden aller IEC Vergleichsoperatoren können vom Typ BOOL, BYTE, WORD, DWORD, INT, DINT, REAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME und STRING sein. In Step 7 müssen häufig spezielle Operatoren für spezielle Datentypen gewählt werden!	
JMP	unbedingter Sprung zur Marke	JMP
JMPC	Sprung zur Marke, wenn Ergebnis des vorherigen Ausdrucks TRUE ist	JMPC
JMPCN	Sprung zur Marke ,wenn Ergebnis des vorherigen Ausdrucks FALSE ist	JMPCN

Tabelle 12: IEC Operatoren (Teil 2)

Bild 62 zeigt, dass die Analogwertauswertung fehlerfrei arbeitet. Im Simulationsmodus wurde ein Digitalwert von 28456 Einheiten geschrieben (vergl. Bild 60). Dies entspricht der Spannung von 8,68 V und der Masse 173,6 g (als Integer gerundet 174g). Folgerichtig hat die Boolesche Variable „*Untergewicht*“ den Wert TRUE. Dass der Divisor mit 200.0 als REAL eingegeben werden muss, scheint ein internes Problem von CoDeSys zu sein. Durch die Wandlung REAL_TO_INT wird dieses Detail nachfolgend korrigiert.

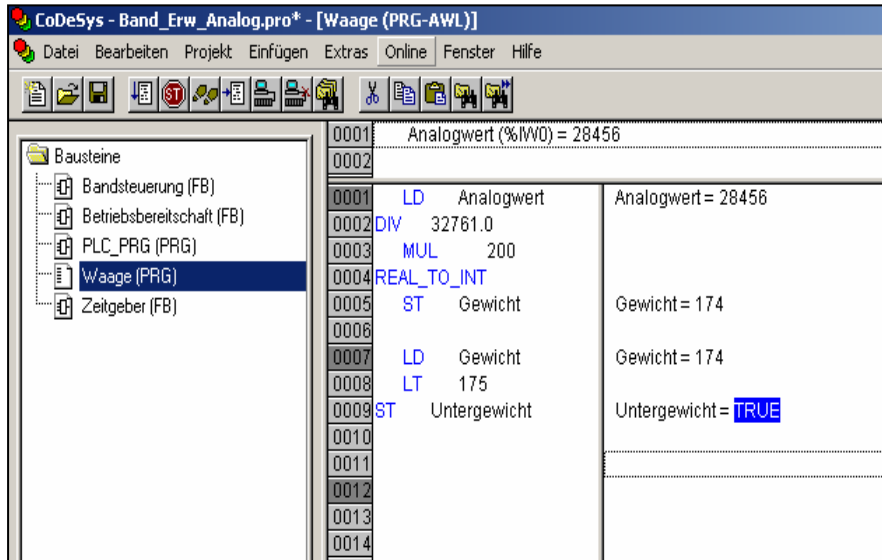


Bild 62: Online-Beobachtung der POE Waage (PRG) im Simulationsmodus

Eine andere Möglichkeit zur Verarbeitung von Analogwerten ist die Verwendung geeigneter Bibliotheken. Auf der Homepage www.wago.com findet man die Bibliothek „*Gebauede_allgemein*“. Hier werden Werkzeuge bereitgestellt für den Einsatz von Busklemmententechnologie in der Gebäudesystemtechnik. Die Funktion „*Fu_Linear-2punkt*“ beschreibt auf der Basis zweier Referenzpunkte eine lineare Kennlinie und gibt auf dieser Grundlage zu jedem (analogen) Eingangswert eine normierte Ausgangsgröße aus. Alle ihre Parameter sind vom Typ REAL. **Bild 63** zeigt oben die graphische Darstellung mit den Werten gemäss Beispiel Bild 60 und unten die AWL (siehe Folge 9). Diese Funktion ist vergleichbar mit den Funktionen FC 105 und FC106 im System Step 7 für die Skalierung von Analogwerten.

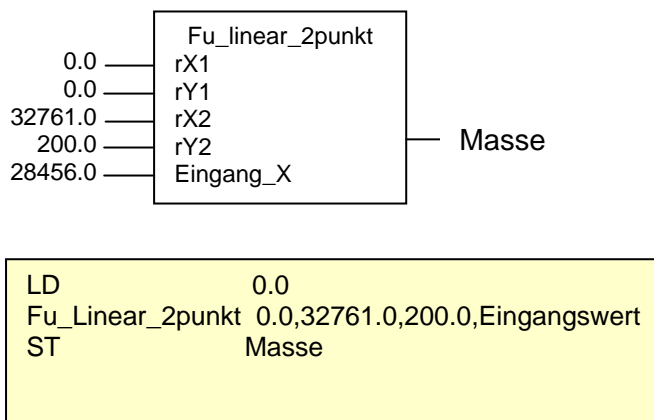


Bild 63: Nutzung des Bibliotheksbausteins „*Fu_linear-2punkt*“ für die Skalierung von Analogwerten

Der interessierte Leser sollte nun in der Lage sein, die POE „Motorsteuerung“ mit Hilfe des Steuerbit „*Untergewicht*“ so zu modifizieren, dass untergewichtige Teile nach Start mit Taster „S1“ nur bis zum Initiator 3 gelangen. Selbstverständlich muss diese Variable dann global deklariert und die POE „*Waage[PRG]*“ im Hauptprogramm aufgerufen werden! Als Ergebnis zeigt **Bild 63** das Detail „*Motorsteuerung*“ in graphischer Sprache.

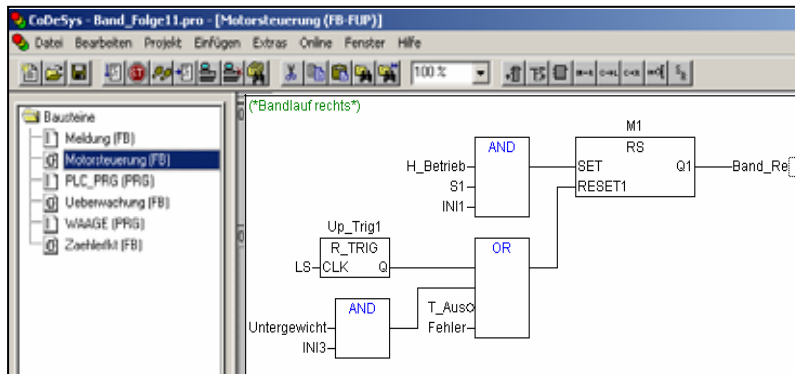


Bild 63: Einbinden der Variablen „Untergewicht“ in die Steuerung des Bandantriebs

Nunmehr wollen wir uns dem etwas komplexeren Problem der Aufgabenstellung 4 zuwenden: Die aktuellen Massen sollen in eine Tabelle eingetragen und ein Mittelwert ermittelt werden. Dieses Problem scheint aufwendig und könnte in Step 7 nur mit Methoden der indirekten Adressierung gelöst werden. Dabei würden die Listeneinträge in Doppelworte eines Datenbausteins geschrieben, die programmtechnisch indirekt fortlaufend zu adressierten sind. Es zeigt sich, dass eine IEC-gerechte Lösung mit wenigen Anweisungen gefunden werden kann, von denen uns darüber hinaus die meisten bereits bekannt sind.

In der nachfolgenden AWL wird das Startsignal des Tasters „S1“ für den Eintrag eines aktuellen Wert in die Liste genutzt. Die Liste erstellen wir mit einem Datentyp ARRAY. Dieser gehört zu den Standarddatentypen und umfasst eine Menge gleichartiger elementarer Daten wie z.B. INT, REAL oder WORD. ARRAY's können lokal oder global deklariert werden.

Im Beispiel wird mit der Deklaration „*Liste: ARRAY [1..10] OF INT;*“ ein Feld von 10 Daten des Typs INT bereitgestellt. Einen speziellen Eintrag daraus spricht man mit dem Index an. Der Index läuft von 1 bis 10 und ist deshalb ebenfalls vom Typ INT. Mit jedem Eintrag wird ein Zwischenwert als Summe der bisherigen Listeneinträge berechnet. Nach zehn Einträgen errechnet sich daraus der Mittelwert. Da die Masse mit gerundeten ganzzahligen Werten beschrieben wird, sind auch Zwischenwert und Mittelwert vom Typ INT.

```

PROGRAM PLC_PRG
VAR
Wert: INT:= 178; (*aktuelle Masse, für die Simulation mit 178g vorgegeben*)
Liste: ARRAY [1..10] OF INT; (*Liste mit 10 Zeilen*)
Index:INT:=1; (*laufende Größe für die Kennzeichnung der Zeilen der Liste*)
Zwischenwert: INT; (*Hilfsgröße für Berechnung des Mittelwertes*)
Mittelwert: INT; (*Mittelwert aus 10 Zeilen der Liste*)
S1 AT %IX2.5: BOOL; (*Taster S1 löst Listeneintrag aus*)
Flankenbewertung: R_TRIG; (*nstanz des Standardfunktionsbausteins*)
END_VAR

LD S1 (*Flankenbewertung des Startsignals Taster S1*)
ST Flankenbewertung.CLK
CAL Flankenbewertung

LD Flankenbewertung.Q (*Ergebnis: Nur bei Betätigung des Tasters S1 wird
JMPCN Ende nachfolgendes Programm bearbeitet, sonst Sprung
zur Marke Ende und damit keine Reaktion*)

LD Wert (*Eintrag der aktuelle Masse in Liste*)
ST Liste[Index]

LD Zwischenwert (*Aufsummieren der Listeneinträge*)
ADD Liste[Index]
ST Zwischenwert

LD Index (*Nach Programmdurchlauf Index um 1 erhöhen*)
ADD 1
ST Index

LD Index (* Nur bei Index >10 nachfolgendes Programm,*)
GT 10 (*sonst Sprung zu Programmende*)
JMPCN Ende

LD Zwischenwert (*Mittelwertbildung und Rücksetzen des*)
DIV 10 (*Zwischenwertes*)
ST Mittelwert
LD 0
ST Zwischenwert

LD 1 (*Rücksetzen des Index auf 1*)
ST Index

Ende: (*Sprungmarke mit Namen "Ende"*)

```

Neu in dieser AWL ist die Verwendung von Sprüngen zu Marken (Tabelle 12). Mit diesen Operatoren kann man verhindern, dass Programmteile bearbeitet werden oder dass ausschließlich bestimmte Programmteile wirksam werden. Ein Sprung erfolgt immer zu einer Sprungmarke (Label), an welcher das Programm fortgesetzt wird. In FUP stellt man Sprünge gemäss **Bild 64** dar.

Bild 65 zeigt einen ersten Test des Programms durch Simulation. Mit -> *Werte schreiben* wurde ein gleichbleibendes Gewicht von 178g vorgegeben. Durch Schreiben des Booleschen Wertes S1 -> TRUE -> FALSE -> TRUE erfolgten zwei Listeneinträge. Danach steht der Index auf dem Wert 3.

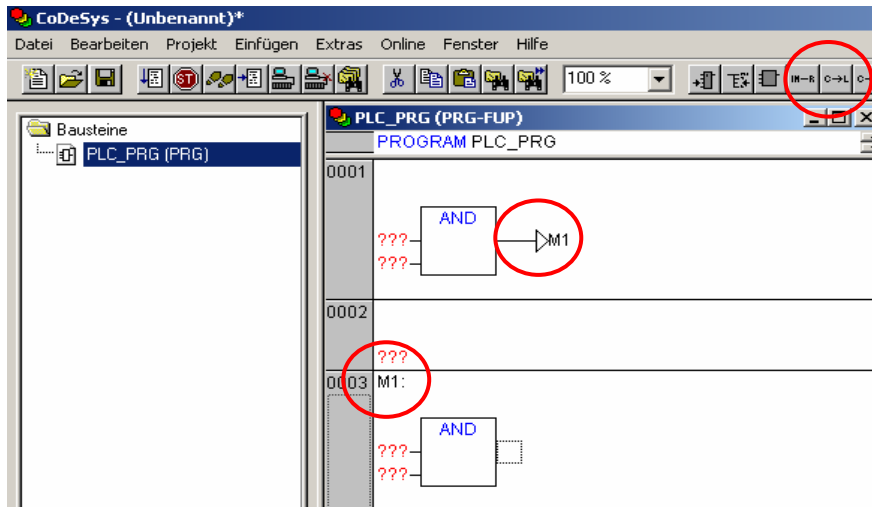


Bild 64: Sprünge in Darstellung FUP

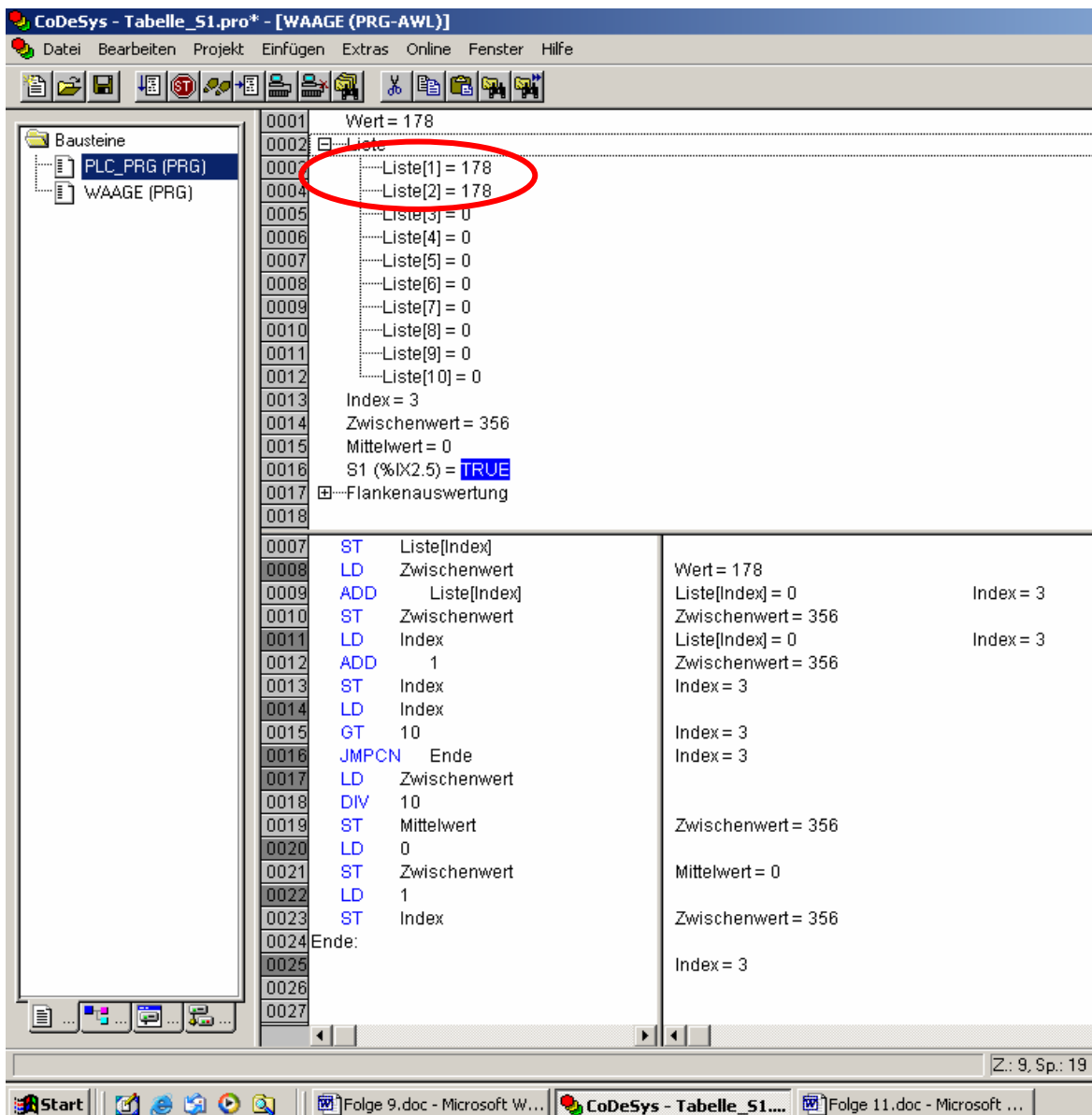


Bild 63: Online-Beobachtung zweier simulierter Listeneinträgen

Fazit:

In dieser Folge wurden Möglichkeiten der Einbindung analoger Signale in Programme nach IEC 61131-3 aufgezeigt. Zum einen gelingt dies durch einfache Arithmetik mit den digitalisierten Werten, zum anderen stehen auch Bibliotheken zur Verfügung. Die Übungen zur Programmierung mit Anweisungslisten wurden durch Einführung von Sprungbefehlen und des Datentyps ARRAY vertieft. Bei Inbetriebnahme und Beobachtung von Programmen ist mitunter die Visualisierung von Zusammenhängen hilfreich. Deshalb wird in der nächsten Folge in die Visualisierung eingeführt, welche im Programmiersystem CoDeSys integriert ist.

Glossar:

Analoge Signale	Analoge Signale nehmen in ihrem Wertebereich alle denkbaren Zwischenwerte an. Im Gegensatz dazu können digitale Signale nur diskrete Werte in bestimmten Stufen durchlaufen.
ARRAY	Variablenfeld. Mit Arrays vereinbart man eine Menge aufeinanderfolgender Daten gleichen Typs, z.B. INT oder WORD. Das einzelne Datum des Feldes spricht man dann mit seinem Index an. Arrays können lokal im Deklarationsteil der POE oder aber global deklariert werden. Neben dem vorgestellten eindimensionalen Feld ermöglicht IEC 61131-3 für komplexe Datenhaltung auch zwei- und dreidimensionale Felder. Im Gegensatz zu Arrays verwalten Strukturen Daten unterschiedlichen Typs.
Auflösung	Bei Analog-Digitalwandlern (ADU) bezeichnet die Auflösung, in wieviel Bits der gewandelte Analogwert geschrieben wird. Damit wird auch die kleinst mögliche Stufung der Digitalwerte festgelegt.
binäres Signal	Signal, welches nur zwei Werte annehmen kann: FALSE oder TRUE als logische Werte bzw. beispielhaft 4 mA oder 20 mA.
digitales Signal	Signal, welches im Wertebereich nur diskrete Werte in festgeschriebenen Stufungen annehmen kann (siehe Auflösung).
Gebäudesystem-technik	Sammelbegriff für technische Systeme in Zweckbauten für automatische Überwachung, Steuerung und Regelung sowie Visualisierung .
indirekte Adressierung	Unverzichtbar bei Programmierung in Step 7. Bei indirekter Adressierung liegt die Adresse eines Operanden nicht unveränderlich fest, sondern wird erst zur Laufzeit des Programms und evtl. wechselnd festgelegt. Das dafür notwendige Format - welches u.a. Adressbyte, Adressbit und evtl. den Operandenbereich enthalten muss - nennt man Zeiger (Pointer). Es wird mit P# bezeichnet. Einfaches Beispiel: Nach L P#30.0 und T MD10 ist die Anweisung L M [MD10] identisch mit L M30.0
Struktur	Zusammenfassung unterschiedlicher Datentypen. Diese wird in die Schlüsselworte SRUCT und END_STRUCT eingeschlossen