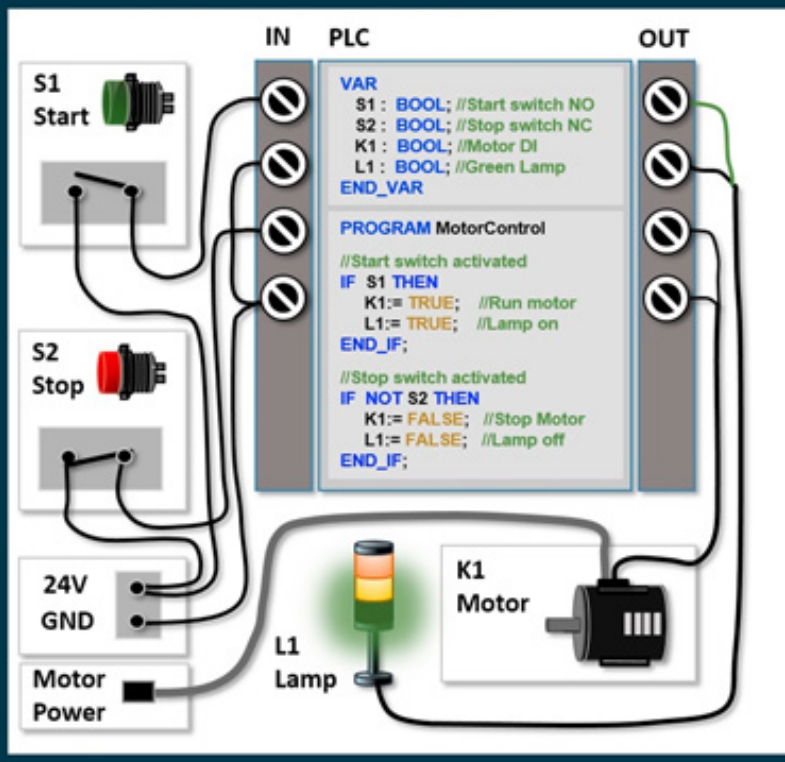
Tom Mejer Antonsen

PLC styring med Structured Text (ST)



IEC 61131-3 og best practice ST-programmering

3. udgave

FORORD

Da jeg i august 2016 startede hos Erhvervsakademi Dania som underviser på Automationsteknolog-uddannelsen (en 2-årig videregående uddannelse), var en af mine opgaver, at finde bøger der kunne bruges i undervisningen. Specielt inden for programmering i Struktureret Tekst (ST) var der ikke meget. Bøger på 300 sider havde kun få sider med ST-programmering og på et højt teoretisk niveau. De studerende efterspurgte eksempler og metoder til ST-programmering. Det betød, at jeg i januar 2017 begyndte at skrive på et materiale, der fik navnet:

"**Kom i gang med Structured Text**"

Siden da, er materialet løbende blev opdateret, udvidet og brugt i undervisningen. Materialet har været meget efterspurgt blandt de studerende og nu er det blevet til en bog, så andre kan få glæde af indholdet.

Jeg håber du bliver glad for bogen.

Tak til studerende og undervisere for feedback og inspiration.

Kommentarer, ris og ros samt forslag til forbedringer modtages gerne. Send dem til TomMejerAntonsen@gmail.com

1. Udgave udkom i marts 2018

2. Udgave udkom i januar 2019 (opdateret og med flere eksempler)
3. Udgave udkom i maj 2020 (antal sider er øget fra 128 til 208)

Denne 3. udgave er opdateret og udvidet med mange af de forslag og spørgsmål som læserne og de studerende er kommet med, herunder ønsket om mange flere illustrationer og programeksempler.

Bogen er også udgivet på engelsk og som e-bog.

God fornøjelse!

Tom Mejer Antonsen

Randers, maj 2020

Indholdsfortegnelse

1 Indledning

Denne bog giver en introduktion til programmeringssproget **S**truktureret **T**ekst (ST), der benyttes i **P**rogrammerbare **L**ogiske **C**ontrollere (PLC).

Bogen går systematisk frem med beskrivelse af de grundlæggende ST-begreber og programmering, herunder tips med inddragelse af forfatterens praktiske erfaring.

Der er mange steder uddybende forklaringer til PLC-koden og der er fokus på at læseren lærer at skrive robust, læsbar, struktureret og overskuelig PLC-kode. Desuden fokuseres der på at kunne skrive PLC-kode, som ikke kræver en bestemt PLC-type og PLC-kode der kan genbruges, og PLC-løsninger, der kan benyttes internationalt.

I overskrifter er de engelske udtryk skrevet i parentes, så læseren nemt kan bruge disse ord til at finde mere information på internettet. Erfaringen viser dog, at det ikke er helt nemt at finde kode-eksempler og programmeringsforslag inden for ST.

Det anbefales at læse bogen helt igennem for at få overblik over indhold og bagefter bruge bogen som opslagsværk.

Bogen er primært udarbejdet til brug på den 2-årige videregående fuldtidsuddannelse **Automationsteknolog** og deltidsuddannelsen **Automation og Drift.**

Bogen tager udgangspunkt i standarden IEC 61131-3. PLC leverandører fortolker standarden forskelligt og ikke alle leverandører følger standarden, selvom de giver udtryk for

dette på deres produkter. Dette betyder at nogle af program eksemplerne her bogen måske ikke virker i den PLC type du bruger.

I en Siemens PLC hedder det programmering i **S**tructured **C**ontrol **L**anguage (SCL), og der er nogle mindre forskelle i forhold til IEC 61131-3 standarden.

Der gives ingen garanti eller support på PLC-kode eksemplerne i bogen.

1.1 Baggrund for ST

ST er et højniveau programmeringssprog, der ligner Pascal programmering. Pascal programmering var meget benyttet i Danmark i perioden fra år 1985 til omkring år 2000,- en periode hvor mange virksomheder begyndte på udvikling af software til PC, først DOS og siden Windows.

Den første udgave af ST blev udviklet og udgivet i 1993 af the **I**nternational **E**lectrotechnical **C**ommission (IEC) under IEC 61131-3, som er en international standard. Standarden indeholder fem PLC programmeringssprog, hvor Ladder programmering (**L**adder **D**iagram LD) er det mest kendte og udbredte. Den seneste danske udgave DS/EN 61131-3 er fra 2013. Der forventes en udgave i 2020/2021. De andre programmeringssprog er **F**unction **B**lock **D**iagram (FBD), **I**nstruction **L**ist (IL) samt **S**equential **F**unction **C**hart (SFC).

ST-programmering til PLC-styringer er fra omkring år 2010 begyndt at blive mere udbredt i Danmark, og siden år 2015 er mange virksomheder i Danmark begyndt udelukkende at levere PLC-styringer, hvor der benyttes ST som det foretrukne programmeringssprog. Dette kræver at flere

medarbejdere kan ST, og det er et af argumenterne for at udgive denne bog.

1.2 Forudsætning for at lære ST

Det er ikke en forudsætning at kunne Ladder-programmering, men ”et vidst kendskab” til matematik, fysik, mekanik, elektronik, maskiner, automation og grundlæggende PLC-kendskab, er nødvendigt for at lære ST-programmering.

Bogen er skrevet på dansk, så sproget ikke er en forhindring for at lære ST-programmering. Dog er variabel navne på engelsk, da ikke alle PLC-typer tillader danske tegn. PLC-kode eksempler og illustrationer er på engelsk, da mange virksomheder bruger engelsk i deres PLC-kode.

De som er uddannet i, eller har erfaring med, et højniveau programmeringssprog til PC (f.eks. VB, .NET, C, C#, Python), har forholdsvis nemt ved at lære ST, da programopbygningen minder meget om hinanden. Programafviklingen i en PLC er dog anderledes end i et traditionelt PC program eller en Web applikation.

Oplæringstiden for ST er som andre tekstprogrammeringssprog typisk 3 til 5 år.

1.3 Vidensgrundlag

Forfatteren har 25-års erfaring inden for specifikation, udvikling og levering af komplekse styringsløsninger og overvågningssystemer. Heraf 7-års erfaring inden for Pascal programmering samt 12-års erfaring med løsninger og systemer, der indeholder PLC. Erfaringen fra ansættelse i fire internationale virksomheder og levering af mere end

1000 systemløsninger til 20 lande, er således en stor del af grundlaget for indholdet i bogen.

Forfatteren har i fire år undervist i PLC-styring på videregående uddannelser, både deltids- og fuldtidsuddannelser. De studerende har inden studiet fra 0 til 20 års erhvervserfaring inden for PLC, automation og teknisk vedligehold.

Grundlaget for bogen er et materiale, som er udarbejdet løbende med feedback fra undervisere og studerende på automationsteknolog-uddannelsen hos Erhvervsakademi Dania. Materialet er således løbende opdateret, så det giver svar på de spørgsmål og udfordringer, de studerende typisk har gennem deres studie.

Endvidere er internettet, standarden DS/EN 61131-3 og en række bøger inden for PLC-styring brugt som inspiration og afklaring på problemstillinger.

Den første udgave af bogen udkom i marts 2018 og efterfølgende har mere end 100 studerende og bogkøbere bidraget med forslag, forbedringer og ønsker.

Forfatteren er Svagstrømsingeniør (B.Sc.E.E.) fra Århus Teknikum.

1.4 Fordele ved ST-programmering

ST er et meget fleksibelt og universelt programmeringssprog. ST-programkode kan nemt kopieres mellem forskellige PLC-typer og sendes via e-mail, da det er baseret på tekst og ikke grafik som ved Ladder programmering.

ST-programkode ligner sætninger og der arbejdes på samme måde som i et tekstbehandlingsprogram (som f.eks. Microsoft Word), hvilket gør det nemt at arbejde med. Der benyttes således de samme metoder, som når der arbejdes i et tekstbehandlingsprogram.

På grund af sin meget strukturerede natur er ST ideel til opgaver, der kræver kompleks matematik, kodegenbrug eller beslutningstagning (f.eks. automatisk energi optimering, algoritmer, dataopsamling og regulering).

Når man har erfaring med ST-programmering, vil overgangen til andre programmeringssprog inden for PLC-styring og automation være nemmere. Det kan være programmering af robotstyringer, servo- eller Visual Basic -programmering.

De senere år er flere og flere virksomheder gået over til ST-programmering. Dette skyldes, at ST har en række fordele sammenlignet med de fire andre PLC programmeringssprog (LD, SFC, FBD og IL). Disse fordele er:

- ST-programmering kan forholdsvis nemt kopieres mellem forskellige PLC-typer og PLC fabrikanter. [1)]
- Det er det nemmeste PLC-sprog til matematiske beregninger, formler og algoritmer [2)] samt store data mængder (Big Data).
- PLC-løsninger er mere krævende i dag end for 20 år siden [3)]
- Mange udbredte programmeringssprog (C++, C#, VB, Java, PASCAL, Python) minder meget om ST-programstruktur.

- De andre PLC-sprog (LD, SFC, FBD) kan kræve, at noget skal programmeres i ST.
- Det fylder mindre, når PLC-programkode skal dokumenteres, beskrives eller udskrives end ved de andre PLC-sprog.
- Det er det nemmeste PLC-sprog at versionsstyre via kommentarer i programkoden eller via GIT[4)] eller Subversion. [4)]

PLC programmeringssproget **I**nstruction **L**ist (**IL**) som benyttes til komplekse PLC-styringer, forventes at udgå i løbet af nogle år (jf. DS/EN 61131-3 afsnit 7.2.1). ST er naturligt at benytte, hvis og når **IL** udgår.

1.5 Udfordringer ved ST-programmering

En stor udfordring er, at mange teknikere og elektrikere kun kan programmere i Ladder og de kan have det svært med ST-programmering, da det er tekstbaseret og ikke grafik, som det kendes ved Ladder-programmering. **5)**

Programmering i ST kan hurtigt blive uoverskueligt, da det kræver en vis erfaring at strukturere et program hensigtsmæssigt.

For den uerfarne, kan det være svært at fejlfinde (debugge) i et ST-program. De mindste PLC-typer giver normalt ikke mulighed for at benytte ST-programmering. Det er ikke muligt at bruge ST-programmering i en sikkerheds-PLC. (Safety PLC) **6)** Læringstiden for ST-programmering er 3 til 5 år efter afsluttet studie/kursus.

---

1) Det er muligt med Copy-Paste og mindre tilpasninger. F.eks. bruger Siemens ”#” foran lokale variabler og Allen Bradley en anden syntaks til funktionskald.

2) De matematiske beregninger ligner matematiske formler. Se afsnit 8.1 side 47.

3) Der er i dag mere fokus på energi optimering, automatisk drift og dataopsamling. Det er alle løsninger som kræver, at der er brug for mere krævende PLC-kode, end bare en almindelig ”relæ styring” med start/stop funktioner.

4) Værktøjerne GIT og Subversion er gode værktøjer til at spore (følge) rettelser og udvidelser i PLC-koden. Dette sikrer, at det er muligt at hente en tidligere version (udgave) af det pågældende stykke PLC-kode.

5) For at hjælpe de, som er gode til Ladder programmering, indeholder bogen udvalgte Ladder programmer og den tilsvarende ST-programmering. Se side 178.

6) Det er en separat PLC eller specielle områder i en almindelig PLC, der benyttes til at afbryde motorer og andre bevægelige maskindele, hvis nødstop knappen aktiveres. Der skal være 100% garanti for, at afbrydelsen sker og derfor afvikles den kode i en sikkerheds-PLC, som er godkendt til formålet.

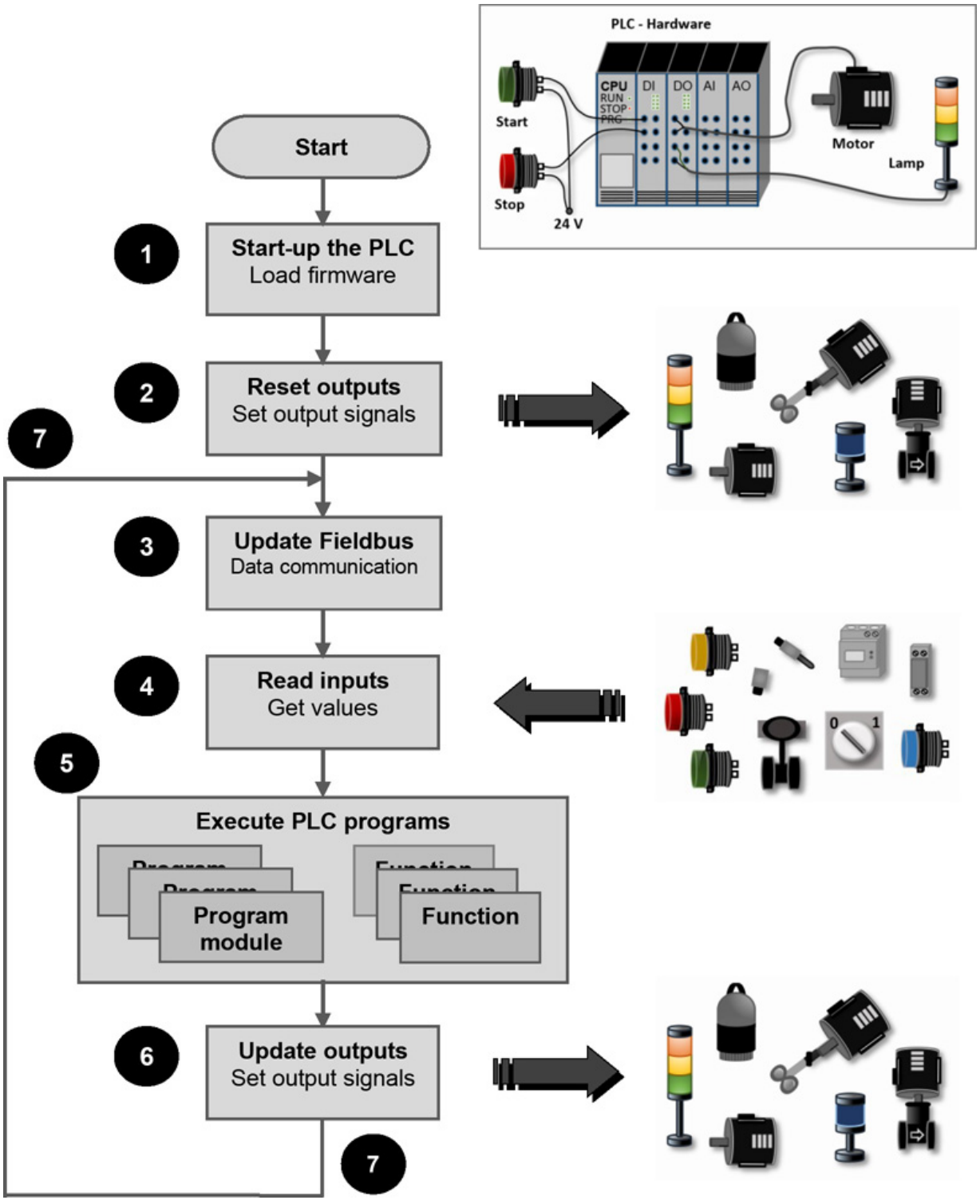
2 Program gennemløb – afvikling af PLC-kode

Det er vigtigt at vide hvordan en PLC afvikler et program, da der skal tages hensyn til dette når programmet skrives.

En PLC afvikler programmer sekventielt i realtid, hvilket betyder, at de enkelte programstumper (programmodulerne) skal være afviklet i løbet af kort tid. Programmodulerne bliver afviklet med et fast interval (PLC scan-time) f.eks. 50 [ms]. Nogle af de hurtigste PLC-typer kan have en scan-tid (cycle time) på 1 [µs].

Det er muligt at have programmoduler med forskellig scan-tider f.eks. 500 [ms] eller hvert minut. Der kan være input signaler fra sensorer, som ikke ændrer værdi hurtigt (f.eks. en temperatur sensor) og derfor er det ikke nødvendigt, at have hurtig scan-tid til alle programmoduler. Et stort program med mange beregninger, tager længere tid at gennemløbe og derfor vil det være nødvendigt, at have forskellige scan-tider på de forskellige programmoduler. Læs mere om dette i afsnit 10.1, side →.

Grundlæggende virkemåde for en PLC:



Flowdiagrammet viser følgende:

1. Når en PLC tændes vil den starte op (boote) og indlæse operativsystemet, som i en PLC hedder firmware. Dette sikrer, at programmet kender den hardware (HW), som er forbundet.

2. Efter opstart sættes udgange til den værdi, de er initialiseret til. Det er vigtigt, at alle udgange har den rigtige startværdi, så maskinen ikke udfører uheldige handlinger, inden PLC-programmet er kommet i gang.

3. Her foretages datakommunikation via netværk (fieldbus). Med datakommunikation modtages og sendes mange værdier til andre enheder (F.eks. betjeningspaneler, andre PLC-styringer eller instrumenter). Der findes mange Fieldbus-typer (f.eks. Profibus, Profinet og EtherNer/IP), men grundlæggede fungerer de ens.

4. Her indlæses værdier fra alle de sensorer, kontakter, afbrydere, instrumenter og komponenter der er på maskinen/anlægget.

5. Afvikling af alle PLC-programmer. Alle programmer gennemløbes (afvikles) én gang afhængig af scan-tid. PLC-programmer er opdelt i:

Programmoduler. Se afsnit 10, side →
Funktioner. Se afsnit 10.2, side →
Funktioner (FC) og Funktionsblokke (FB). Se side →

1. PLC-programmer skal opdeles for at give en god programstruktur.

2. Udlæsning af værdier til alle de fysiske udgange. Det kan være nye indstillinger til motorer, ventiler, lamper og instrumenter.

3. Alle forløb, punkt 3 til 6 gentages. Det kaldes for ét program-scan. Programafviklingen stopper først hvis:

    - PLC-programmet sættes i STOP mode
    - Der sker en fejl i programafviklingen (run time error)
    - PLC slukkes

---

3 Kommentarer i programkode (comments)

Kommentarer er en meget vigtig del af programmeringen. Kommentarer i programmeringskoden hjælper dig og den, som skal rette og tilføje til koden senere.

Brug kommentarer til at forklare, hvad et konkret stykke PLC-kode gør, så du selv kan huske det senere. PLC-koden kan være selvforklarende i mange tilfælde, så det er bedst kun at skrive kommentarer, når noget PLC-kode er komplekst.

Der er to typer af kommentarer i ST:

Linje kommentar

```
// Line comment. Forward-slash is written in front of EVERY line.
// Here you can write comments

//You can write comments before a code section starts
IF S1 THEN
 K1:= TRUE; //Or write comments after the code at the same line
END_IF;

//Do not execute the code below
//K2:= B2
```

En linje kommentar starter altid med **//** og placeres på samme linje foran eller *efter* PLC-koden som vist herover.

En linje kommentar kan også sættes foran programkode linjer som ikke skal afvikles (ud-kommentering). Dette er en fordel frem at slette koden.

Blok kommentar

```
(* Block comment is initiated by a start parenthesis and a star. It is finalized by a star and end parentheses. They are used for making more lines of PLC code inactive i.e. create multiplecomment lines *)
```

Kommentarer placeret mellem **(*** og ***)** kaldes blok kommentar og bruges til at udmaske flere linjer PLC-kode eller skrive kommentarer der fylder flere linjer.

Til hvert program modul eller funktion benyttes kommentarer som overskrift, så en anden programmør hurtigt får en indledning til program modulet eller funktionen. Feltet skal (bør) indeholde en versionslog, så det er muligt at se, hvad der løbende er blevet ændret i koden og af hvem:

```
////////////////////////////////////////////////////////////////////////////////
/////////////
/// OP002 Parking house
////////////////////////////////////////////////////////////////////////////////
/////////////
// Action for each connected sensor
//
//***********************************************
// Version 1.0, Created. Date 06.05.2020 TMA
```

```
// Version 1.1, TempVar3 changed 10.05.2020 TMA
// Version 1.2, Button S1 added 1.05.2020 TMA

IF S1 THEN //First line of PLC code
 K1:= TRUE;
END_IF;

//SetLamps(); //Do not run the SetLamps program module
```

Enkelte PLC-typer kan ikke håndtere, de danske speciel bogstaver som æøåÆØÅ i kommentar linjer. Det anbefales derfor generelt at bruge engelsk i både kommentarer og programmeringen, da danske tegn ofte ikke accepteres og desuden vælger mange virksomheder at skrive PLC-kode på engelsk. En anden grund til at skrive PLC-kode på engelsk er, at mange virksomheder er internationale, og de programmører som skal læse og måske rette i din kode kan ikke læse dansk.

VIGTIGT! Husk at rette kommentarer og versionslog, hvis der efterfølgende bliver ændret noget i PLC-koden.

TIPS: En god idé er, at bruge kommentar linjer til at beskrive, hvad et stykke PLC- kode skal gøre, *inden* man begynder at skrive PLC-koden. Det hjælper en selv, til at få struktur på PLC-koden og hjælper andre programmører til at forstå PLC-koden.

4 Datatyper (Data types)

I lighed med andre programmeringssprog har IEC 61131-3 standarden mange forskellige datatyper, både standard (elementære) og afledte datatyper. En datatype definerer hvor meget hukommelse, der skal bruges til en variabel og derved den største og mindste værdi for variablen.

4.1 Standard datatyper (INT, REAL, BOOL…)

Følgende (udvalgte) er de datatyper, som er standard i de fleste PLC:

Datatype	Bits	Talsystem	Note	Lavest og højeste værdi	Eksempel
BOOL (Bit)	1	Boolean (Boolsk)		FALSE/TRUE eller 0 til 1	TRUE
BYTE	8	HEX (Hexa decimal)		16#0 til 16#FF	16#10
WORD **UINT**	16	Binært tal HEX (Hexa decimal) BCD Heltal uden fortegn (kun positive tal)		2#0 til 2#1111111111111111 16#0 til 16#FFFF C#0 til C#999 0 til 65535	2#0001000000000000 16#1000 C#998 564
DWORD (Double word)	32	Binært tal		2#0 til 2#1111111111111111 1111111111111111	2#10000001000110001 011101101111111
		HEX (Hexa decimal)		16#00000000 til 16#FFFFFFFF	16#00A21234
		Heltal uden fortegn (kun positive tal)		0 til 4294967295 (4.29 mia.)	435
INT (Integer)	16	Decimal Heltal med fortegn		-32768 til 32767	101
DINT (Double integer)	32	Decimal Heltal med fortegn		-2147483648 til 2147483647 (2.1 mia.)	107
REAL (Floating-point number)	32	IEEE 754 Floating-point number (Decimal tal)	1	Laveste tal: +/-3.402823E+38 Højeste tal: +/-1.175495E-38	1.234567e+13
LREAL (Long Real)	64	Dobbelt Float (Decimal tal) IEEE 754		Laveste:-1.7976931348623E308 Højeste: 1.79769313486232E308	3432.54
TIME (IEC time) **LTime**	32 64	IEC tid opløsning: 1 [ms] el. 1 [ns]	4	T#1ns til T#24d20h31m23s	TIME#10s T#10d14h11m23s T#5s12ms23us300ns
DATE (IEC date)	16	IEC dag, Opløsning 1 dag		D#1990-1-1 til D#2168-12-31	D#1996-3-15 DATE#1996-3-15
TIME_OF_DAY (Time)	32	Tid i en opløsning på 1 [ms]	4	TOD#0:0:0.0 til TOD#23:59:59.999	TOD#1:10:3.3 TIME_OF_DAY#1:10:3.3
CHAR **WCHAR**	8 16	ASCII karakter (1 bogstav)	2	'A', 'B' etc.	'E'
STRING		Tekst og tegn	3	Op til 255 tegn	”Dette er en tekst”

Alle variabler skal have en datatype. Hvis en variabel bliver tildelt en værdi, der er uden for min. og max. for variablens datatype område, kan der opstå et overløb på variablen eller run time error (program fejl) og PLC kan stoppe

programafviklingen. Det kan også medføre en underlig opførsel under programafviklingen (programmet kan virke ustabilt).

Enkelte PLC-typer kan have flere datatyper end dem, der er listet herover. Generelt anbefales det, at holde sig til nogle få datatyper, så PLC-koden nemmere kan kopieres til andre PLC-typer. F.eks. kan specielle datatyper som **S5TIME**, **LWORD** og **ULINT** ikke benyttes af alle PLC-typer. Det betyder, at kopiering af PLC-kode eller opgradering til en større PLC giver en del arbejde og risiko for at introducere fejl.

De 3 mest brugte datatyper er **BOOL**, **INT** og **REAL**. Grunden til at **INT** benyttes mere end **WORD**, er at **INT** altid er den samme data størrelse som bit-størrelsen i en PLC og er derved en hurtig datatype. Hvis der arbejdes med en **REAL** datatype, vil PLC oprette bagvedlæggende maskin-kode, da en PLC kun kan håndtere heltal. En **REAL** er altså mere krævende, for en PLC at arbejde med.

Ulempen ved **INT** er ved brug i data kommunikation mellem 2 computere, hvor den ene computer er en PLC som anvender 16-bit og den anden en PC som anvender et 64-bit operativ system eller en lille 8-bit computer (embedded computer). Den lille embedded computer kan være en sensor, et måleinstrument, et proces analyseapparat eller andet udstyr på et anlæg. Læs mere om data kommunikation på side →.

NOTER til ovenstående tabel

1) En **REAL** indeholder max. 7 betydende cifre. Dette betyder at hvis en variabel tildeles værdien 1234.56789, vil variablen ikke kunne indeholde tallet. Værdien i variablen vil blive ændret til 1234.567 (dog vil nogle PLC vise et ciffer mere: 1234.5678)

I nogle PLC-typer hedder denne datatype en FLOAT

Da nogle computere tolker en **REAL**/**FLOAT** forskelligt kan det give udfordringer ved datakommunikation mellem computerne. For at afhjælpe dette, kan en **REAL** ”flyttes” til en **INT** eller **DINT** variabel ved at gange med 100, og når data er modtaget i den anden computer divideres med 100. På den måde kan et decimal tal med 2-cifre overføres uden problemer. Se også side →.

2) ASCII karakterer benyttes typisk, når der skal bruges tekster på f.eks. brugergrænseflader, datalogning til filer, kommunikation mellem instrumenter eller andre PLC. Samt data fra et tastatur.
Da en PLC ”kun kan” arbejde med heltal, har bogstaver og tegn fået et nummer i en ASCII tabel.
Datatypen **CHAR** er 8-bit (kan indeholde 255 forskellige tegn). En **CHAR** datatype kan typisk bruges til 1 til 5 forskellige lande sprog. **WCHAR** er 16-bit og benyttes til Unicode (ISO 10646, globale tegn). Unicode er til internationale PLC-styringer.
WCHAR bruges typisk, når samme PLC-kode bruges i flere lande med forskellige sprog i brugergrænsefladen.

3) En **STRING** består af et **ARRAY** OF **CHAR** og er normalt fastsat til 255 tegn (**CHARS**)
Se også ovenstående note 2). Samt afsnit side → og side →, **WSTRING** bruges til Unicode (ISO 10646, globale tegnsæt) og består af et **ARRAY** med **WCHAR**.

Bemærk, at nogle PLC-typer kan have max. 80 karakterer i en **STRING**, hvis **ARRAY** ikke er begrænset til f.eks. 10. Det er god programmering at begrænse længden af **ARRAY**, så der ikke bruges unødvendigt meget hukommelse.

4) **TIME**/**DATE** er internt i en PLC beregnet ud fra et heltal, der tæller tiden fra 1.1.1970 kl 00:00 og kan derfor kun konverteres til et heltal. (se dokumentation fra den enkelte PLC-producent).

En PLC henter den aktuelle tid i en elektronik komponent, der er indbygget i PLC-hardwaren. Den tid er ikke særlig nøjagtig. En nøjagtig tid skal hentes fra et atom ur. Dette gør en PC helt automatisk i dag, hvis den er koblet på internettet. En PLC kan så hente tiden fra en PC. Dette kan f.eks. gøres en gang om dagen. Det er vigtig, at alle PLC i netværket har samme tid, så alarmer og tidsstempling af loggede data (f.eks. eventlog – log af f.eks. ændringer som brugeren foretager i styringen) har samme klokkeslæt.

Når en variabel tildeles et tal, er tallet i 10 talsystemet. Hvis det tildelte tal er i 2-talsystemet (det binære), skal der anføres 2# foran og hvis tal er et HEX-tal, skal der anføres 16# direkte foran tal. F.eks. 2#101 = 5 eller 16#10 = 16.

Normalt bruges en **INT** variabel til tællere og det er vigtigt at være opmærksom på, hvor stort et tal, der kan være i en **INT**. Hvis **INT** f.eks. bruges som ”time tæller” - TACHO HOURS på en motor (en tæller, som viser det samlede antal timer en motor har kørt, og det tal benyttes for at se, hvornår der skal foretages service på motoren). Hvis motoren kører 20 timer pr. døgn og motoren har en forventet levetid på 10 år, vil den samlede tæller værdi nå op på:

```
Timer pr. døgn * dage pr. år * år = 20 * 365 * 10 =
73.000 [timer]
```

Dette bevirker, at variablen ikke kan være af datatypen **INT**, da **INT** er max 32767. Der skal benyttes en **DINT** (**D**obbelt **Int**eger) eller endnu bedre en **DWORD** datatype, der kan indeholde et større tal.

DWORD kan indeholde heltal fra 0 til 4,29 mia.

Hvis der alligevel er benyttet en **INT**, vil variablen vise: 7466, da **INT** har to ”overløb”. Der sker overløb hver gang tallet bliver over 32767 og ved overløb ”nulstilles” variablen til –32768 (som er mindste værdi for **INT**).

4.2 Afledte datatyper

Der er mulighed for at definere sammensatte og tilpassede datatyper. Det er ofte nødvendigt for at spare tid med at skrive PLC-programmet og det giver en bedre struktur på programmet.

De kaldes afledte datatyper og er omsluttet af **TYPE** og **END_TYPE**.

Der er tre afledte datatyper: **ENUM** (Enumerated data type), som opretter en liste af konstante tal. **STRUCT** (Structured data type) som kan samle forskellige variabler i en struktur, samt **ARRAY** som indeholder en række af ens variabler.

BEMÆRK
Hvis man er nybegynder inden for PLC-programmering, er det vigtigt at vide, at de afledte datatyper ikke er nødvendige for at få PLC-programmer til at virke. Det betyder, at man i første omgang, kan springe de næste

afsnit over og gå tilbage til dem, når man har mere erfaring med PLC-programmeringen.

4.3 Nummer datatype, ENUM

Nummer datatypen **ENUM** indeholder en liste af unikke konstante navne. Navne er listet i en parentes. Udtryk begynder med **TYPE** og slutter med **END_TYPE**. Brug sigende navne, der beskriver, hvad de bruges til.

For eksempel:

```
TYPE LightTYPE :
    (RED, YELLOW, GREEN);
END_TYPE
```

Datatypen **LightTYPE** i eksemplet herover, kan enten være RED (rød), YELLOW (gul) eller GREEN (grøn) og kunne f.eks. bruges til et trafik lys, en operatør signallampe (se billede) på en maskine eller som status på en ventil.

LightTYPE vil altid være én af de definerede typer: RED, YELLOW eller GREEN.

En **ENUM** *skal tildeles* en start værdi (default), ellers er det usikkert, hvad start værdien er. For eksempel, som vist herunder hvor **LightTYPE** skal have værdien RED, når der kommer strøm på PLC:

```
TYPE LightTYPE :
    (RED, YELLOW, GREEN):= RED;
END_TYPE
```

PLC-compileren sætter automatisk et fortløbende tal ind for hver tekst: RED = 0, YELLOW = 1 og GREEN = 2, da en CPU kun kan arbejde i tal. Det er deraf navnet kommer: **ENUM**, som kan oversættes til automatisk nummereret rækkefølge. Metoden benyttes, da det er nemmere at huske en tekst frem for et tal og programmøren skal ikke bruge tid på at skrive de bagvedlæggende fortløbende tal.

Der er muligt at definere en fast værdi ved hvert navn, frem for en fortløbende:

```
TYPE LightTYPE :
    (RED:= 10, YELLOW:= 20, GREEN:= 30) :=
    RED;
END_TYPE
```

Ulempen ved **ENUM** er at alle tal ligger på en fortløbende række. Hvis der tilføjes nye navne midt i rækken, forskubbes tal rækken og det kan give udfordringer, hvis **ENUM** benyttes, hvor data overføres mellem flere PLC, da begge PLC skal opdateres med ny PLC-kode på samme tid. **ENUM** har datatypen **INT**.