CODEN:LUTEDX/(TEIE-3043)/1-37/(2014)

Industrial Electrical Engineering and Automation

Autogenerering av PLC kod

- Undersökning av möjligheten att externt manipulera projektfiler

Lloyd Tran

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Autogenerering av PLC kod

- Undersökning av möjligheten att externt manipulera projektfiler

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Avdelningen för Industriell elektroteknik och automation

Examensarbete:
Lloyd Tran

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Avd. för Industriell Elektroteknik och Automation
Lunds universitet
Lund 2014

Sammanfattning

Texo Application AB är ett företag som tillverkar lyft- och godshanteringanordningar som har kontor i Älmhult. Idag använder de Siemens S7 som PLC-system, vilket företaget anser vara något krävande att arbeta med. De önskar ett nytt utvecklingsverktyg med möjligheten att kunna autogenerera PLC kod och det är på grund av detta initiativ som detta examensarbete har kommit till. Genom att generera koden kommer företaget att kunna korta ner ledtider/idrifttagningstider.

Två olika fabrikat av PLC-system har valts ut för att jämföras. Fabrikaten är Beckhoff och B&R. Målet är att kunna externt manipulera projektfiler och skapa en projektstruktur som innehåller både lokala och globala variabler. Det ska även skapa programkod som byggs ihop utifrån ett bibliotek samt att hårdvara ska kunna kopplas till programkod som t.ex. I/O. Programkod skulle kunna skapas i Strukturerad text(ST) eller Continuous Functional Chart(CFC).

Examensarbetet inleddes med introduktionskurs hos B&R och Beckhoff. Detta var ett viktig steg i arbetet då genomförandet kräver en grundkunskap för utvecklingsmjukvarorna. Identifieringen av vad som efterfrågas gällande automatisk generering har skett genom kommunikation med personal på B&R och Beckhoff.

Nyckelord: Automation studio, TwinCAT 3, generering, PLC, Test, filmanipulering, IEC 61131-3

Abstract

Texo Application AB is a company that manufactures lifting and handling equipment with offices in Älmhult. Today, they are using the Siemens S7 as their PLC, which the company find quite demanding to work with. They want a new development tools with the ability to auto generate PLC code. The code generation will enable shorter lead times.

Two different PLC brands have been selected to be compared. The chosen manufacturers are Beckhoff and B&R. The goal is to externally manipulate project files and create a project structure that contains both local and global variables. It should be possible to create program code that is assembled from a library and hardware should be connected to the program code such as I/O. Program code should be possible to be created in Structured Text (ST) or Continuous Functional Chart (CFC).

The thesis work started with an introductory course at B & R and Beckhoff. This was an important step in the process when the implementation requires a basic knowledge for developing software. The identification of the demands regarding the automatic generation has been done by communicating with personnel at B&R and Beckhoff.

Keywords: Automation studio, TwinCAT 3, generate, PLC, Test, file manipulation, IEC 61131-3

Förord

Detta examensarbete omfattar 22,5 högskolepoäng och är den avslutande delen för högskoleingenjörsutbildning på programmet Elektro- och Automationsteknik vid Lunds Tekniska Högskola/Campus Helsingborg. Arbetet har utförts under år 2014 tillsammans med Texo Application AB.

Ett speciellt tack till Olle Lindblom, min handläggare på Texo Application AB och Mats Lilja, min examinator på LTH. Jag vill även passa på att tacka Partic Thysell och Henrik Jönsson för att har varit med och hjälpt till stora delar av examensarbete. Detta examensarbete har varit väldigt lärorikt och intressant och därför vill jag tacka Anders Gydell för denna möjlighet att utföra det hos er.

PS. All kod som är i grå ruta är skriven i pseudo kod.

Lloyd Tran

Innehållsförteckning

1 Inledning

I detta kapitel ges en kort introduktion om Texo Application AB, bakgrund, syfte och avgränsning för examensarbetet. Detta examensarbete har utförts på uppdrag av Texo Application AB.

1.1 Bakgrund

Företaget Texo Application AB, som startades 2001, har huvudkontor och produktutveckling i Älmhult. Företagets fokus ligger på automationslösningar för lagerhantering. Idag använder de Siemens S7 med drives från Lenze.

Efter att ha varit i kontakt med företaget meddelades att det finns ett tänkbart examensarbete. Vid möte med Texo Application AB berättar företaget att de inte är nöjda med deras nuvarande PLC-system och att det är allmänt tungt att arbeta med. Därför vill de ha ett nytt system.

I detta projekt vill de undersöka möjligheten att kunna autogenerera PLC kod. Anledning till detta är att kunna korta ner deras ledtider/idrifttagningstider. De har valt ut två leverantörer som kan tänkas tillfredsställa deras krav, nämligen B&R och Beckhoff. De har skrivit en kravspecifikation på vad de har förväntat sig av systemen.

Båda fabrikaten utvecklar mot en IEC 61131-3 standard som är viktig för många PLC-företag. Implementerar man standarden underlättar det för kunder, då skillnaden mellan programmen blir mindre och byte mellan olika fabrikat blir lättare. IEC 61131-3 version 3.0 publicerades år 2013 med följande programmeringsspråk för PLC:

- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Structured Text (ST)
- Instruction List (IL)
- Sequential Function Chart (SFC)

[1]

Både Beckhoff och B&R erbjuder en ytterligare utökning av IEC-standarden med följande programmeringsspråk:

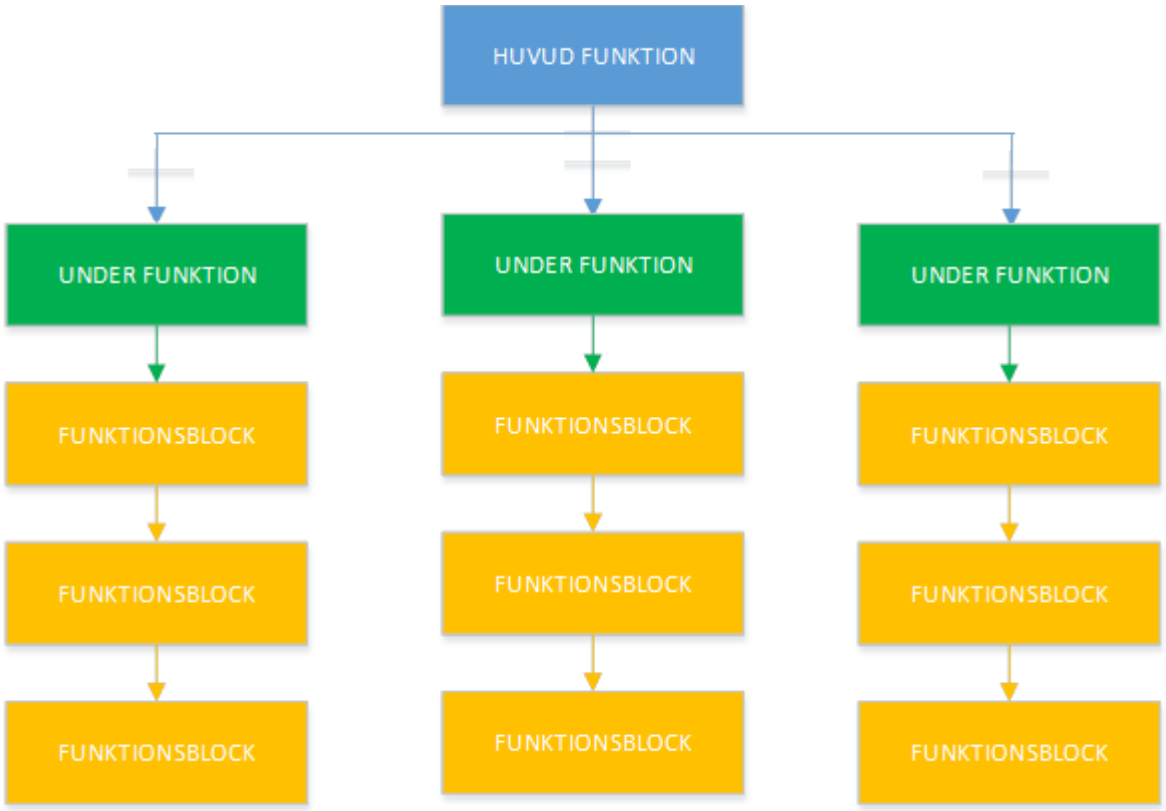
- Continuous Function Chart (CFC)
- C/C++
- MatLAB/Simulink

1.2 Syfte och Mål

Syftet är att undersöka om det finns möjlighet för att automatiskt skapa en projektstruktur utifrån ett färdigt bibliotek samt presentera ett koncept på hur programmet kan se ut. Kodningen skall vara gjord i Strukturerad Text och CFC för presentation för att kunder ska kunna se hur det är kopplat.

Vad är bakgrunden till generering av kod?

- Vinna mer tid
- Utnyttja tiden till effektivare arbete
- Mer ekonomisk för Texo Application AB
- Billigare för kunder
- Sätta igång projekt snabbare



Figur 1.1 *En huvudfunktion som styr alla underfunktioner, representerad i CFC*

Systemet skall vara uppbyggt som ett hierarkiskt system som figur 1.1 visar, där ett gult eller grönt block kan vara en typ av maskin t.ex. en rullband. Målet är att kunna bygga upp detta system med hjälp av autogenereringen. För en bättre förståelse för detta finns det beskrivet i kapitel 4.1. En jämförelse mellan B&R och Beckhoffs system görs i slutet av examensarbetet.

1.3 Problemformulering

- Går det att skapa en färdig projektstruktur genom extern manipulering av filer?
- Kan projektet skapas i Strukturerad Text(ST) och Continuous Functional Chart(CFC)?
- Kan dessa projekt kopplas till hårdvaror som exempel I/O eller Frekvensomformare?
- Hur genereras kod?
- Jämförelse av fabrikaten.

1.4 Avgränsningar

Koden som idag finns i Texo Application AB Siemens S7 används inte i detta projekt eftersom tiden inte räcker till, utan ett eget funktionsbibliotek skapas i PLC:n och används för att bygga upp projektet. Endast mjukvaran från fabrikaten Beckhoff och B&R kommer att bli utvärderad eftersom tiden inte räcker till för att testa hårdvaror.

2 Verktyg

Här beskrivs det kortfattat om program som detta examensarbete omfattar.

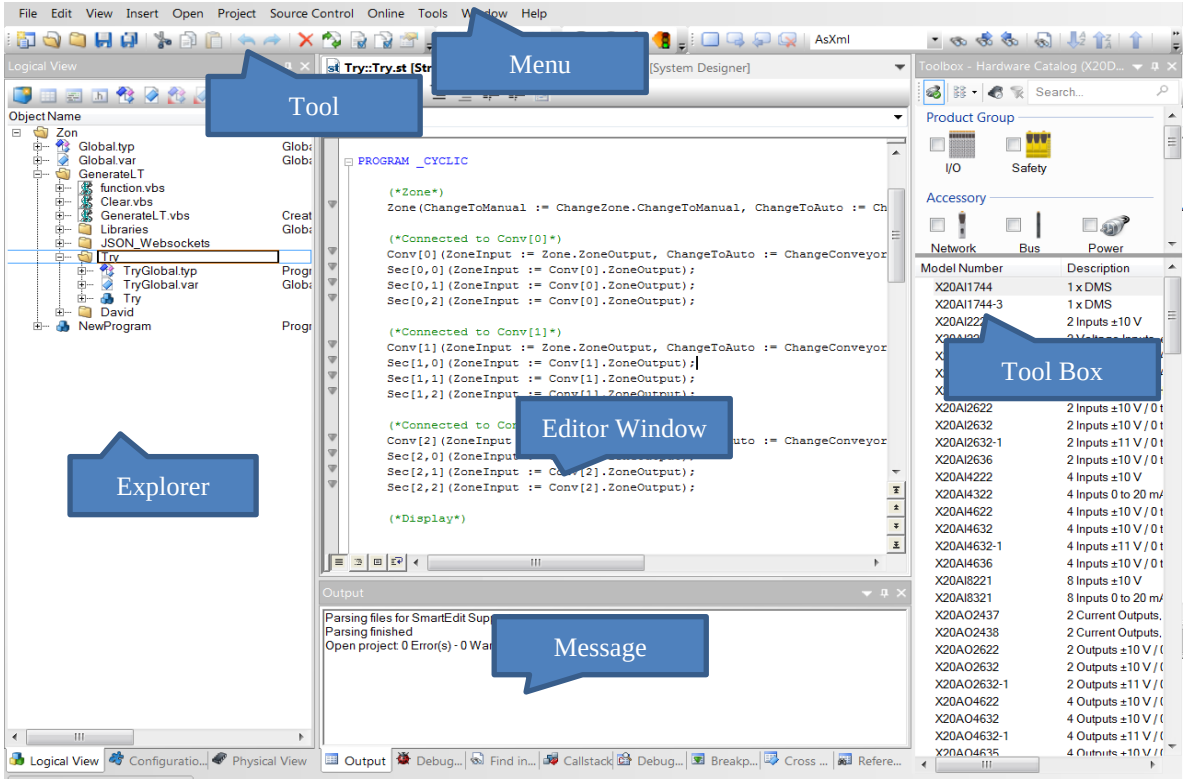
2.1 Mjukvarorprogram

2.1.1 Notepad++

Notepad++ är ett gratis textredigeringsprogram för Microsoft Windows. Jämfört med den vanliga Windows notepad så har den stöd för flikar och gör det möjligt att ha flera filer öppna samtidigt. Notepad++ är utvecklad i C++ och har funktioner som underlättar för programmerare att skriva kod i olika programspråk, [2].

2.1.2 Automation studio 4

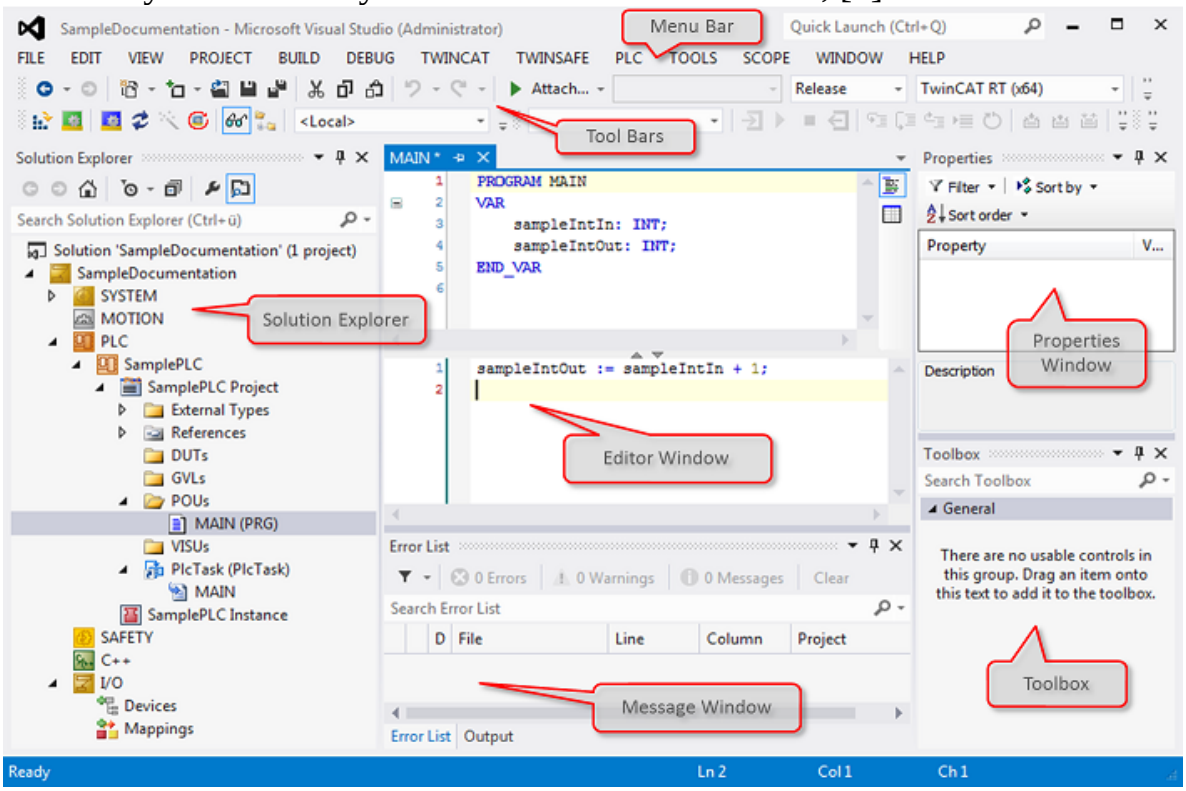
Automation Studio är en PLC-mjukvara för B&R. De använder Powerlink som protokoll för kommunikation mellan PLC och dator, vilket gör att man kan testa program på en dator. Utvecklingsverktyget stödjer IEC 61131-3, ANSI C/C++ och MATLAB/Simulink med objekt-orienterad programmering. Automation runtime körs cykliskt, vilket innebär när en cykel körs anropas programmet och när programmet är färdigt väntar den tills nästa cykel ska köras, cykeltiden kan ändras till olika värden, [3].



Figur 2.1 *Arbetsyta i Automation Studio*

2.1.3 TwinCAT3

TwinCAT3 är en PLC-mjukvara för Beckhoff som använder sig av Visual Studio shell som påminner mycket om Microsoft Visual Studio i utseende, se figur 2.2. Kommunikationen mellan en dator och PLC sker med EtherCAT som protokoll och gör det möjligt att testa kod på en dator. Om man använder en renodlad TwinCat 3 finns det IEC 61131-3 för programmering med objektorientering, skulle man installera Microsoft Visual Studio bli C/C++ och MATLAB/Simulink tillgängligt för användning i TwinCat 3. TwinCAT runtime systemet körs cykliskt som Automation studio, [4].



Figur 2.2 *Arbetsyta i TwinCAT3*

2.2 Programspråk

2.2.1 Strukturerad Text (ST)

Strukturerad text är ett programmeringsspråk som är en standard i IEC 61131-3 och är ett textbaserat språk. Språket liknar PASCAL.

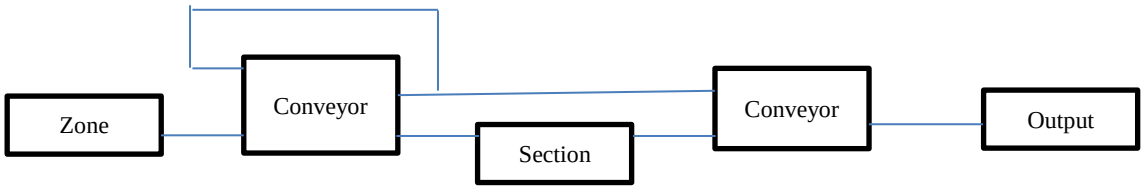
Exempel kod:

```
Case Number OF
        0: Counter = Counter + 1;
        1: Counter = Counter + 10;
        2: Counter = Counter + 100;
END CASE
```

Kod skriven i Strukturerad Text

2.2.2 Continuous Function Chart (CFC)

CFC är ett grafiskt programmeringsspråk som inte är en standard i IEC 61131-3. Det är block som kan flyttas fritt på arbetsytan och det är möjligt att göra återkopplingar med linjer.
Exempel kod:



Figur 2.3 *Kod skriven i CFC*

2.2.3 VBScript

Microsoft Visual Basic Scripting Edition är ett språk utvecklat av Microsoft och har rötter från Visual Basic. Det används till många olika miljöer, bland annat Internet Explorer eller filhantering för att skapa/ta bort filer. VBScriptet har många egenskaper gemensamma med andra programmeringsspråk men en tydlig skillnad är att om man deklarerar variabler skrivs det ”dim minVariabel”. Sedan kan minVariabel användas som ett BOOL uttryck eller string till och med som integer. Som utveklare kan man använda VBScriptet kostnadsfritt för eget bruk, [5].

2.2.4 JavaScript

JavaScript är ett populärt språk för HTML och vanligvis inbäddas JavaScript i, eller inkluderas från HTML-sidor. JavaScript är snarlikt Java och har möjlighet att använda samma programstrukturer som t.ex. for, if och while. I scriptet så deklareras variablerna med kommando ”var” och samma användning som i VBScriptet. De flesta webläsare är anpassade för att kunna exekvera JavaScript, [6].

2.3 Övrigt

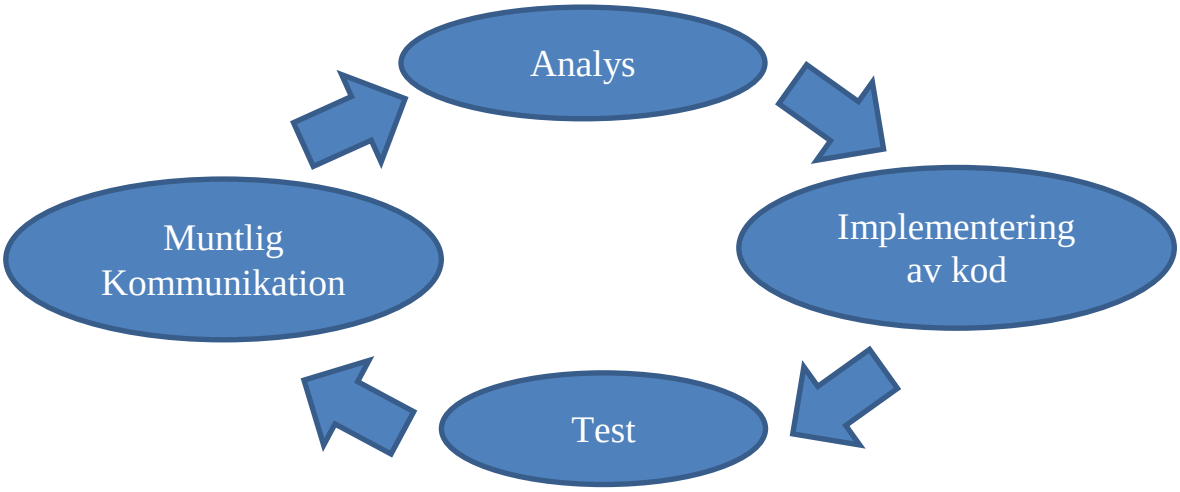
2.3.1 TwinCAT Automation interface

Automation Interface gör det möjligt att automatiskt skapa och manipulera TwinCAT-projekt via programmering eller skriptkod så länge programmeringsspråket kan lägga till COM-objekt. De program som har stöd för det är t.ex C++ eller .Net, det som är rekommenderat är C# i .Net, [7].

3 Metod

I detta kapitel beskrivs val av metod och informationsinsamling för examensarbetet.

Målet med arbetet har varit att ta fram en lösning till autogenerering av kod och koppla dessa projekt till hårdvaror. Arbetssättet illustreras i Fig. 3.1



Figur 3.1 *Arbetsmetoden*

Först identifieras behovet, vilket är den absolut viktigaste delen genom intervjuer. Sedan analyseras det och en kravspecifikation formuleras för vad som ska uppfyllas för att möta behovet för Texo Application AB, kravet delas upp i mindre storlek för bearbetning. Implementering för en eller flera delmål sker och programmet testas efteråt. Därefter återgår man till muntlig kommunikation med företaget för att säkerställa att man har uppnått delmål. Denna metod itereras om och om igen för att kunna besvara de flesta frågorna.

Examensarbetet inleds med en introduktionskurs hos Beckhoff och B&R för att ge en bra förståelse för deras mjukvara och för att kunna komma igång snabbare. Där går man igenom grunderna i deras mjukvara som t.ex. deklarering av variabler, koppling till I/O och mycket mer.

4 Analys och genomförande

4.1 Befintligt system

I systemet Siemens S7 är mjukvaran och hårdvaran rätt hårt knutna då man måste koppla variabler till en adress och sedan använda adressen till att koppla till hårdvara som input eller output. Istället vill man kunna koppla variabler direkt till en hårdvara, vilket gör det lättare att förstå en hårdvaras koppling. Det är bland annat därför B&R och Beckhoff är utvalda som tänkbara fabrikat till Texo Application AB.

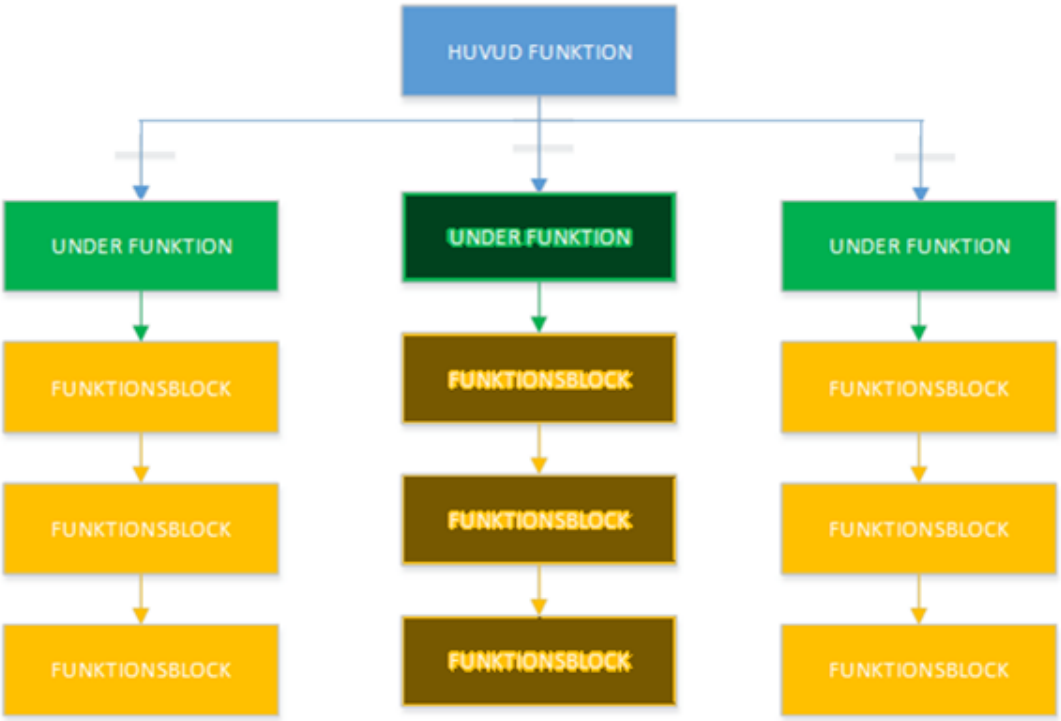
Figur 4.1

Figur 4.1 illustrerar på hur Texo Application AB arbetar idag. De identifierar en kunds behov och sedan kan de påbörja deras arbetet. De programmerar, konfigurerar upp hårdvara och kopplar sedan ihop hårdvaran med mjukvaran. De vill nu kunna generera så mycket som möjligt så att de vinner mer tid och utnyttjar den tiden bättre, exempelvis till att utveckla mer produkter.

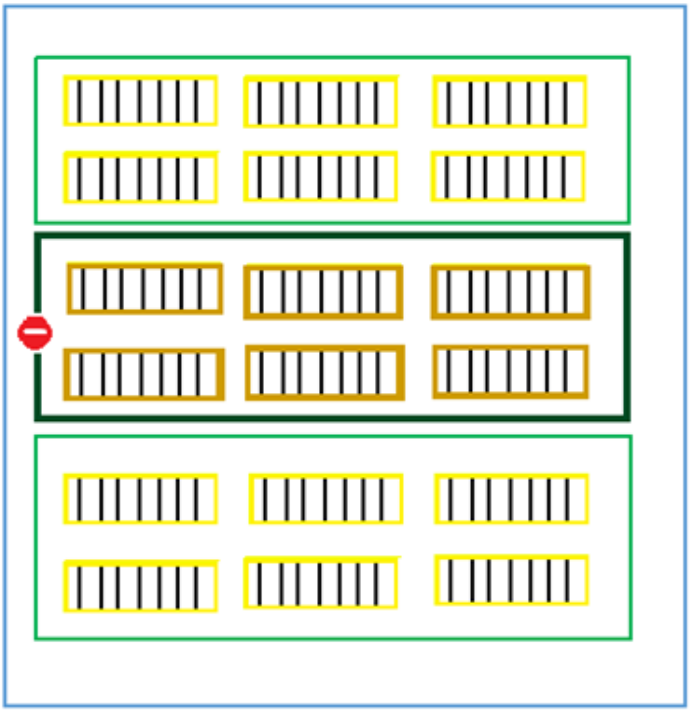
För att kunna generera kod så behöver man ett färdigt funktionsblock som man bygger ifrån ett bibliotek. Eftersom programkod från siemens s7 inte ska användas till detta projekt så ska ett eget bibliotek skapas. Det ska ha funktioner som visas i figur 1.1.

I figur 4.3 så kan vi tänka oss att det gula ska vara rullband och det gröna är ett delområde och det blåa är hela området och det kan representeras i CFC kod i figur 4.2 då ”huvudfunktionen” motsvarar hela området och ”underfunktionen” är ett delområde. Om en ”underfunktion” ställs om till manuellt läge ska alla funktioner som ligger under denna nivå också bli manuella. Detta illustreras i figur 4.2 då den mörkare delen befinner sig i manuellt läge. Det går också ställa om den blåa delen och allt som befinner sig under blir manuellt och det går också att ställa om den gula delen, men Texo Application AB tycker att det räcker med att man kan operera på den blåa och gröna delen.

Idag kan man enbart ställa om så att alla maskiner körs på auto eller manuellt läge med hjälp av en inställning i ett program d.v.s de kan bara styra den ”huvudfunktionen” och Texo Application AB vill ha en ny funktion där de kan ställa om ett grönt område till ett önskat läge utan att påverka de andra gröna områdena. Man vill gärna kunna styra det längre ner eftersom om läget på huvudfunktionen är manuellt så står alla maskiner stilla, vilket gör att maskinerna inte arbetar effektivt. Nu vill företaget utöka det så att man kan ställa om ett delområde till manuell så att man till exempel kan göra service på den delen under tiden som resten av maskinen arbetar på som vanligt.



Figur 4.2 *Underfunktion befinner sig i manuellt läge, representerad i CFC*



Figur 4.3 *Blå = Hela området, grön = delområde, gul = rullband. En överblick på ett system.*

4.2 Implementering

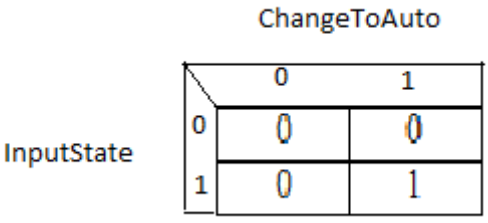
För att skapa projektstrukturen så kan varje funktion i figur 4.2 representeras av ett funktionsblock och kommunikationen mellan blocken kan ske via BOOL-variabler. För att skapa en ”huvudfunktion” så behöver den kunna ändra sitt eget läge till auto eller manuell. Detta kan man göra med följande ST-kod där FALSE eller 0 motsvarar manuell och TRUE eller 1 motsvarar auto:

```
Auto := ChangeToAuto;
Manual := NOT ChangeToAuto;
```

Variablerna Auto och Manual är output med typen BOOL och ChangeToAuto är input med typen BOOL. Sedan kan man göra det till ett funktionsblock och skapa en instans av det så att det ser ut som i figuren 4.4. Den vänstra sidan om blocken är inputs och högra sidan är output.

Figur 4.4 *CFC block*

För att bygga vidare med ”underfunktion” läggs det till en variabel InputState som ska hämta ett BOOL värde från ett ”högre” funktionsblock. När värdet på InputState har hämtats in behandlas det värdet. Om ”huvudfunktion” är på Auto då finns möjlighet för ”underfunktion” att ställa sig på Auto eller manuell men om ”huvudfunktion” befinner sig i manuellt läge kan inte ”underfunktion” ställa sig på auto läge utan bara manuell. Detta kan beskrivas med karnaughdiagram.

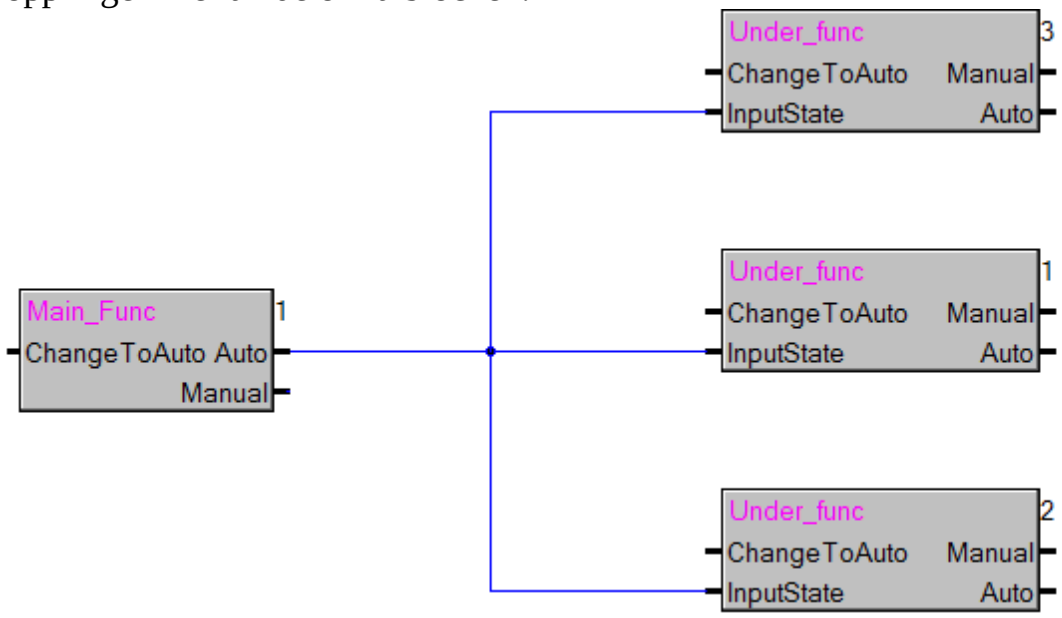


Figur 4.5 *karnaughdiagram 0 = Manuell, 1 = Auto*

Detta blev implementeringen i Strukturerad Text:

```
Auto := ChangeToAuto AND InputState;
Manual := NOT Auto;
```

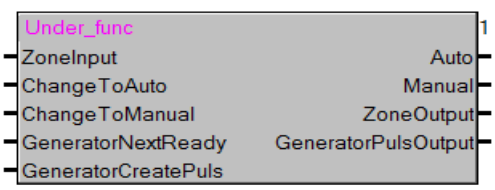
Koden läggs in som ett bibliotek efteråt och kunde användas till CFC kod vilket är en mer visuell variant av programmering. Figuren 4.6 visar kopplingen mellan de olika blocken.

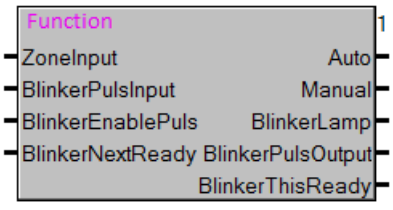


Figur 4.6 *CFC blockschema*

Nästföljande funktionsblock fungerar då som en ”underfunktion” som också har en InputState och utför samma kod för att ändra sitt tillstånd. När detta byggs upp i CFC blir strukturen som i Figur 1.1. Programstrukturen kan skrivas i ST och koden visas i kapitel 4.4 (Autogenerering).

Denna metod som har använts har modifierats något efter en diskussion med personalen i B&R. Just nu sker kommunikationen mellan blocken med en BOOL-variabel men istället kan en pekare användas som ger mer fördelar, eftersom om man använder BOOL kan man skicka eller ta emot en variabel men med en pekare kan man skicka eller ta emot fler variabler. Kopplingen och strukturen är fortfarande den samma men kommunikationen sker med pekare istället. Sedan har en utökning av funktionsblocken gjorts för att ge den fler funktioner som ska användas för koppling till hårdvara.



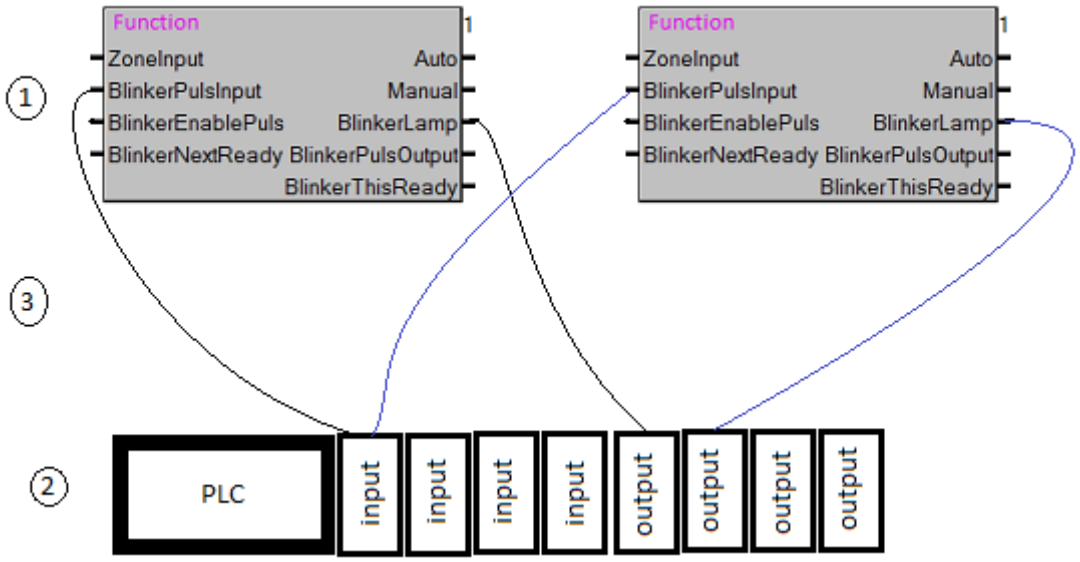


Figur 4.7 *“Underfunktion” CFC block* *”funktion” CFC block*

Eftersom både B&R och Beckhoff arbetar med utveckling mot samma IEC-standard blir detta examensarbete något lättare. Det går att föra över projekt från en PLC tillverkare till en annan, med några små förändringar. Detta är en principkod som används för detta projekt för att kunna komma vidare till hårdvarukonfigurering och sedan kunna koppla ihop det med denna kodimplementering.

4.3 Hårdvara

För att skapa hårdvaran så har BlinkerPulsInput som är en input respektive BlinkerLamp som är en output valts ut för att användas till hårdvaran, det vill säga för varje block som instansierats ska de utvalda variablerna kopplas mot en input eller output slice. Till detta projekt används input- och output-moduler som har stöd för upp till 8 kanaler och det innebär att max 8 variabler som kan kopplas mot en modul. (Se Figur 4.8.)



Figur 4.8 *En illustrativ bild av koppling mellan variabler och hårdvara*

Figur 4.7 beskriver metoden för att koppla ihop projektet med hårdvaran. Först så skapas en projektstruktur och sedan sätter man upp hårdvara som ska anpassas efter storleken på projektstrukturen. Tredje steget är att länka ihop program med hårdvara genom att koppla variabelnamn mot en input eller output slice.

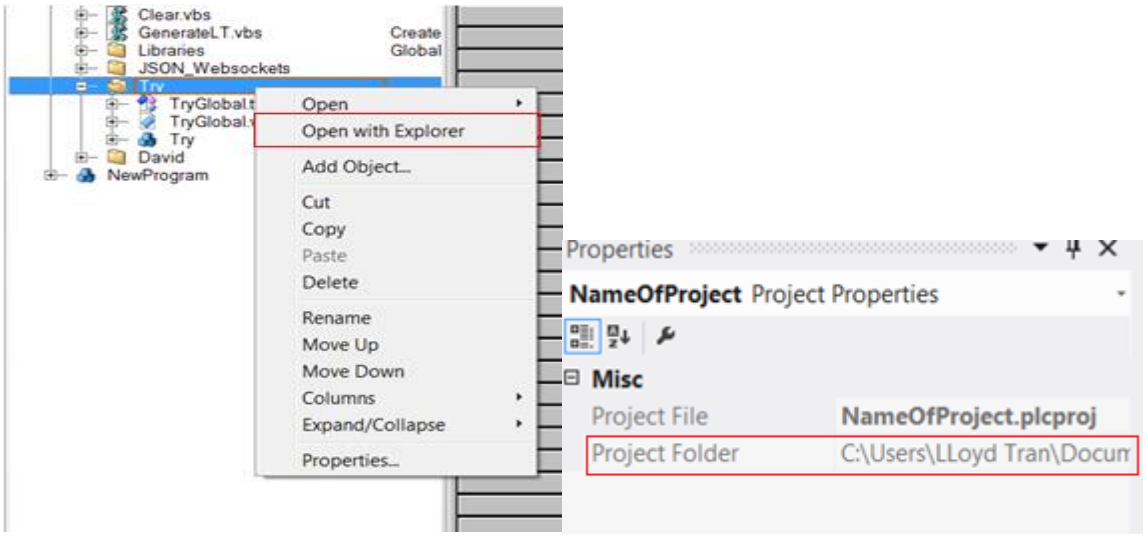
4.4 Autogenerering

4.4.1 Mjukvaran

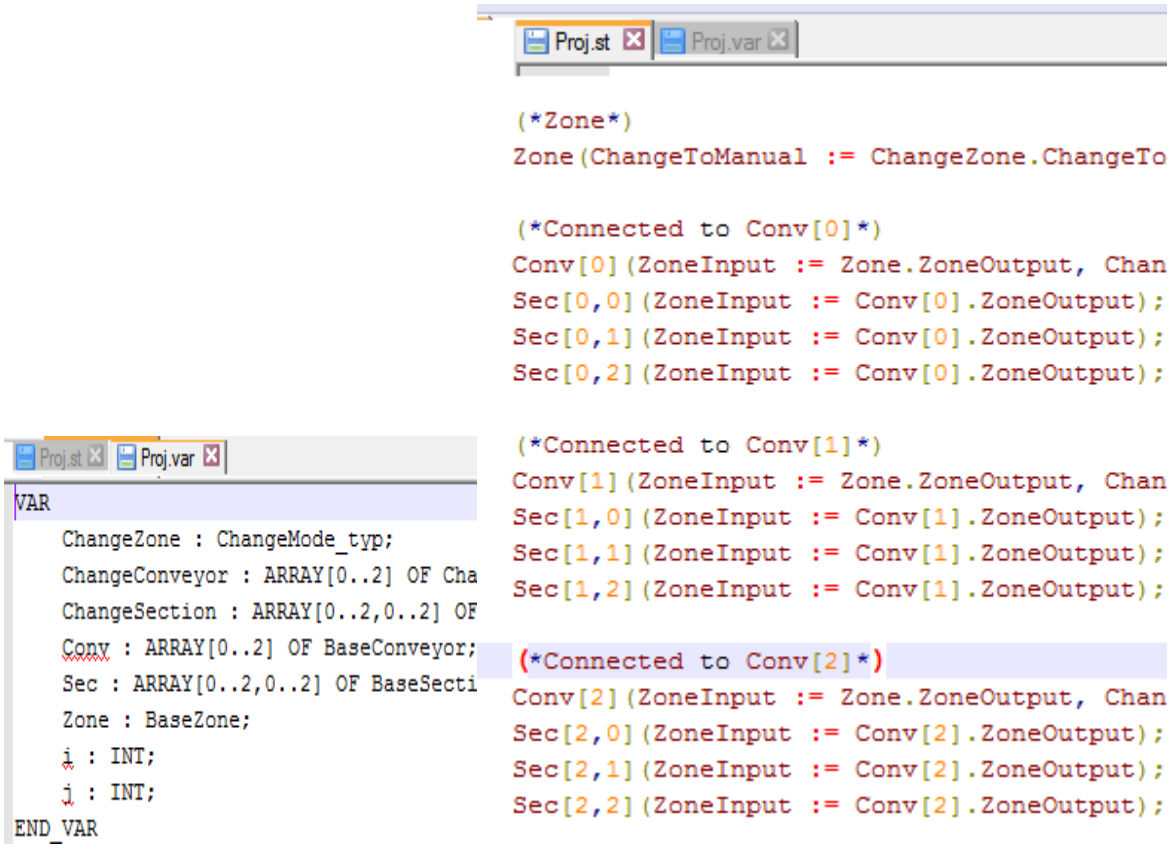
För denna autogenerering är tankegången att använda ett programspråk som java eller liknande som opererar utanför både Automation Studio och TwinCAT3. Men för att kunna göra det behöver man kunna öppna filer utanför utvecklingsprogrammet som är nödvändiga för projektet och det bör vara läsbart för att kunna redigera i filerna.

För Automation Studio används Visual Basic Scripting istället för java. Visual Basic Scripting används eftersom det redan använts till samma idé i detta examensarbete. Det som behöver ändras är projektkoden, skapa hårdvaran och koppla ihop projekten. För TwinCAT3 används Automation Interface som är ett gränssnitt som är byggt för att manipulera allt i TwinCAT3 som till exempel att skapa projekt eller lägga till hårdvara m.m.

För att kunna manipulera utanför behöver man leta upp var projektet finns på hårddisken. I Automation Studio kan man högerklicka ”explorer window” och sedan på ”open with explorer” så att man kommer fram till rätt destination medan i TwinCAT3 står sökvägen i egenskaper hos projektet som man kopierar. När man är framme i projektmappen ska man leta upp rätt fil, för Automation Studio är det variabelfil, strukturerad text fil och continuous function chart fil som ska letas upp och de har ändelserna .var, .st respektive .cfc. I TwinCAT3 räcker det att leta upp .TcPOU –fil som innehåller både deklarationstext med strukturerad text eller deklarationstext med CFC. Filerna öppnas med Notepad++ vilket gör att man kan se filerna i rå text.



Figur 4.9 (a) *Automation studio explorer vy* (b)*TwinCat3 egenskaper vy*

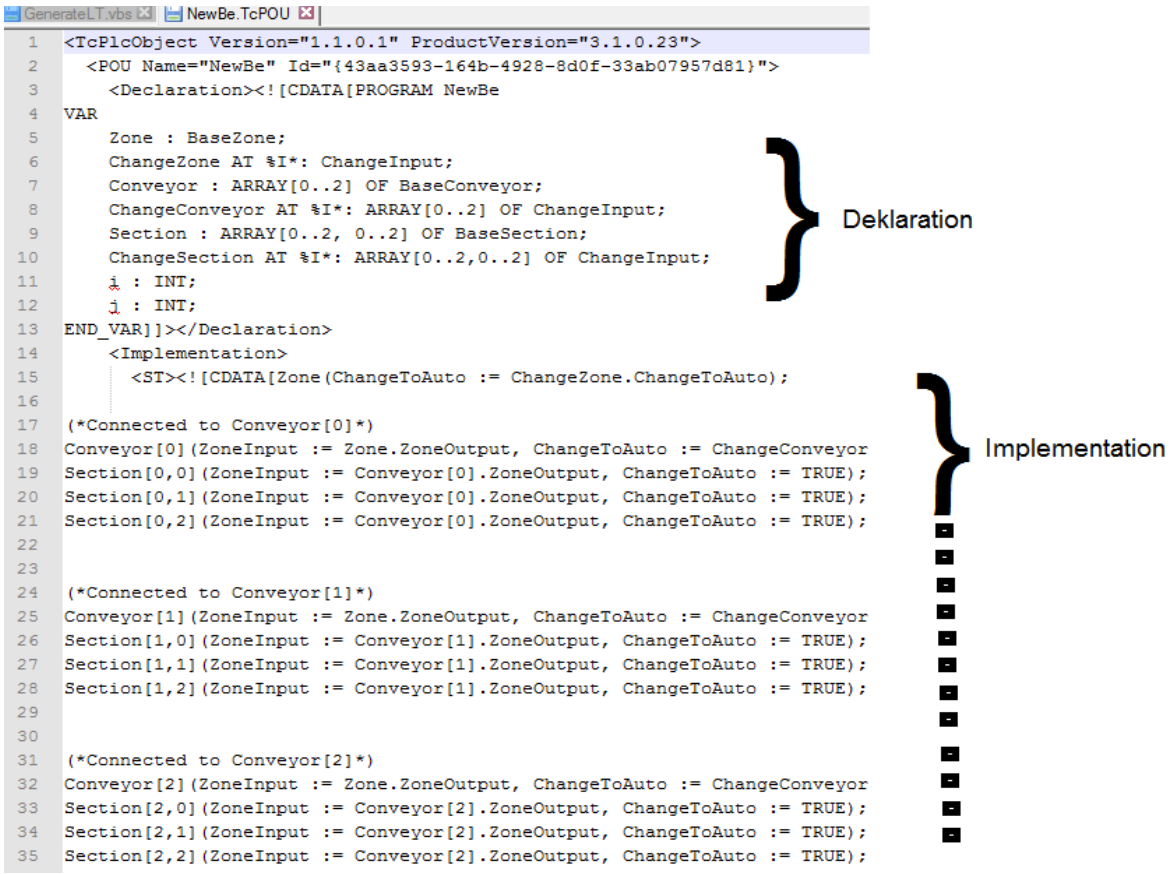


Figur 4.10 *Filer från Automation Studio öppnade i notepad++*

Genom att öppna filerna som är skapade i Automation Studio kan man se att .var-fil och .st-fil är i klartext och är precis samma kod som är skriven i Automation Studio, vilket underlättar användningen av Visual Basic Scripting, se figur 4.10 (vänster är deklaration och höger är implementation).

```
Set fso = CreateObject("Scripting.FileSystemObject")
set b = fso.CreateTextFile ("skriv in sökvägen t.ex. C:\project\Proj.st")
'b.WriteLine ("Lägg till kod t.ex.")
b.WriteLine ("Zone(ChangeToManual := ChangeZone.ChangeToManual);")
'b.WriteLine ("lägg till mer kod")
b.Close();
```

Koden tilldelar fso till FileSystemObject vilket gör att en variabel kan användas vid filhantering för att skapa en fil som heter Proj.st och lägger in en kodrad med texten ”`Zone(ChangeToManual := ChangeZone.ChangeToManual);`” och sedan så stänger den filen med b.Close. Sedan kan man gå vidare till .var-fil och göra liknande men istället lägga till deklarationstext.

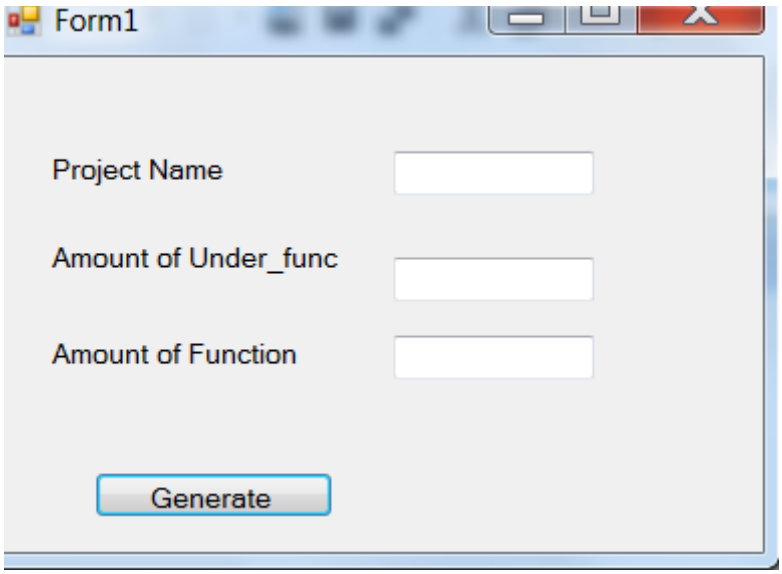


Figur 4.11 *Öppen TwinCAT3 fil i Notepad++*

I TwinCAT3 används det Automation Interface med Microsoft Visual Studio C# programmering. Öppnar man program filen .TcPOU kan man se i figur 4.11 att det innehåller både deklarationstext och implementationstext i samma fil. För att komma år deklarationstext eller implementationstext i Automation Interface behöver man skriva in sökvägen till projektkoden med kommandot ”LookupTreeItem” och sedan kan man båda läsa ut och skriva till deklareringar och implementering. Man kan läsa ut befintligt deklarering eller implementering till t.ex. en string för att vidare bearbeta koden. För att skriva tillbaka till deklarering eller implementering kan man se i koden och då skriver den över den gamla koden.

```
PLC = sysManager.LookupTreeItem("sökvägen till PLC");
//Läsrättighet
string declText = PLC.DeclarationText;
string implText = PLC.ImplementationText;
//Skrivrättighet
PLC.DeclarationText = “Mina deklareringar”;
PLC.ImplementationText = “Min implementation”;
```

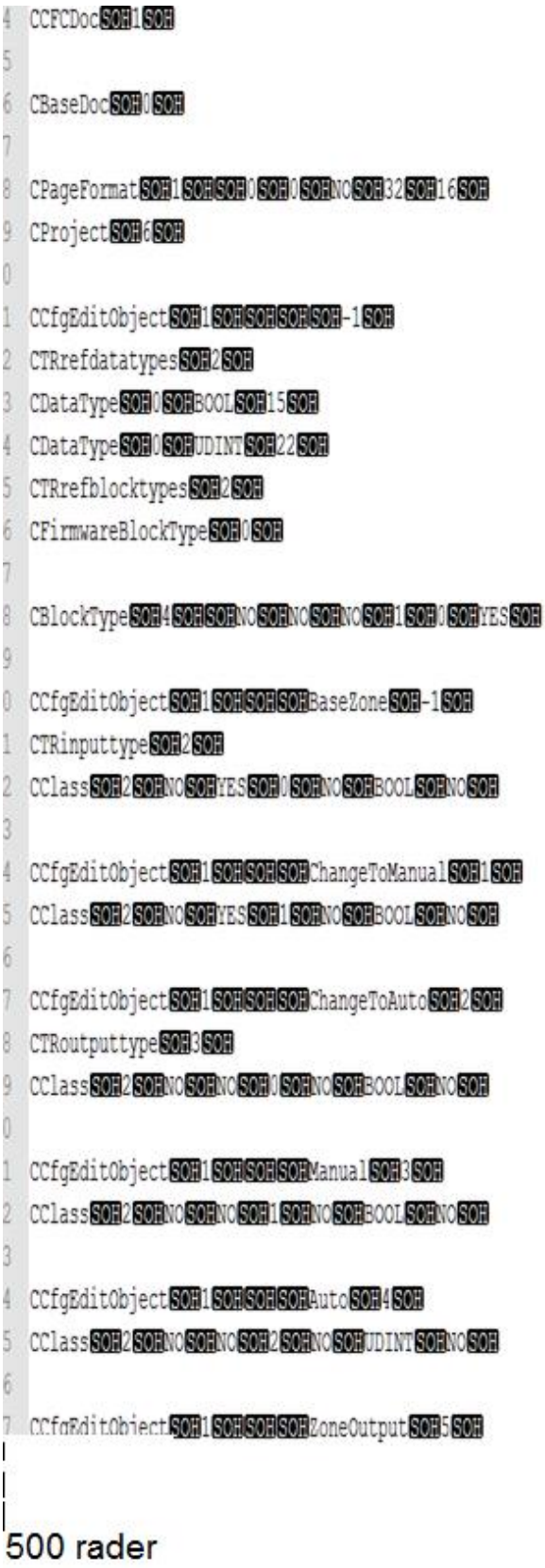
När det lyckades att få programmet att skapa mycket av projektet automatiskt så insåg man att det skapar en exakt identisk struktur varje gång utan att en användare kan välja storlek på strukturen. Då utvecklades det vidare så att användaren kan välja antalet ”underfunktioner” och antalet ”funktioner” och på så sätt variera storleken på projekten.



Figur 4.12 *Fråga användaren om information*

Under arbetets gång upptäcktes det att denna grafiska del CFC är svårt att tolka utanför utvecklingsmjukvaran. För att skapa programstruktur som figur 4.6 utanför både Automation Studio och TwinCAT3 kan man se CFC fil öppnad i Notepad++ i figur 4.13. I figuren 4.13 innehåller det en liten del av vad som faktiskt finns i filen för CFC, det fortsätter i ungefär 500 rader. CFC-programmering verka inte vara avsedd för att arbeta med utanför utvecklingsmjukvaran Automation Studio eller TwinCAT3 eftersom det är grafiskt. Det är inte omöjligt att arbeta utanför men att göra det kan vara svårt.

Texo Application AB vill ha funktionen för att kunna presentera den grafiska bilden för deras kunder som är intresserade. Eftersom det inte går att skapa CFC utifrån så kommer förslag att ersätta det med något annat. Det man kan göra är att läsa upp variabler som är intressanta och se och visa upp det på ett program eller en websida. I nästa kapitel beskrivs ersättaren till CFC.



Automation Studio TwinCAT3

Figur 4.13 *CFC fil öppnad i Notepad++*

4.4.2 Hårdvaran

För att generera upp hårdvaran krävs det att man hittar hårdvaru-filen som har filnamnet Hardware.hw. När man öppnat filen kan man se att den är uppbyggd i XML och har en struktur som går att förstå. För att lägga till en modul skriver man ett komplett element med starttagg och sluttagg, och mellan taggarna skriver man in modulens namn och vilken typ modulen är som t.ex. input/output modul. En koppling till någon punkt ska också göras och det görs med hjälp av ”<Connection>”.

```
</Module>
<Module Name="Try_X20DI8371_CI0" Type="X20DI8371" Version="1.0.1.0">
  <Connection Connector="SS1" TargetModule="Try_X20TB12_CI0" TargetConnector="SS" />
  <Connection Connector="SL" TargetModule="Try_X20BM11_CI0" TargetConnector="SL1" />
  <Parameter ID="Supervision" Value="off" />
</Module>
```

```
For i = 0 To antalModul

        b.WriteLine("</Module>")
        b.WriteLine("<Module Name=”Modul Namn” Type="Input/Output" />")
        b.WriteLine("<Connection TargetConnector="" />")
        b.WriteLine("</Module>")
Next
```

När hårdvaran är klar så behövs den kopplas till projektet. För att göra det finns det en fil som har ändelse .iom som används för att länka ihop både mjukvaran och hårdvaran. Varje rad i IOmap.iom är en länkning mellan en variabel och en digital input eller output.

```
    ::Try:ChangeSection[2,1].ManualEnable AT %IX."Try_X20DI8371_CI2".DigitalInput02;
    ::Try:ChangeSection[2,2].ManualEnable AT %IX."Try_X20DI8371_CI2".DigitalInput03;
    ::Try:Sec[0,0].BlinkerLamp AT %QX."Try_X20DO8322_CO0".DigitalOutput01;
    ::Try:Sec[0,1].BlinkerLamp AT %QX."Try_X20DO8322_CO0".DigitalOutput02;
```

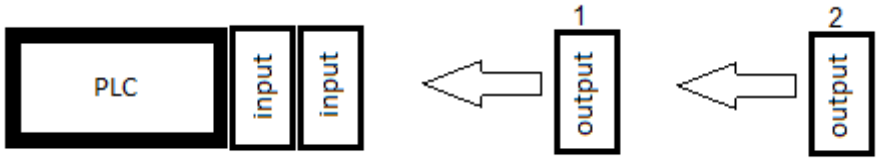
```
For i = 0 To antalInputOutput

        b.WriteLine("minInputVar AT %IX."modul_namn_Input".Input;")
        b.WriteLine("minOutputVar AT %QX."modul_namn_Output".Output1;”)

Next
```

I TwinCAT3 är det betydligt svårare att hitta hårdvaru-filen men det behöver man inte heller göra. Man har Automation Interface till sin hjälp och kan använda LookupTreeItem() för att hitta hårdvarunoden. För att hitta

hårdvarunoden har den en beteckning ”TIID”, då blir kommandot LookupTreeItem(”TIID”). När man har hittat noden är det fritt fram att lägga till hårdvaror. Med funktionen CreateChild() lägger den till sina önskade hårdvaror och hårdvaran hamnar efter varandra automatisk i den ordningen som koden exekverats. Det vill säga om man kör en for-sats med CreateChild(”output”) så lägger den till IO-slice efter varandra, se figuren 4.14.



Figur 4.14 *Varje gång CreateChild() exekverats läggs det nya moduler efter*

För att länka ihop variabler med hårdvaran används funktionen LinkVariables() och inom parentesen skriver man vilken variabel som ska länkas till vilken input- eller output-kanal.

```
//Hitta Hårdvaor noden
ITcSmTreeItem PLC = sysManager.LookupTreeItem("TIID");

// Skapa hårdvara
for (int i = 0; i <= antalInputOutput; i++)
{
  PLC.CreateChild("Module_name_Input", "Hardware_Input");
  PLC.CreateChild("Module_name_Output", "Hardware_Output");
}

//länka hårdvara
for (int i = 0; i <= antalLänkning; i++)
{
  sysManager.LinkVariables("myVariable", "Module_name_Input^Input");
  sysManager.LinkVariables("myVariable", "Module_name_Output^Output");
}
```

4.5 Ersättare till CFC

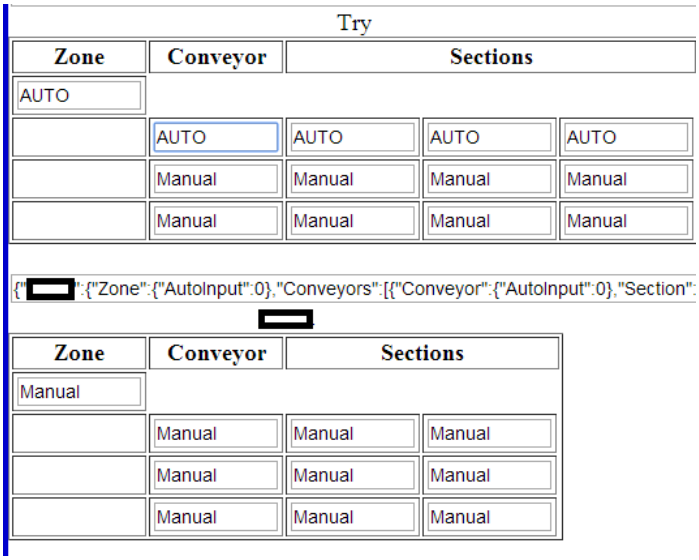
4.5.1 HTML

Eftersom CFC är svårt att generera så är ett förslag att ersätta CFC genom att använda en websida som kan hämta variabler från PLC och göra en tänkbar presentation på hemsidan. Informationen som kan vara intressant är att kolla om en maskin är på manuellt läge eller auto läge. Eftersom man kan återskapa koden i Strukturerad Text så kan man ta kodens variabler och skicka upp det till en hemsida.

För att skapa en struktur som figur 1.1 kan man använda sig av en tabell för att visa information. Varje cell i tabellen kan representeras som ett block och i cellen innehåller det en eller flera variabler som är intressant, men för detta projekt blir det bara en variabel som varje cell innehåller.

För att göra detta möjligt så behöver PLC:n skapa en websocket, vilket gör att man kan hämta variabler från det med funktionen JSON.Parse i JavaScript. Sedan läggs variablerna i en tabell. En Update funktion skapades och funktionen anropar JSON.Parse efter en viss tids för att kunna kontrollerar så att rätt värde från PLC visas på HTML sidan. I tabellen kan man även få det att skifta mellan auto och manuellt genom att klicka på det. Då har tabellen en onClick funktion och när den anropas så kan den skicka ner nytt värde till PLC med websocket, när funktionen UpDate bli anropad så uppdateras hela tabellen.

Varje PLC får en egen websida för att visa sin status. Då kan man komma åt hemsidan med en surfplatta eller en smartphone om man är uppkopplad i det lokala nätverket och knappa in IP-adressen i adressfältet.



Denna websida gjordes och kopplades mot Automation Studio men TwinCAT 3 har också stöd för denna metod.

Figur 4.15 *Hemsidan visar två skapade projekt i en PLC*

4.5.2 ADS

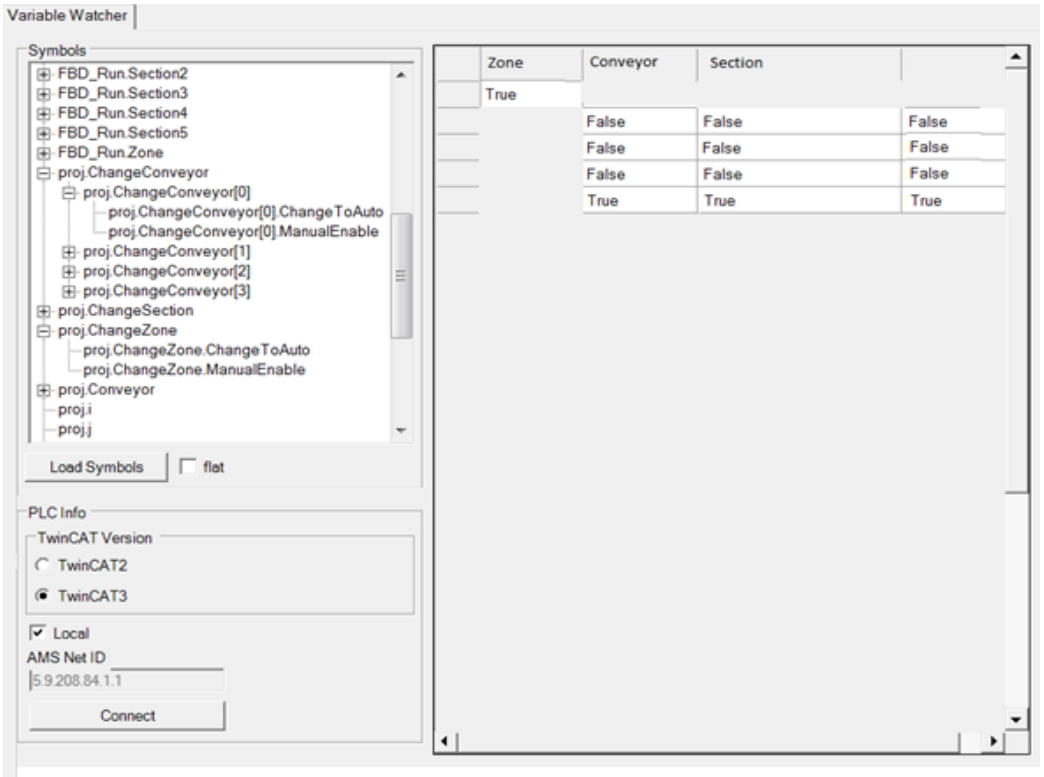
En annan metod är att skapa ett program som kan hämta värden från en PLC. ADS står för Automation Device Specification och är ett protokoll i transportskiktet (transport layer) i TwinCAT system. Det är en teknik som finns hos Beckhoff.

Det behövs koppla sig mot PLC och sedan skriver man in adress till den PLC man är intresserad av. Med detta program så itererar den igenom alla variabler i PLC:n och lägger i en ”Symbols” lista och sedan läggs det in i en datagridview så att det ser ut som en tabell, se figur 4.16.

Detta har samma funktioner som HTML-lösningen:

- Hämta variabler från PLC
- Skapa en tabellpresentation
- Uppdaterar tabellen efter en visst tidsintervall
- Skicka ner värde till PLC

Detta kan utvecklas vidare till att en PLC ska kommunicera med andra PLC:er. Den har även stöd för den äldre versionen av TwinCAT 2.



Figur 4.16 *Program som kan läsa variabler från PLC*

Detta gjordes och kopplades mot TwinCAT 3 men B&R har också stöd för denna metod med ett annat protokoll.

5 Resultat

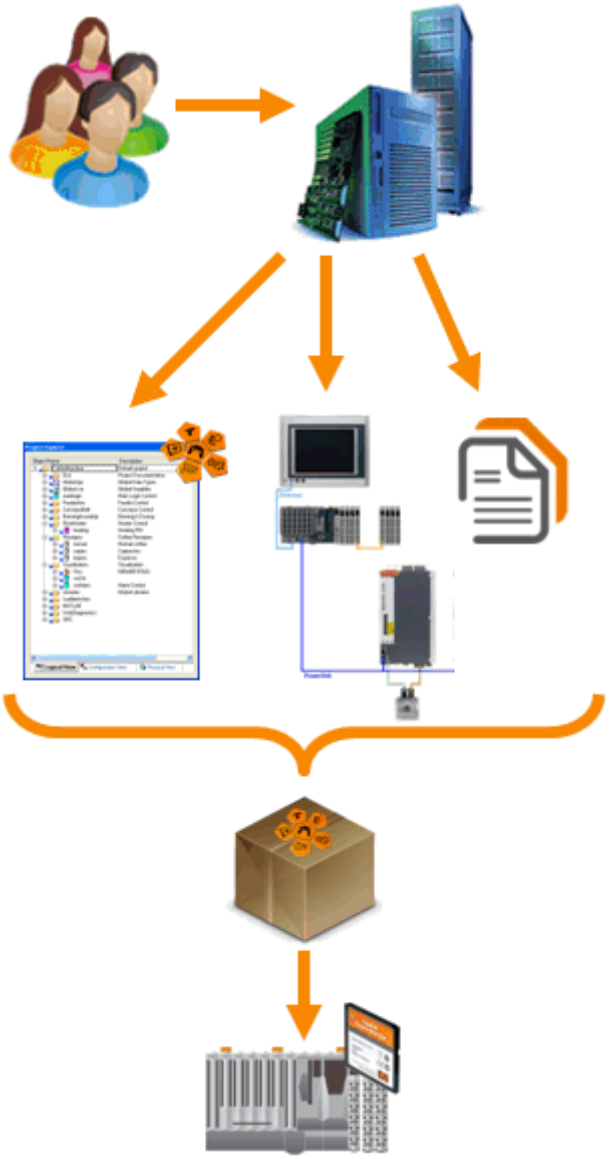
5.1 Auto generering

Det fungerar att autogenerera PLC-kod för båda fabrikaten och kompilera koden. Genereringsprogrammen frågar användaren ett antal frågor som denne ska skriva in och med informationen kan programmet bygga upp önskat projekt. När programmet är uppbyggt kan användaren skicka ner det till PLC. Om man skulle exekvera genereringsprogrammen igen bygger det upp en ytterligare projektstruktur med mer hårdvaror. Programmet skapar även HTML-sidan så att användaren kan se en presentation av det skapade projektet.

Det innebär att programmet innehåller väldigt mycket kod, för att det ska göra följande:

- Fråga användaren om information
- Deklarera och implementera
- Skapa hårdvaror
- Länka ihop mjukvaran och hårdvaran
- Genererar/manipulerar HTML-fil för att anpassa presentation

I ADS-programmet behöver man inte göra någon anpassning eftersom datagridview kan ändra sin storlek jämfört med HTMLs tabell.



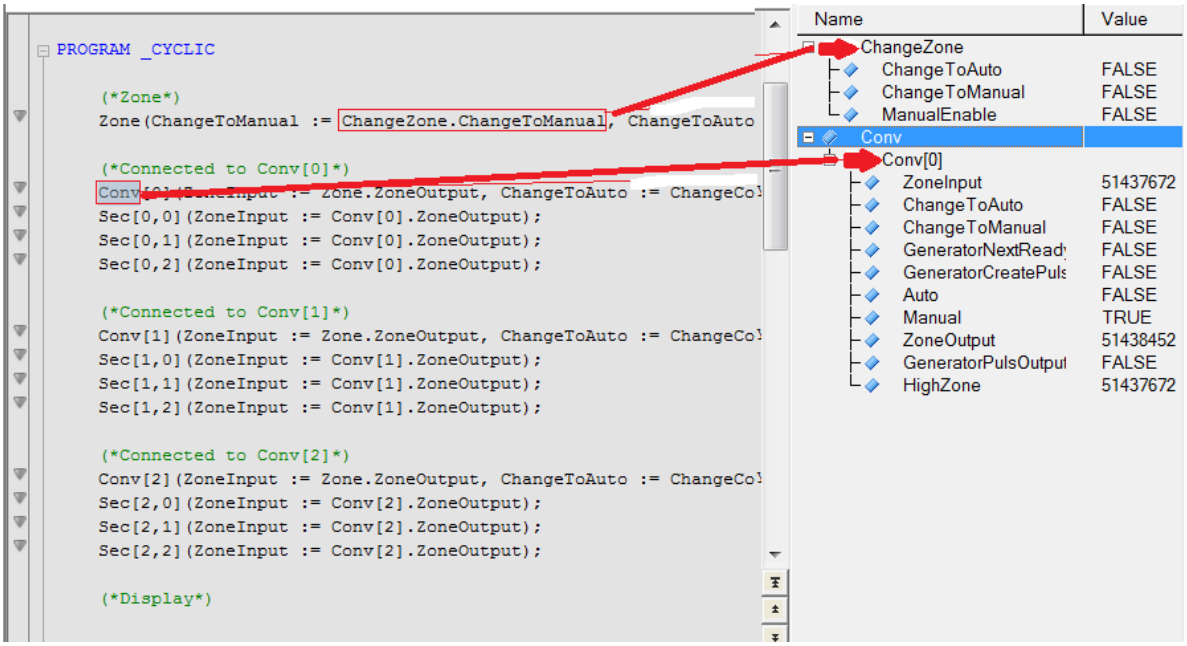
Figur 5.1 *En överblick på auto genereringsprogram*

5.2 Automation Studio och TwinCAT 3

Design på båda mjukvarorna Automation Studio och TwinCAT 3 ser helt olika ut men de har väldigt mycket gemensamt. På programmeringssidan är det lika. Där finns följande:

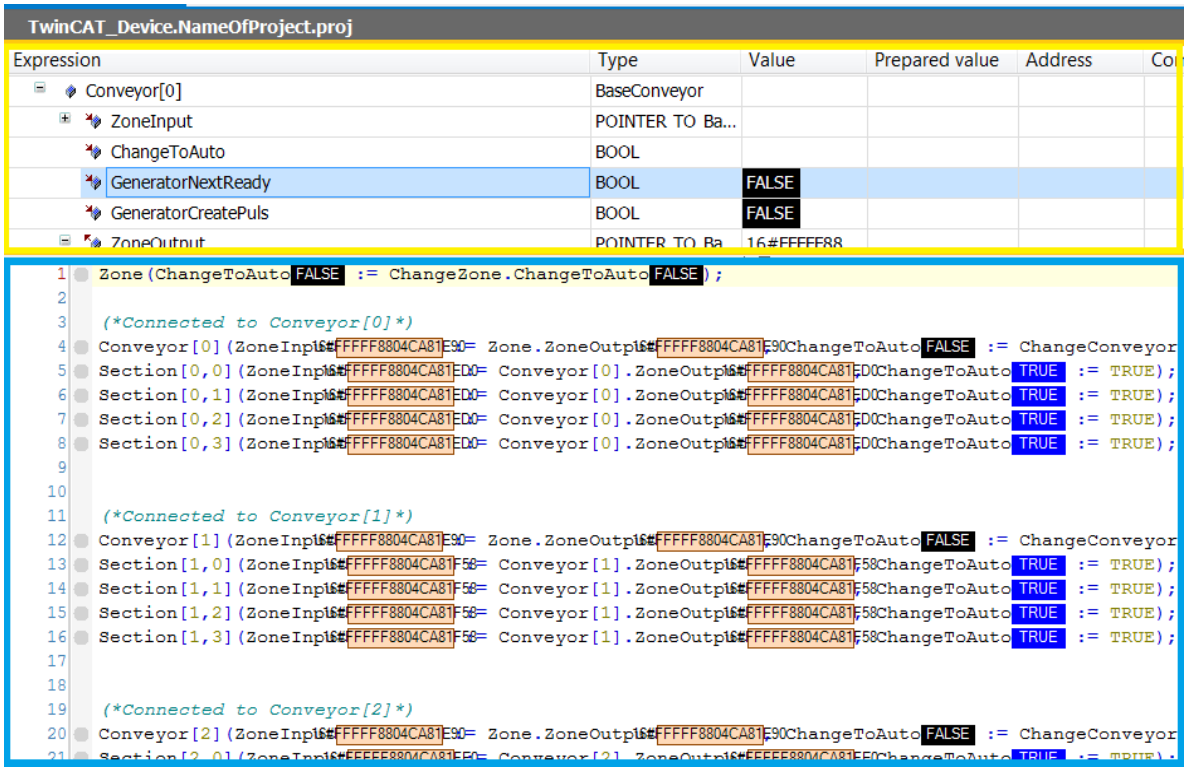
- IEC 61131-3 standard med objektorientering
- Continuous Function Chart (CFC)
- C/C++
- MatLAB/Simulink

Det finns även andra funktioner som debug i utvecklingsprogrammet som underlättar en användare för programmering, för att man kan se hur variabelvärden förändrats efter det att koden exekverats. Man kan även titta på variablernas värde under körning utan att gå in i debug mode vilket gör att man kan få reda på variabelns värde direkt. I Automation Studio kan man göra ”Drag and drop” på variabler till en” watch list” och då får man upp värde på variablerna. Det går även att dra dit en instans av ett funktionsblock eller en struct då dyker det upp all information som de innehåller. Det går även att ta bort de överflödiga variablerna. Se figur 5.2 för exempel.



Figur 5.2 *Automations Studios övervakningsyta*

Om man i TwinCAT3 gör en ”Login” så kommer man till en yta där det går att övervaka variablerna nästan på samma sätt som i Automation Studio. I den blå ramen, se figur 5.3, så är det en programmeringsyta där kan man få en snabb överblick på vilka värden vissa av variablerna innehåller. Skulle man vilja ha mer detaljerad information finns det i den gula ramen. Där kan man hitta alla variabler, struct eller funktionsblock och sedan leta efter den önskade variabeln.



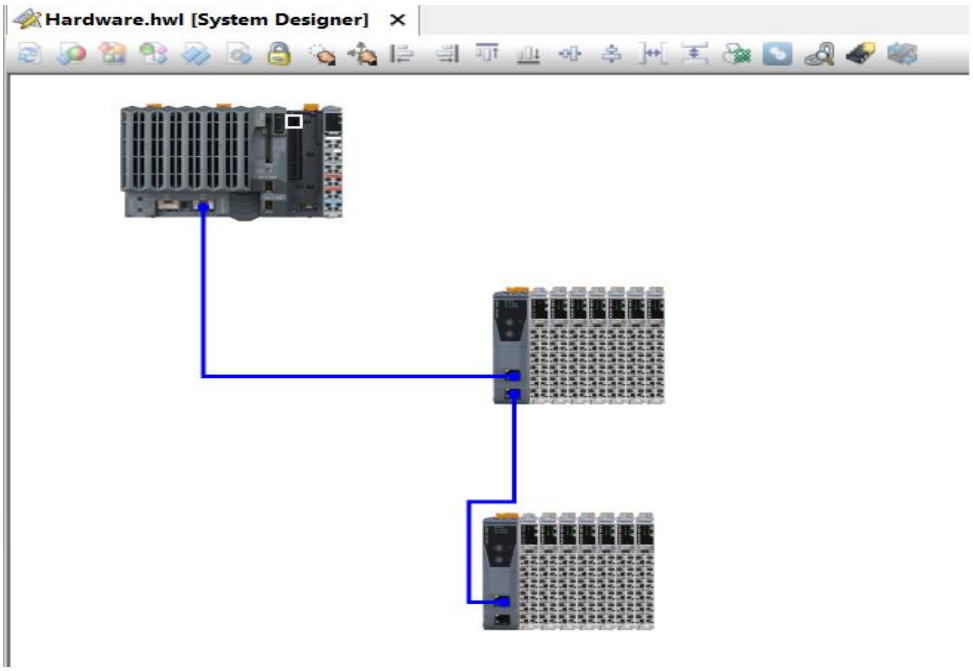
Figur 5.3 *TwinCAT3 i Login läge*

Även om det finns väldigt många likheter i båda mjukvarorna så finns det också olikheter. Protokollen är olika då B&R använder sig av Powerlink och Beckhoff använder sig av EtherCAT.

Det man ser i explorer-fönstret i Automation Studio går att öppna utanför utvecklingsprogrammet, det vill säga att man kan komma åt programmeringsfilen, hårdvarufilen och IOmapfilen(länkning mellan variabler och hårvaror). Eftersom det går att arbeta utanför Automation Studio kan man använda programmeringsverktyg som hanterar filer som t.ex radera/skapa filer, flytta filer eller lägga till/ta bort data i en fil.

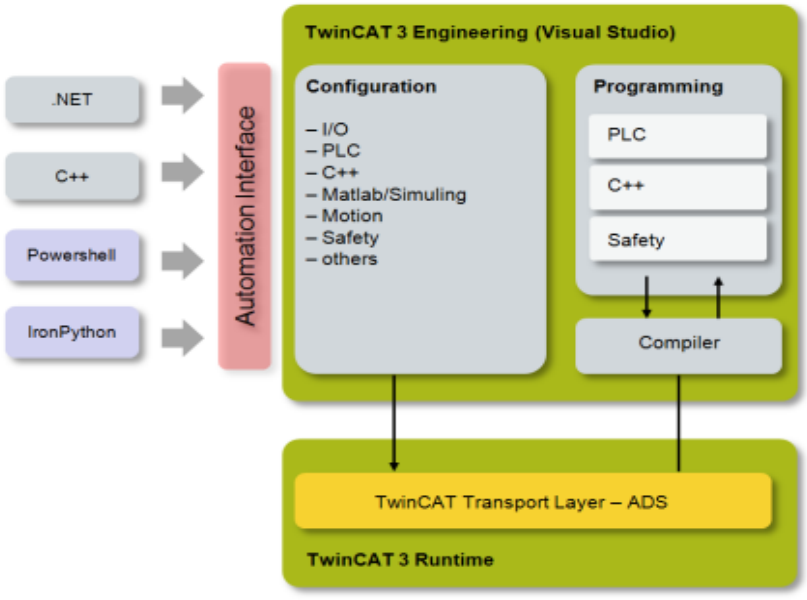
Automation Studio har även en annan sak som är olik, det är hardware.hwl –fil och det är en grafisk bild av hårdvarans uppsättning. Det visar vilka hårdvaror som används till PLC:n och det går att lägga till ytterligare hårdvaror genom att ”drag and drop” från Tool box. Det är fritt att flytta runt sakerna på

arbetsytan och blå koppling är en representation av Powerlink koppling, se figur 5.4.



Figur 5.4 *Hardware.hwl fil öppen i Automation Studio*

För att kunna använda Automation Interface behövs det ett programmeringsspråk som kan lägga till COM objekt som t.ex. C++ eller .NET. Automation Interface är ett gränssnitt som kan manipulera TwinCAT3 utifrån, som att lägga till PLC, Motion, I/O, safety m.m. Men det som skiljer sig mer är att det går att kompilera utanför också, det vill säga att det går att generera ett projekt och sedan skicka ner det direkt till PLC.



Figur 5.5 *En överblick på hur Automation Interface opererar*

6 Slutsats

Det finns möjlighet att generera ett helt projekt för att kunna skicka ner det till en PLC. Många mål lyckades att uppnå men inte allt och det blev en förändring under arbetets gång och det är CFC som är betydligt svårare att skapa utanför Automation Studio och TwinCAT3. Då är förslaget att ersätta CFC med något annat som HTML eller program som läser värdet från PLC. Att göra både HTML och programmet är för att presentera olika möjligheter för att komma runt CFC. Anledningen till att generera kod är för att kunna vinna mycket tid för att kunna lägga ner mer tid på en mer effektiv arbetsplats som t.ex. utveckla bibliotek för PLC. Tidsvinsten gör att det kan bli billigare för kunder att köpa lagersystem från Texo Application AB och att kunden kan i princip efter ett avslutat möte få med sig ett USB-minne med ett färdiggenererat projekt.

Skillnaden mellan Automation Studio och TwinCAT3 är väldigt liten, men det kan också vara för att båda utvecklar mot IEC 61131-3. Eftersom det mesta av tiden har ägnats åt autogenereringsprocessen vilket gör att denna jämförelse är mer baserad på vilka funktioner som finns och vad som syns i utvecklingsprogram. En annan jämförelse kan vara någon form av prestandajämförelse. Följande är en sammanfattning av skillnaderna:

<u>Automation Studio</u>

- Powerlink
- Filer i explorer är tillgänglig
- HWL fil (grafiskt)

<u>TwinCAT3</u>

- EtherCAT
- Automation interface

7 Framtida utvecklingsmöjligheter

En möjlig fortsättning på detta projekt är att översätta kod från Texo Application AB gamla system till det nya. Sedan följer automatisk generering av kod för projektet. En annan möjlighet är att få två eller flera PLC att kommunicera med varandra vilket gör att man kan bygga ut väldigt stora anläggningar.

Eftersom Automation Studio använder sig av ett filformat som är öppet för redigering även utanför utvecklingsverktyget, så kan Automation Studio vara ett smidigare och mer flexibelt alternativ att använda för Texo Application AB.

8 Terminologi

PLC – Programmable Logic Controller
TwinCAT 3 – Mjukvara för Beckhoff
Automation Interface – Gränssnitt som kan manipulerar TwinCAT
Automation Studio – Mjukvara för B&R
XML – Extensible Markup Language
COM – Component Object Model
I/O – Input/Output
Datagridview – Visar data i en tabell
Cycle – Scan igenom programmet på ett tidsintervall och gör en paus tills nästa skanning

9 Källkritik

PLCopen används eftersom det stödjer IEC61131-standarden. Mycket av information som har samlats in från w3school anses vara pålitlig eftersom det fungerar till mina program som är skapade i Javascript och VBScript. Andra källor som ”infosys.beckhoff” och ”B&R Help explorer” kan man lita på eftersom det innehåller information om utvecklingsprogrammet. Även information från anställda i B&R och Beckhoff kan anses tillförlitlig eftersom de har mycket goda kunskaper i Automation Studio respektive TwinCAT 3.

Referenser

[1]: PLC open. [Hämtad: 2014-06-03]
http://www.plcopen.org/pages/tc1_standards/iec61131-3/

[2]: Notepad++ [Hämtad: 2014-06-07]
http://notepad-plus-plus.org/

[3]:Automation Studio [Hämtad: 2014-05-14]
http://www.br-automation.com/en/products/innovations-2014/automation-studio/

[4]: TwinCat 3 [Hämtad: 2014-05-25]
http://infosys.beckhoff.com/content/1033/tc3_overview/html/tc3_overview_philosophy.htm?id=12543

[5]: VBScript [Hämtad: 2014-06-12]
http://msdn.microsoft.com/en-us/library/t0aew7h6(v=vs.84).aspx

[6]: JavaScript [Hämtad: 2014-06-12]
http://www.w3schools.com/js/

[7]: Automation interface [Hämtad: 2014-05-25]
http://infosys.beckhoff.com/content/1033/tc3_automationinterface/242682763.html?id=15082