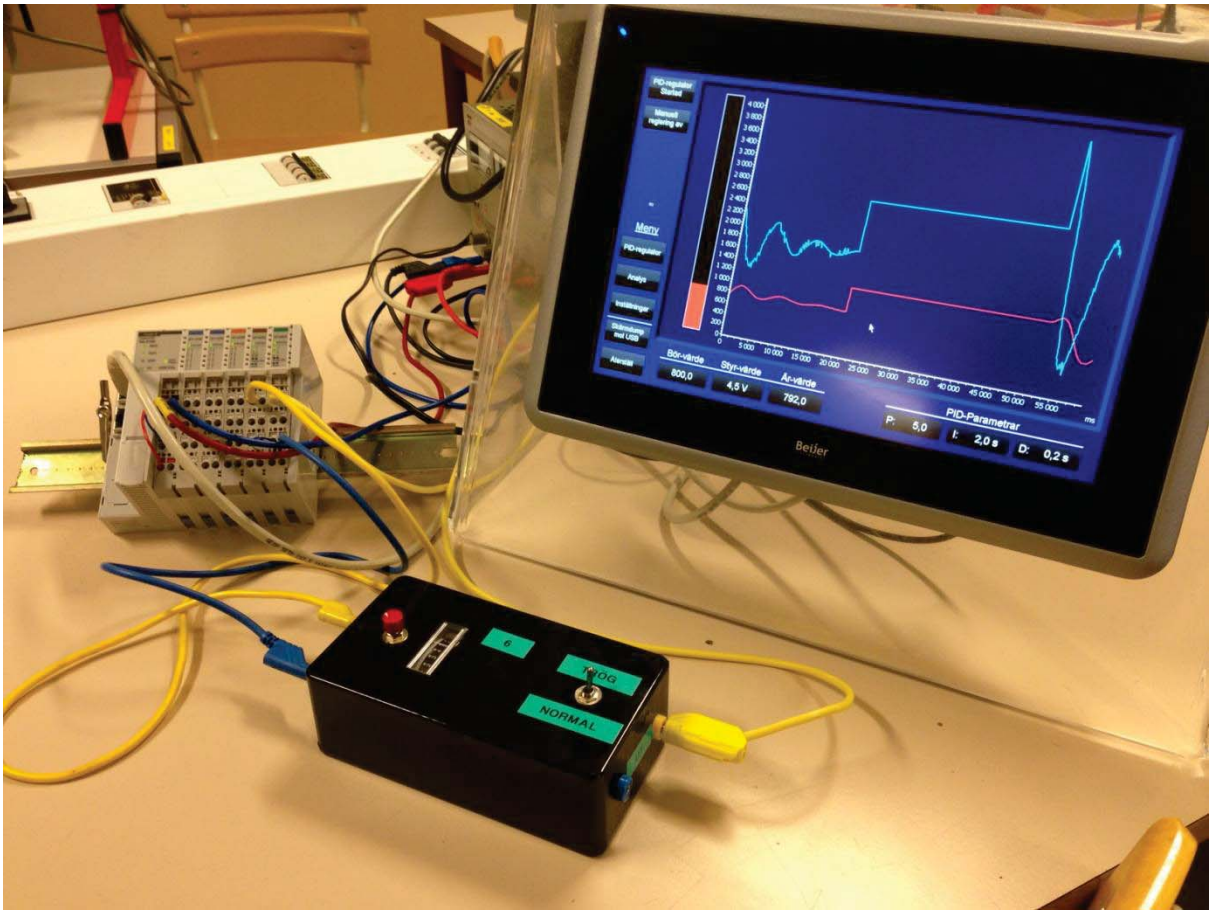
CHALMERS



Utrustning för identifiering och reglering av processer

Equipment for identifying and controlling processes

Examensarbete inom högskoleingenjörsprogrammet Elektroingenjör

Daniel Andersson

Johannes Anderzon

Institutionen för Signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

FÖRORD

Detta examensarbete är utfört vid institutionen Signaler och system på Chalmers tekniska högskola. Examensarbetet omfattar 15 högskolepoäng och vi som utfört detta har studerat på elektroingenjörprogrammet i tre år. Arbetet är gjort åt Chalmers tekniska högskola med universitetslektor Manne Stenberg som examinator och tekniklektor Morgan Osbeck som handledare. Examensarbetet är utfört i Chalmers Lindholmens lokaler och önskemålet med detta examensarbete var att framställa ett modernt laborationssystem för reglerteknik.

Vi vill tacka Morgan Osbeck för den tid, engagemang och hjälp vi fått av honom genom projektet samt forskningsingenjör Sakib Sistek för visat intresse. Vi vill också tacka studenterna Ola Vingren och Linda Alexandersson, som samtidigt bedrivit examensarbete, för att de har bidragit till en bra stämning på arbetsplatsen.

SAMMANFATTNING

Det här projektet har utförts på Chalmers Lindholmen och syftet är att studenterna där ska få nyttja ny utrustning inom reglerteknikslaborationer, då den gamla laborationsutrustningen är föråldrad. Projektet sammanfattar konstruktionen av ett program vars mål är att övervaka och styra en process med en PID-regulator. Programmet kan genomföra en analys av en process med frekvens, ramp och stegsvarsanalys och i analysen kan mätningar utföras på ett enkelt sätt. Programmet konstrueras för en touchbaserad panel där alla ingående system är inkluderade och all kontroll sköts mot ett grafiskt gränssnitt. Efter utfört projekt är resultatet en lovande konstruktion som fungerar bra. Hårdvaran som används är något svag för denna uppgift.

SUMMARY

This project has been performed at Chalmers Lindholmen and the purpose is to give the students an opportunity to use new equipment in automatic control engineering, the previous laboratory equipment is outdated. The project summarizes a construction of a program whose goal is to control and monitor a process with a PID controller and to perform frequency, ramp or step response analysis and to be able to make measurements at the result. This is done for a touch based operating panel where all necessary systems is integrated. The control is operated to a graphical interface. When the project was completed the result is a good construction. The hardware is somewhat slow for this task.

Innehållsförteckning

BETECKNINGAR

PLC - Programmable Logic Controller, Programmerbart Styrsystem

softPLC - Programmerbart styrsystem som med mjukvara installeras på person dator.

CODESYS - Utvecklingsmiljö för PLC programmering

ST - Strukturerad Text, textbaserat programmeringsspråk för PLC

FBD - FunktionsBlockDiagram, objektbaserat programmeringsspråk för PLC

HMI - Human Machine Interface, operatörsgränssnitt

T12B - Operatörspanel från Beijer Electronics

iX Developer - Utvecklingsmiljö för HMI, bl.a. T12B

EtherCAT - Fältbuss

PID-regulator - Proportionell, Integrerande, Deriverande regulator

I/O - Input/Output

C# - Textbaserat programmeringsspråk

INT - Integer, heltal mellan -32767 och 32767

UINT - Unsigned Integer, heltal mellan 0 och 65535

ULINT - Unsigned Long Integer, heltal mellan 0 och 4294967295

BOOL - Booleskvariabel, kan vara "TRUE" eller "FALSE"

FLOAT - Positiva och negativa decimaltal mellan $\pm 3{,}4 * 10^{38}$

REAL - Positiva och negativa decimaltal mellan $\pm 3{,}4 * 10^{38}$

MUX - Multiplexer

1 INLEDNING

I detta kapitel tas bakgrunden, syftet, målen och avgränsningar upp.

1.1 Bakgrund

Chalmers Campus Lindholmens situation inom reglerteknik är att dess laborationsutrustning kan anses något föråldrad. Laborationsutrustningen består endast av en process och en regulator och för att utföra en analys av processen görs detta manuellt med mätsticka. Om skolan ska kunna erbjuda modern kunskap och praktik inom reglerteknik samt fortsätta att ligga i framkant inom standarden på utbildningen måste den regelbundet utvecklas. Inköpet av ny utrustning 2013 har möjliggjort detta projekt som syftar till att utveckla möjligheter för utbildningens praktiska delar.

1.2 Syfte

Uppdraget är att skapa färdigutvecklad laborationsutrustning med pedagogisk utbildningsmiljö för reglerteknik. I utbildningsmiljön ska möjlighet finnas för steg-, ramp- och frekvenssvarsanalys samt en PID-regulator. Detta ska utvecklas till en touchbaserad operatörspanel. Utöver projektet i sig så är skolan intresserad av hur dess utvecklingsmiljöer lämpar sig i utbildningen för studenterna samt operatörspanelerna som verktyg.

1.3 Mål

Målet är att skapa en funktionell och lättanvänd laborationsutrustning i form av ett analysverktyg samt en regulator. Följande punkter är specificerade mål:

- Skapa en analysdel
- Skapa en regulatordel
- Inkludera en PID-regulator
- Kontinuerligt övervakningsdiagram av regulator och process
- Möjlighet att utföra steg-, ramp- och frekvenssvarsanalys av en process
- Kunna visualisera två signaler samtidigt under analys
- Möjlighet att enkelt kunna utföra mätningar av svaret ur en analys
- Hålla HMI-delen stilren, funktionell och tidlös

1.4 Avgränsningar

Följande begränsningar i funktionalitet har gjorts på grund av tidsbegränsningen och att funktionerna inte är nödvändiga:

- Skapa en simulerad process så systemet skulle kunna köras utan verklig process
- Möjligheten att ändra gränsvärden för de analoga in- och utsignalerna
- Ge möjligheten att ändra den totala körningstiden för analysdiagrammet

2 TEKNISK BAKGRUND

I detta kapitel finns bakgrunden till de system som är nödvändiga för att förstå innebörden av rapporten och projektet.

2.1 Regulatorer

En regulator används för att styra en process automatiskt. Regulatorn använder sig av tre signaler; börvärdet, styrvärdet och ärvärdet. Börvärdet är det värde som ställs in av användaren och är det värde som processen ska eftersträva oberoende av störningar. Regulatorn använder styrvärdet för att påverka processen. Ärvärdet är den nivå som regulatorn får som svar från processen beroende av styrvärdet. En vanlig typ av regulator är PID-regulatorn.

2.2 Programmerbart styrsystem

Programmerbara styrsystem är framtagna för att hantera automationsuppgifter och förkortas ofta med PLC. Ett programmerbart styrsystem skiljer sig från ett vanligt datorsystem genom att programmeringsspråken är anpassade till användningsområdet. PLC-systemet är oftast uppbyggt i moduler och monteras vanligen på DIN-skenor. I projektet används inte ett klassiskt PLC-system utan här används ett så kallat softPLC. Skillnaden mellan dessa två system är att softPLC enbart består av mjukvara som installeras på en persondator där sedan datorns portar används för kommunikation med processen. Detta öppnar för mindre men kraftfulla installationer av styrsystem. För att programmera ett PLC-system brukar tillverkaren av systemet även förse det med utvecklingsprogramvara eller används fristående programvaror som i detta projekt där programvaran CODESYS används. I projektet används två språk, strukturerad text (ST) vilket är ett textbaserat programmeringsspråk liknande C. ST används mest för konstruktion av funktionsblock, där dessa sedan sätts in i programspråket funktionsblockdiagram (FBD), vilket är ett grafiskt programmeringsspråk.

2.3 Decentraliserad Input/Output

Decentraliserat Input/Output, förkortas I/O, är in- och utgångar vilket ett PLC-system har kontakt med, men som inte befinner sig vid styrsystemet, utan decentraliserat med en seriekommunikation mellan enheterna. Portarna på den decentraliserade I/O-enheten kan vara både digitala och analoga. Seriekommunikationen kallas ofta för en fältbuss där antalet protokoll och typer som används är många. I detta projekt används fältbussen som kallas EtherCAT och baseras på Ethernet-protokollet.

2.4 Operatörsgränssnitt

Ett operatörsgränssnitt är ett verktyg för att kunna övervaka och påverka ett eller flera styrsystem. I projektet används en touchpanel där modellen kallas T12B och tillverkas av Beijer Electronics. Denna panel är på 12,1 tum, har ett bildförhållande på 16:10 och har en Microsoft Windows-baserad kärna. Det grafiska gränssnittet utvecklas i tillverkarens egna

programvara, iX Developer, vilket har en grafisk programmeringsmiljö och möjlighet till skrift med programmeringsspråket C# i script. T12B-panelen är också utrustad med ett softPLC som är frånskiljt från operatörsgränssnittet men som kan kopplas ihop i form av delade variabler. SoftPLC-delen används som styrsystem i detta projekt tillsammans med ett decentraliserat I/O över EtherCAT.

2.5 Ny utrustning på Chalmers

T12B-panelerna användes för första gången under den första läsperioden år 2014 i kursen Industriella styr- och övervakningssystem (LEU340) och då utnyttjades enbart HMI-delen medan softPLC-delen utelämnades. Utvecklingsmiljön CODESYS är nyinförskaffad och används för första gången i detta projekt. Ett helt nytt decentraliserat I/O med både digitala och analoga in- och utgångar från CREVIS har köpts in vilket använder EtherCAT-protokollet. I HMI-delen kan beräkningar utföras med hjälp av script som skrivs i C#, detta är ett programspråk som inte stötts på tidigare i utbildningen.

3 METOD

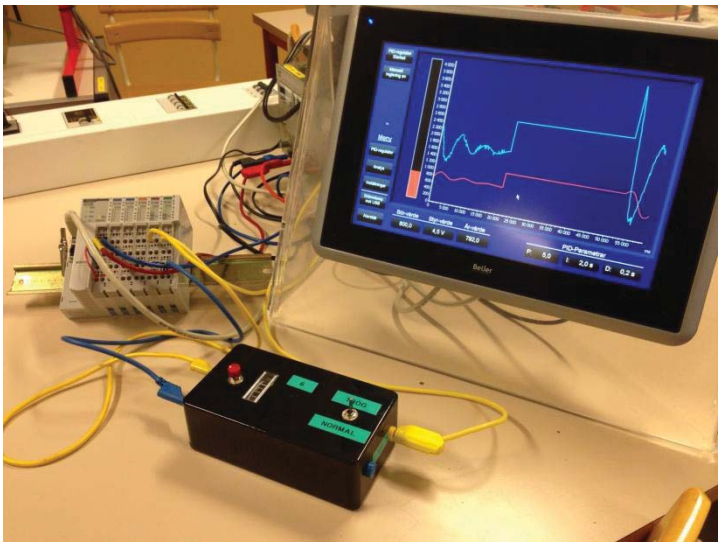
Först konfigurerades CODESYS för T12B-panelen, enligt konfigurationsmanual i Bilaga A. För softPLC-delen gjordes konfigurering av den decentraliserade I/O-enheten med hjälp av medföljande datablad från Beijer Electronics och konfigurationsmanual enligt Bilaga C. Den decentraliserade I/O-enheten använder sig av EtherCAT som fältbuss vilken inte behöver någon konfigurering. Efter konfigurering av hela systemet påbörjades programmeringen för softPLC-delen. Det som gjordes var att genomsöka bibliotek efter befintliga funktionsblock som finns i CODESYS. De funktioner som saknades behövdes då konstrueras. De funktionsblock som konstruerades gjordes med hjälp av inbyggda manualer i CODESYS och med tidigare kunskaper från PLC-programmering i annan utvecklingsmiljö.

HMI-delen konfigurerades enligt Bilaga B. Först användes HMI-delen som en testbänk där de olika funktionerna placerades ut. När funktionerna fungerade som de var tänkt i softPLC-delen så standardiserades de till en egen projekt standard. För att konfigurera mer komplicerade objekt i HMI-delen användes den inbyggda manualen utöver tidigare kunskaper inom iX Developer. Designen av HMI-delen gjordes mot slutet av projektet och baserades på tidigare kunskaper från kursen Industriella styr- och övervakningssystem. Utöver detta användes också yrkeserfarenheter i ämnet webbdesign. Vissa beräkningar görs i HMI-delen med hjälp av script i programmeringsspråket C#. Tidigare erfarenheter inom språket saknades och problemet löstes genom att finna grundläggande kunskaper inom C# (1). Därefter med tidigare programmeringskunskaper så testades konstruerade script tills de fungerade.

Arbetssättet har genomgående för hela projektet varit enligt trial and error-metoden och utifrån tidigare erfarenheter genom hela studietiden varit ett välfungerande arbetssätt.

4 PRESENTATION AV SYSTEMET

Systemet är uppbyggt med T12B-panelen tillsammans med en decentraliserad I/O-enhet. I/O-enheten är kopplad till en process vilket i detta arbete har varit en simulerad värmeugn samt en vattentank med pump, se Figur 4.1. T12B-panelens softPLC har programmerats fristående och delar signaler med HMI-delen. HMI-delen använder sig av signalerna från softPLC-delen och visar dessa värden samt ger möjligheten att manipulera värdena i softPLC-delen. Systemet har två huvudfunktioner där den ena är en analysdel och den andra är en regulatordel.



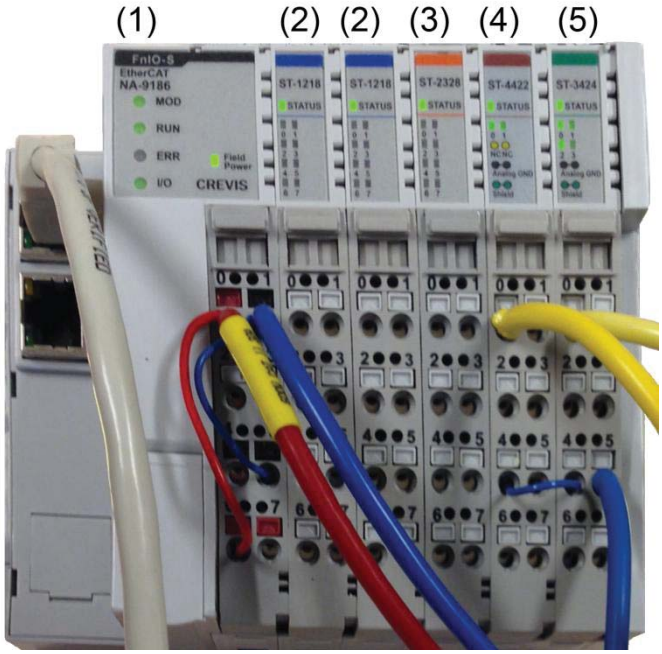
Figur 4.1 - Överblick av hela systemet, i bild syns till vänster den decentraliserade I/O enheten, till höger syns T12B-panelen och i mitten syns en process i form av den simulerade värmeugnen.

Analysdelen består av tre olika generatorer; steggenerator, rampgenerator och signalgenerator. Dessa kan användas till att utföra olika analyser av en valfri process. För att kunna analysera en process sparas ärvärdet från processen och jämförs mot styrsignalen där dessa sparas i var sin array med 60000 element. Detta visualiseras i ett diagram där de två signalerna ritas upp. Möjligheten att placera ut mätpunkter i diagrammet i form av plustecken. De precisa värdena av mätpunkterna kan utläsas.

Regulatordelen innehåller en PID-regulator för att kunna testa olika parameterinställningar till denna typ av regulator. Med regulatordelen finns också ett diagram där styrvärdet och ärvärdet kan följas kontinuerligt. Diagrammet är endast till för att observera processens beteende med regulator, vilket medför att mätningar inte kan utföras med hjälp av mätpunkter i detta diagram.

5 HÅRDVARUKONFIGURATION

Hårdvarukonfigurationen påbörjas genom att koppla ihop en dator med T12B-panelen över ett nätverk via en switch. Om endast en panel ska användas så räcker det med en korsad nätverkskabel mellan dator och T12B-panelen. Den decentraliserade I/O-enheten tillkopplas expansionsmoduler enligt *Figur 5.1*.



Figur 5.1 - Decentraliserat I/O (1) som består av två moduler med åtta digitala ingångar vardera (2), en modul med åtta digitala utgångar (3),en modul med två analoga utgångar (4) samt en modul med fyra analoga ingångar (5).

Modulerna monteras på en DIN-skena för att stabilisera dem samt att spänningsmatningen kopplas enligt medföljande beskrivning. T12B-panelen och den decentraliserade I/O-enheten med dess expansionsmoduler matas med 24 V.

6 MJUKVARUKONFIGURATION

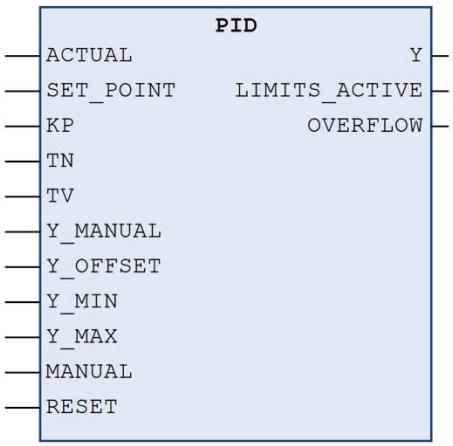
- CODESYS konfigureras för att hantera T12B-panelens softPLC, konfiguration samt inställningar finns i Bilaga A. Detta är ett tillägg i CODESYS då panelen annars inte skulle kännas igen av CODESYS.
- För att sammanlänka HMI-delen med softPLC-delen konfigureras iX Developer enligt Bilaga B. Detta möjliggör import av variabler till iX Developer från CODESYS.
- För att konfigurera den decentraliserade I/O-enheten i CODESYS se Bilaga C. Denna konfiguration gör att softPLC-delen får kontakt med den decentraliserade I/O-enhetens alla variabler för de fysiska in- och utgångarna.
- För att överföra de variabler som ska användas i iX Developer från CODESYS, se Bilaga D. Detta förklarar exporten från CODESYS och importen i iX Developer av valda variabler.

7 FUNKTIONSBLOCK I CODESYS

I detta kapitel beskrivs alla block som används i PLC-programmen, både egenkonstruerade och befintliga mer komplexa funktionsblock. I de flesta delkapitel finns en figur på det berörda blocket samt en tabell med dess in- och utsignaler samt förklaringar och datatyper för dessa signaler.

7.1 PID-blocket

PID-blocket enligt Figur 7.1 finns i tilläggsbiblioteket som heter "Util". Detta block används senare i kapitel 8.3. I biblioteket finns också en PD-regulator och en PID-regulator med valbar samplingstid. Beskrivning av regulatorns in- och utsignaler finns i Tabell 7.1. Blocket är en PID-regulatorfunktion där P-, I- och D-värde kan manipuleras, det finns även offset och gränsvärden.



Figur 7.1 - Illustration av PID-blocket från CODESYS.

Regulatorn får initialvärden vid uppstart för att undvika frysning av softPLC-delen. Vissa värden kan inte vara noll vilket är initialvärdet för alla variabler om inget annat anges. PID-blocket arbetar med typen REAL som fungerar som flyttal men är- och styrvärdet är konfigurerade enligt UINT vilket endast innehåller positiva heltal, därför sitter vissa typkonverteringar mot PID-blocket. PID-regulatorn testades mot en simulerad ugn i form av en svart låda med in- och utsignal kopplade till regulatorn genom de analoga portarna på den decentraliserade I/O-enheten. Utifrån enklare tester fungerar PID-regulatorn mot ugnen tillfredsställande.

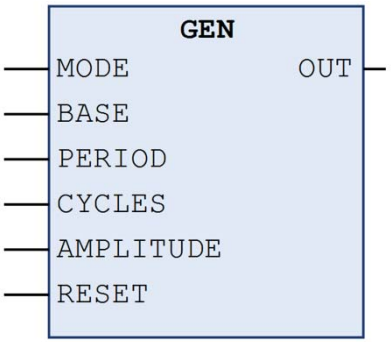
Tabell 7.1 - Beskrivning av PID-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
ACTUAL	REAL	Ärvärdet
SET_POINT	REAL	Börvärde
KP	REAL	Proportionell konstant
TN	REAL	Integreringstiden i sekunder
TV	REAL	Deriveringstiden i sekunder
Y_MANUAL	REAL	Värdet sätts på Y om MANUAL=TRUE
Y_OFFSET	REAL	Förskjutning av värdet på Y
Y_MIN	REAL	Minsta värdet vilket Y kan erhålla
Y_MAX	REAL	Största värdet vilket Y kan erhålla
MANUAL	BOOL	Om MANUAL=TRUE flyttas värdet på Y_MANUAL till Y, annars bestäms Y av regulatorn
RESET	BOOL	Kopierar värdet från Y till Y_OFFSET och återställer integraldelen

Y	REAL	Styrvärdet
LIMITS_ACTIVE	BOOL	Om regulatorn försöker ge ett värde på Y utanför Y_MIN och Y_MAX är denna signal TRUE
OVERFLOW	BOOL	Överströmning i integraldelen

7.2 GEN-blocket

Signalgeneratorn enligt Figur 7.2 finns i tilläggsbiblioteket "Util" under namnet "GEN". Detta block används senare i kapitel 8.1. Typkonverteringar sitter i anslutning mot GEN-blocket så att variablerna stämmer mot övriga variabler. Beskrivning av signalgeneratorns in- och utsignaler finns i Tabell 7.2. Blockets funktion är att skapa olika signalformer; sinus, cosinus, triangel, fyrkant och sågtand. Faktorer för att skala om signalen till korrekta intervall har implementerats då signalen i sitt grundutförande kan variera i intervallet 0 till 4095 som motsvarar 0 till 10 V. En offsetfunktion har komplementerats utanför detta block för att möjliggöra andra likspänningsnivåer än 0 V. För att kunna växla mellan olika signalformer har en multiplexer anslutits mot MODE-signalen som innehåller typnamnen för de olika formerna för denna specialtyp. Till ingången "PERIOD" kopplas "Freq_Period_converter" som förenklar inmatningen av periodtiden genom att möjliggöra inmatning av frekvens också, se kapitel 7.3. Ett egenkonstruerat fördröjningsblock kopplas till på signalgeneratorblocket för att möjliggöra en korrekt start där den valda signalen startar efter rätt tid vid signalens startpunkt när alla grafer är redo, se kapitel 7.4.



Figur 7.2 - Illustration av signalgeneratorblocket från CODESYS.

Ett oscilloskop kopplades mot den decentraliserade I/O-enhetens ena analoga utgångsport där signalgeneratorn sänder ut signalen. Under detta test upptäcktes det att signalgeneratorns upplösning var undermålig vilket resulterade i en ändring av uppdateringstiden på EtherCAT-bussen från 20 ms till 2 ms vilket är den decentraliserade I/O-enhetens minimala uppdateringstid. Efter justeringen fungerade signalgeneratorn tillfredsställande med tillräcklig upplösning. Dock upptäcktes det att utsignalen kunde få ett värde som skulle motsvara mer än 10 V som resulterade i att signalen började om från 0 V. Detta löstes genom att ansluta blocket "LIMIT" från standardbiblioteket som begränsar signalen inom 0 till 10 V.

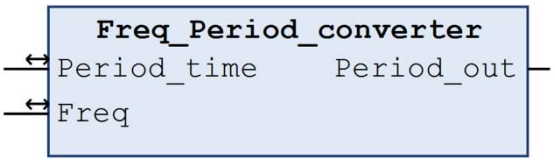
Tabell 7.2 - Beskrivning av signalgeneratorblockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
MODE	GEN_MODE	Val av signalform mellan triangel sågtand stigande, sågtand fallande, fyrkant, sinus och cosinus
BASE	BOOL	Val mellan PERIOD (TRUE) och CYCLES (FALSE)
PERIOD	TIME	Periodtiden då BASE=TRUE
CYCLES	INT	Antal samplingar per period då BASE=FALSE (används inte i denna lösning)

AMPLITUDE	INT	Amplitud
RESET	BOOL	Återställning
OUT	INT	Genererad utsignal

7.3 Freq_Period_converter-blocket

Freq_Period_converter-blocket enligt Figur 7.3 är ett egenkonstruerat block vars uppgift är att omvandla inmatad frekvens till periodtid samt att omvandla inmatad periodtid till frekvens. Detta block används senare i kapitel 8.1. Blocket skickar även tillbaka det nya värdet på den kanal som inte blev uppdaterad utifrån för att detta ska kunna presenteras på operatörspanelen. Beskrivning av Freq_Period_converter-blockets in- och utsignaler finns i Tabell 7.3. Test av blocket genomfördes mot signalgeneratorblocket vilket gav bra resultat, för närmare studie se koden i Bilaga E.



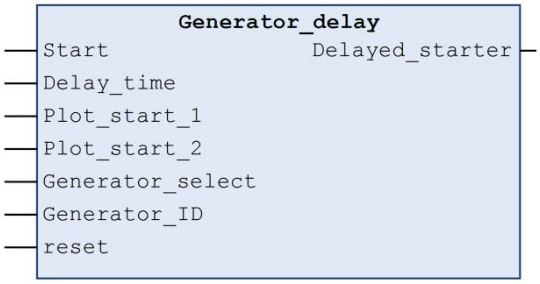
Figur 7.3 - Illustration av egenkonstruerat Freq_Period_converter-block.

Tabell 7.3 - Beskrivning av Freq_Period_converter-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
Period_time	UINT (in_ut)	Inmatat periodtid som även matas tillbaka om frekvensen ändras
Freq	REAL (in_ut)	Inmatat frekvens som även matas tillbaka om periodtiden ändras
Period_out	UINT	Utmatad periodtid till tillkopplat block

7.4 Generator_delay-blocket

Generator_delay-blocket enligt Figur 7.4 är ett egenkonstruerat block vars uppgift är att kunna ge en fördröjd start av tillkopplad generatorfunktion. Detta block används senare i kapitel 8.1. Blocket ska också vänta på att kurvorna som ska rita upp signalerna har startat innan fördröjningstiden börjar räkna ner. Beskrivning av Generator_delay-blockets in- och utsignaler finns i Tabell 7.4. Blocket testades mot signalgeneratorblocket och efter enklare justeringar fungerar det tillfredsställande, för närmare studie se koden i Bilaga F.



Figur 7.4 - Illustration av egenkonstruerat Generator_delay-block.

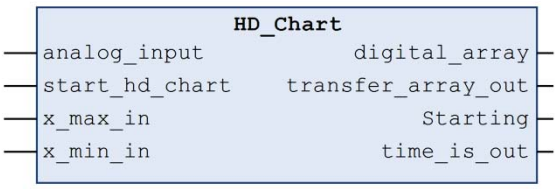
Tabell 7.4 - Beskrivning av Generator_delay-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
Start	BOOL	Startar blocket

Delay_time	UINT	Sätter fördröjningstid i ms
Plot_start_1	BOOL	Väntar på att första kurvan startar
Plot_start_2	BOOL	Väntar på att andra kurvan startar
Generator_select	UINT	Val av generator som aktiverar blocket om samma som "Generator_ID"
Generator_ID	UINT	Nummer för att kunna särskilja flera likadana block åt
reset	BOOL	Återställning för att kunna utföra ny start
Delayed_starter	BOOL	När allt ovan är uppfyllt och fördröjningstiden är över ges en puls

7.5 HD_Chart-blocket

HD_Chart-blocket enligt Figur 7.5 är ett egenkonstruerat block vars uppgift är att spara ner värden till en stor array som används för att presentera en kurva. Detta block används senare i kapitel 8.2. Arrayen är på 60000 element och måste skalas ner då HMI-delen inte kan hantera mer än 512 element i en array. Nedskalning görs då mot en mindre array på 500 element. Blocket plockar ut jämnt fördelade värden ur den stora arrayen som kopieras över till den mindre arrayen. Denna urplockning av värden kan justeras för att få fram ett önskvärt intervall ur den stora arrayen. Beskrivning av HD_Chart-blockets in- och utsignaler finns i Tabell 7.5.



Figur 7.5 - Illustration av egenkonstruerat HD_Chart-block.

HD_Chart-blocket testades med signalgeneratorblocket som insignal och värdena som sparades ner skickades till HMI-delen för att kunna presenteras som en kurva. Från början var HD_Chart-blockets inre uppdelat i flera egenkonstruerade block vilka sammanfogades i utanpåliggande block, vilket gav två nivåer av egenkonstruerade block. Den djupaste nivån av egenkonstruerade block var skrivna i ST och nivån ovanför var uppbyggd i FBD med vissa standardblock för att få de egenkonstruerade blocken att arbeta ihop. När denna konstruktion fungerade tillfredställande konstruerades den om till enbart en nivå i språket ST för en effektivare kod. För närmare studie se koden i Bilaga G.

Tabell 7.5 - Beskrivning av HD_Chart-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
analog_input	UINT	Insignal som sparas
start_hd_chart	BOOL	Start av blocket med reset som sker först
x_max_in	UINT	Sätter max värde som skalas om till "transfer_array_out" och skapar ett intervall med "x_min_in"

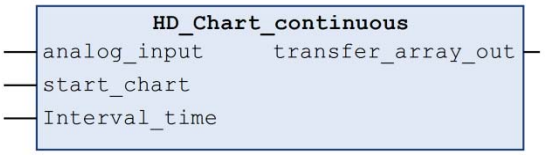
x_min_in	UINT	Sätter min värde som skalas om till "transfer_array_out" och skapar ett intervall med "x_max_in"
digital_array	ARRAY[0..60000] OF UINT	Alla värden som sparats ner
transfer_array_out	ARRAY[0..499] OF UINT	Array som tar ut 500 värden beroende på "x_max_in" och "x_min_in"
Starting	BOOL	Nu är reset klart och värden börjar sparas ner
time_is_out	BOOL	"digital_array" är full, värden sparas inte längre från "analog_input"

7.6 Programmet X_scaler

X_scaler är en funktion som i detta fallet inte behöver vara i ett funktionsblock eftersom denna funktion endast används en gång oberoende av andra block. X_scaler-programmets uppgift är att fördela ut 500 värden i en array på ett inställbart intervall. Detta används i samband med grafutskrift i HMI-delen med HD_Chart. För närmare studie se koden i Bilaga H.

7.7 HD_Chart_continuous-blocket

HD_Chart_continuous-blocket enligt Figur 7.6 är ett egenkonstruerat block vars uppgift är att sampla värden till en array med 500 element i ett rullande schema med ett valbart intervall. Detta block används senare i kapitel 8.3. och är en vidareutveckling av blocket HD_Chart vilket kan läsas om i kapitel 7.5. Beskrivning av blocket HD_Chart_continuous in- och utsignaler finns i Tabell 7.6. Blocket testades med en process vilket styrdes av en PID-regulator där styr- och ärvärdet presenterades med ett diagram i HMI-delen. Testet gav goda resultat då blocket har likheter med HD_Chart-koden. För närmare studie se HD_Chart_continuous-koden i Bilaga I.



Figur 7.6 - Illustration av egenkonstruerat HD_Chart_continuous-block.

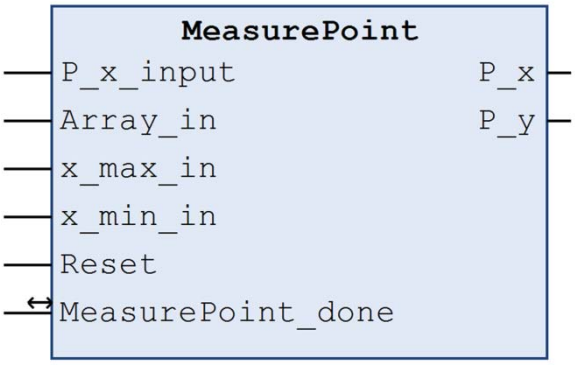
Tabell 7.6 - Beskrivning av HD_Chart-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
analog_input	UINT	Insignal som sparas
start_chart	BOOL	Start av blocket med reset som sker först
Interval_time	UINT	Sätter intervalltid i sekunder för hur ofta värden ska sparas ned från "analog_input" till "transfer_array_out" om programmet körs med 4 ms cykeltid. Kan endast sättas från två sekunder och uppåt i jämna tal

transfer_array_out	ARRAY[0..499] OF UINT	Array som sparar 500 värden från "analog_input" och sedan sparar över de äldsta värdena när intervalltiden löper ut

7.8 MeasurePoint-blocket

MeasurePoint-blocket enligt Figur 7.7 är ett egenkonstruerat block vars uppgift är att placera ut ett plustecken på inmatad array vilket resulterar i en variabel mätpunkt. Blocket används senare i kapitel 8.2. Blocket är framtaget för att enkelt kunna göra mätningar på en kurva. Beskrivning av MeasurePoint-blockets in- och utsignaler finns i Tabell 7.7. "Array_in" får en array med max 60001 element som innehåller alla mätpunkter med 4 ms avstånd, detta är samma array som används för att forma kurvan där plustecknet sätts ut. "P_x_input" pekar på det efterfrågade elementet och MeasurePoint-blocket formar ett plustecken centrerat på denna punkt. För att öka smidigheten samt att begränsa tung programmering i iX Developer finns det en typkonvertering framför "P_x_input" och en multiplikation med 500. Enheten som matas in blir i sekunder med decimaler istället för en halv millisekund per steg. Tester utfördes genom utsättning av mätpunkten på det analysdiagrammet. Sedan studerades resultatet för att se om punkten hamnade på rätt ställe samt att övriga funktioner inte stör mätpunkten. Efter tester gjordes justeringar vilka löste problemen. För närmare studie se koden i Bilaga J.



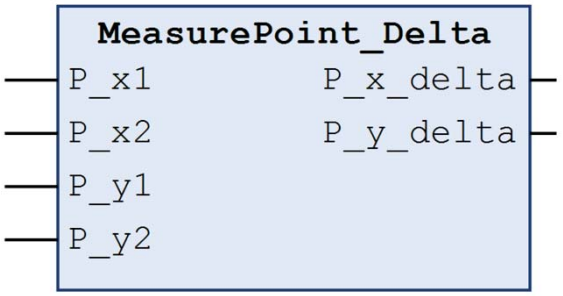
Figur 7.7 - Illustration av egenkonstruerat MeasurePoint-block.

Tabell 7.7 - Beskrivning av MeasurePoint-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
P_x_input	UINT	Begärt X-värde på "Array_in"
Array_in	ARRAY [0..60000] OF UINT	Array där Y-värdet kan hämtas
x_max_in	UINT	Används för att bestämma bredden på plustecknet
x_min_in	UINT	Används för att bestämma bredden på plustecknet
Reset	BOOL	Sätter arrayerna P_x och P_y till "0"
MeasurePoint_done	BOOL (in-ut)	Mätpunkten färdigbehandlad efter ändring, värdet inverteras
P_x	ARRAY [0..4] OF UINT	X-värden för plustecknet, begärt värde finns på P_x[2]
P_y	ARRAY [0..4] OF UINT	Y-värden för plustecknet, begärt värde finns på P_y[2]

7.9 MeasurePoint_Delta-blocket

MeasurePoint_Delta-blocket enligt Figur 7.8 är ett egenkonstruerat block vars uppgift är att jämföra två X-koordinater med varandra samt två Y-koordinater och presentera skillnaden mellan dessa. Detta block används senare i kapitel 8.2. Blocket förenklar mätningar i grafer för användaren då den kontinuerligt räknar ut skillnaden mellan två punkter. Beskrivning av MeasurePoint_Delta-blockets in- och utsignaler finns i Tabell 7.8. Testerna vilket genomfördes för kontroll av funktion visade goda resultat. En justering gjordes så att de två olika mätpunkterna från MeasurePoint-blocken kunde placeras i valfri ordning. För ytterligare studie av lösningen se Bilaga K.



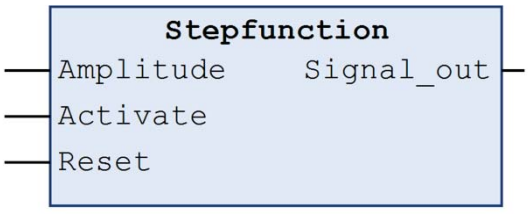
Figur 7.8 - Illustration av egenkonstruerat MeasurePoint_Delta-block.

Tabell 7.8 - Beskrivning av MeasurePoint_Delta-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
P_x1	UINT	Första X-värdet som ska jämföras
P_x2	UINT	Andra X-värdet som ska jämföras
P_y1	UINT	Första Y-värdet som ska jämföras
P_y2	UINT	Andra Y-värdet som ska jämföras
P_x_delta	ULINT	Skillnaden mellan första och andra X-värdet
P_y_delta	UINT	Skillnaden mellan första och andra Y-värdet

7.10 Stepfunction-blocket

Stepfunction-blocket enligt Figur 7.9 är ett egenkonstruerat block vars uppgift är att på given signal kunna skicka ut ett steg till inställd nivå. Blocket används senare i kapitel 8.1. Beskrivning av Stepfunction-blockets in- och utsignaler finns i Tabell 7.9. Det egenkonstruerade funktionsblocket "Generator_delay" används i anslutning till detta block för att möjliggöra en fördröjd start, se kapitel 7.4. Funktionstest av blocket genomfördes med goda resultat. För närmare studie se Bilaga L.



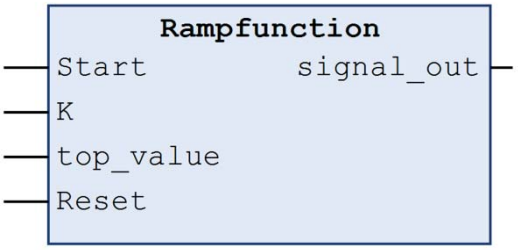
Figur 7.9 - Illustration av egenkonstruerat Stepfunction-block.

Tabell 7.9 - Beskrivning av Stepfunction-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
Amplitude	UINT	Förinställning av nivå för steget
Activate	BOOL	Sänder ut inställd nivå i ett steg
Reset	BOOL	Återställer nivån till "0"
Signal_out	UINT	Signalen presenteras här

7.11 Rampfunction-blocket

Rampfunction-blocket enligt Figur 7.10 är ett egenkonstruerat block vars uppgift är att skicka ut en ramp med inställd lutning samt inställd amplitud vid given signal. Blocket används senare i kapitel 8.1. Beskrivning av Rampfunction-blockets in- och utsignaler finns tillgängliga i Tabell 7.10. Det egenkonstruerade funktionsblocket vid namn "Generator_delay" används i anslutning till detta block för att möjliggöra en fördröjd start, se kapitel 7.4. Funktionstest av blocket genomfördes med goda resultat, för närmare studie se Bilaga M. Detta block konstruerades då blocket som redan fanns i biblioteket "Util" har en besvärlig inställning av hur lutningen justeras. Om det färdiga blocket hade använts skulle det krävas avancerad logik för att översätta denna inställning.



Figur 7.10 - Illustration av egenkonstruerat Rampfunction-block.

Tabell 7.10 - Beskrivning av Stepfunction-blockets in- och utsignalers variabelnamn.

Variabelnamn	Typ	Beskrivning
Start	BOOL	Startar rampen
K	REAL	Antal steg på Y-axeln per sekund
top_value	UINT	Värdet där rampen övergår till plan nivå
Reset	BOOL	Återställer generatorn för nytt test
signal_out	UINT	Signalen skickas ut

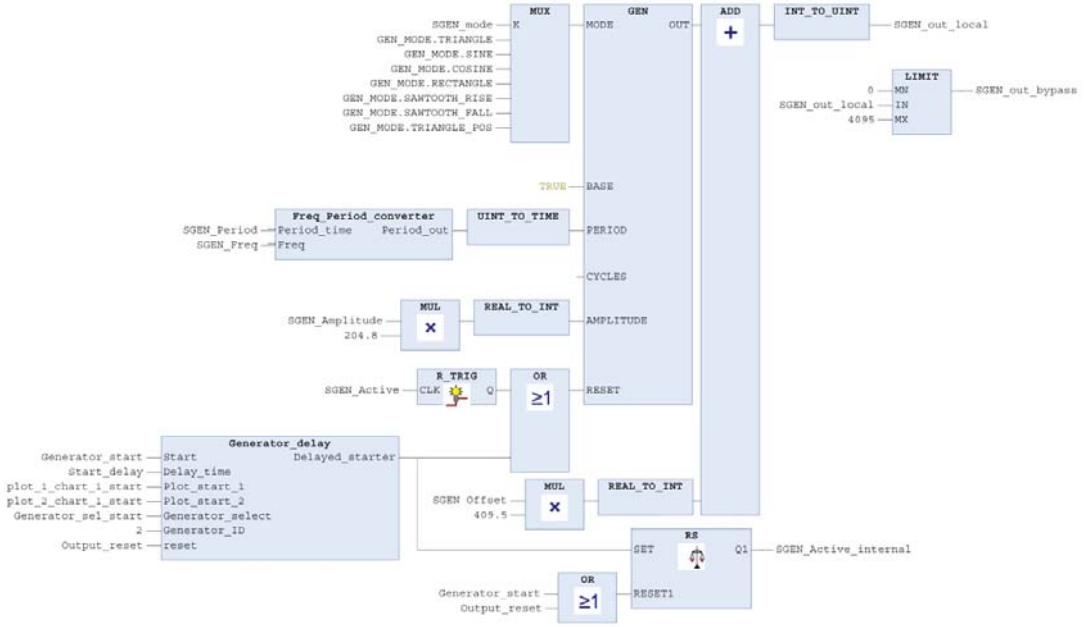
8 PROGRAM (PRG) I CODESYS

I detta kapitel beskrivs programkoderna som används i softPLC-delen I varje delkapitel finns figurer av koden samt hänvisningar till tabeller och den kommenterade koden.

8.1 Generators

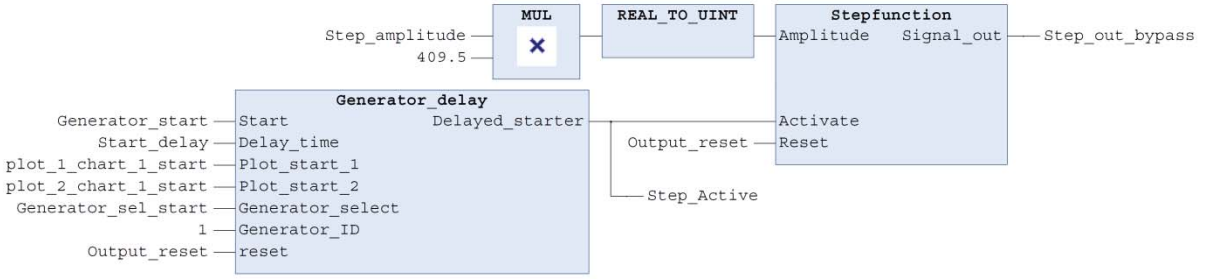
I detta program finns generatorerna placerade men de är uppdelade i olika nätverk. Första nätverket innehåller allt som behövs för signalgeneratorn enligt Figur 8.1. GEN-blocket är kärnan i första nätverket där valet av signalform styrs av "SGEN_mode" som flyttar över den specialtyp som GEN-blocket behöver med hjälp av en MUX (multiplexer). För att ställa in periodtiden används blocket Freq_Period_converter där signalerna "SGEN_Period" och "SGEN_Freq" indikerar inställd periodtid respektive frekvens. Ingången "BASE" sätts till "TRUE" vilket resulterar i att endast ingången "PERIOD" i GEN-blocket används. Amplituden för signalen ställs in med "SGEN_Amplitude" som skalas om från 0-10 V för att arbeta i intervallet ±2048.

När signalgeneratorn ska startas tillsammans med diagrammet används blocket Generator_delay som inväntar att diagrammet ska vara redo för start samt fördröjer starten om så är inställt. För att omstart ska kunna ske måste först "Output_reset" bli "TRUE" för att återställa signalen. Startsignalen från blocket Generator_delay går till "RESET" i GEN-blocket så att signalen ska börja där den valda signalformen börjar. Det går dessutom att starta signalgeneratorn genom att hålla "SGEN_Active" signalen "TRUE". Eftersom signalen från signalgeneratorn sänder på intervallet ±2048 och de analoga utportarna ligger i intervallet 0-4095 finns även en offsetdel i detta program där en fast nivå läggs till på signalen. Det sista steget innan signalen sänds ut är filtreringen där signalen begränsas inom det möjliga intervallet så om användaren ställer signalen utanför övre eller undre gränsvärdet kapas signalen där. För mer information kring de variabler som används se Bilaga N.

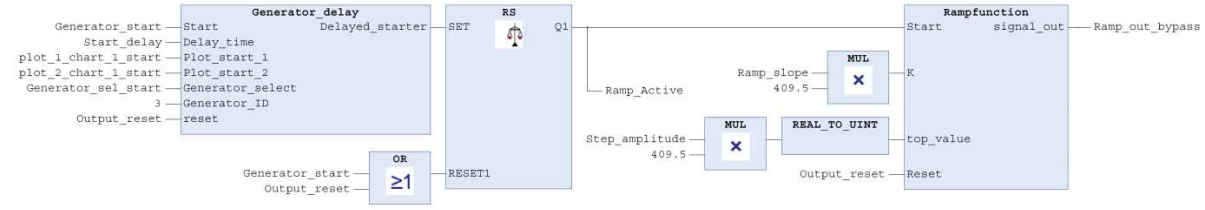


Figur 8.1 - Illustration av nätverket som innehåller signalgeneratorn.

Det andra nätverket innehåller alla delar för steggeneratorn vilket kan beskådas i Figur 8.2. Kärnan i denna programdel är blocket Stepfunction där inställning av steget sker med "Step_amplitude" vilket skalas om från 0-10 V till intervallet 0-4095. Denna generator är försedd med blocket Generator_delay för att kunna starta generatorn vid önskat tillfälle efter diagrammet har startat. För mer information kring de variabler som används se Bilaga N.



Figur 8.2 - Illustration av nätverket som innehåller steggeneratorn.



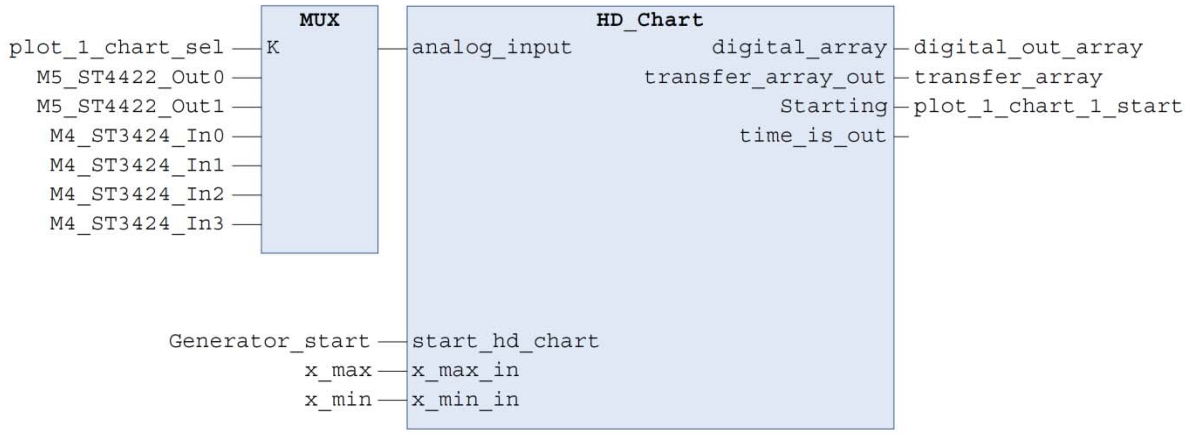
Figur 8.3 - Illustration av nätverket som innehåller rampgeneratorn.

Det tredje nätverket i detta PLC-program innehåller alla delar för rampgeneratorn vilket kan beskådas i Figur 8.3. Kärnan i denna programdel är blocket Rampfunction där rampens lutning ställs med "Ramp_slope" som har enheten volt per sekund. för att sätta slutnivån på rampen sätts "Step_amplitude" till den önskade nivån i volt. Rampgeneratorn startas med blocket Generator_delay vilket innebär att även diagrammet spelar in händelserna. För mer information kring de variabler som används se Bilaga N.

8.2 Chart_Plot_Complete

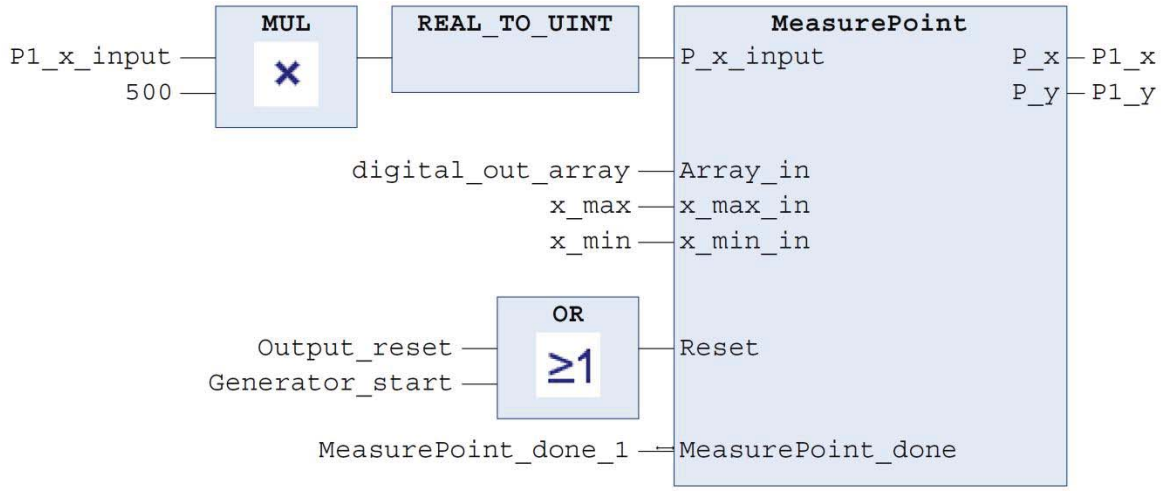
I programmet Chart_Plot_Complete sparas mätvärden från vald kanal och formar en visningskurva för HMI-delen. Programmet innehåller även två mätpunkter som också jämförs med varandra i två dimensioner. Av följande program finns två identiskt uppbyggda versioner där de behandlar varsin kurva.

Ett av nätverken innehåller det egenkonstruerade blocket HD_Chart, för val av analog insignal till blocket anslöts en MUX (multiplexer) där de olika analoga signalerna finns att välja mellan. När "Generator_start" är "TRUE" så startas blocket, men nedsparning av mätvärden börjar inte förrän "plot_1_chart_1_start" blir "TRUE". "x_max" och "x_min" är värden vilket kommer ifrån HMI-delen där inställning av visningsintervallet ställs in. Alla mätvärden som tas görs tillgängliga via "digital_out_array" och "transfer_array" är en mindre nedskalad array som används av HMI-delen. Detta nätverk finns i Figur 8.4 och de variabler som används se Bilaga N.



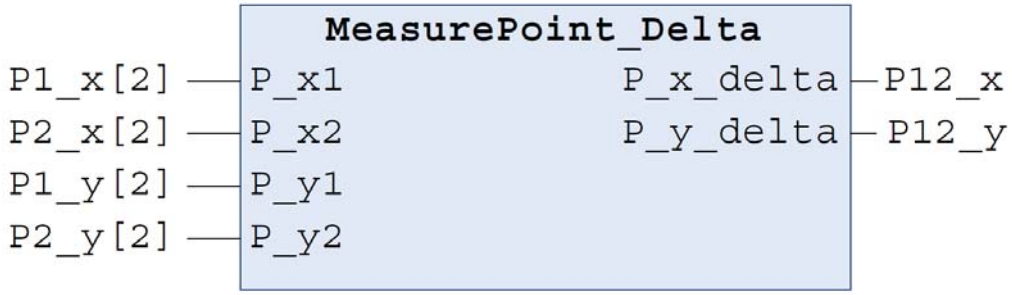
Figur 8.4 - Illustration av nätverket som innehåller HD_Chart.

Nätverket vilket innehåller blocket MeasurePoint skapar ett plustecken i form av två arrayer på den begärda positionen på X-axeln i analysdiagrammet. "P1_x_input" skalas upp för att motsvara den punkt vilket är ekvivalent i "digital_out_array" vilket lagrar mätpunkterna. "x_max" och "x_min" används för att skala plustecknet i X-led samt för att kunna avaktivera tecknet när det hamnar utanför intervallet. "Reset" plockar bort mätpunkten vid antingen återställning från HMI-delen eller "Generator_start" då analysdiagrammet också återställs. "MeasurePoint_done_1" används av HMI-delen för att kunna räkna om värdet då det inte går att överföra direkt mellan softPLC-delen och HMI-delen, värdet inverteras vid klarsignal. Utsignalerna från MeasurePoint-blocket används i HMI-delen där de formar ett plustecken samt innehåller det exakta mätvärdet i element två. Nätverket kan beskådas i Figur 8.5 och programmet innehåller två identiska nätverk av denna sort. De variabler som används i nätverket finns i Bilaga N.



Figur 8.5 - Illustration av nätverket som innehåller MeasurePoint.

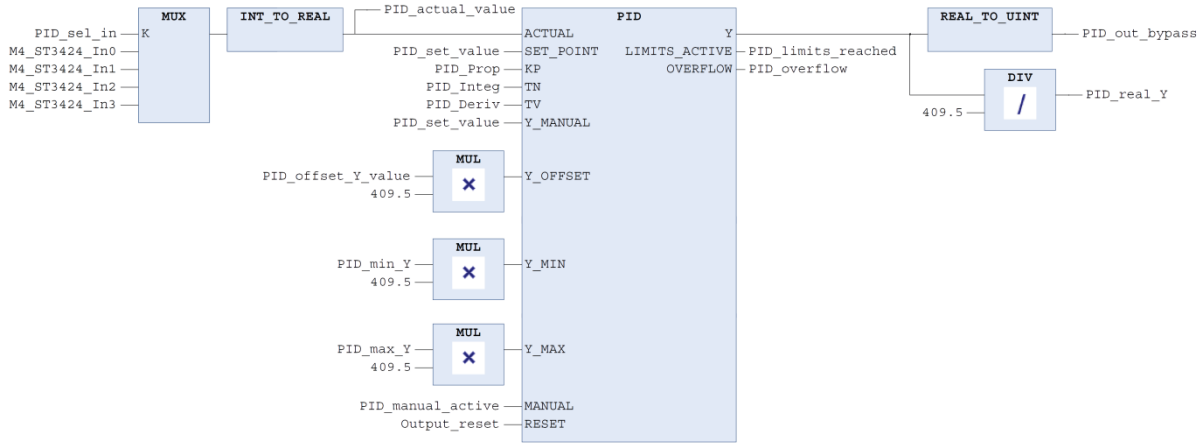
Nätverket som innehåller blocket MeasurePoint_Delta behöver inga andra block för att utföra dess uppgift. MeasurePoint_Delta använder element nummer två i de arrayer som de två MeasurePoint-blocken skapar då detta motsvarar det exakta värdet i plustecknet. Utsignalerna är den uträknade skillnaden mellan punkterna i X-led samt Y-led. Nätverket kan ses i Figur 8.6 och signalerna finns beskrivna i Bilaga N.



Figur 8.6 - Illustration av nätverket som innehåller MeasurePoint_Delta.

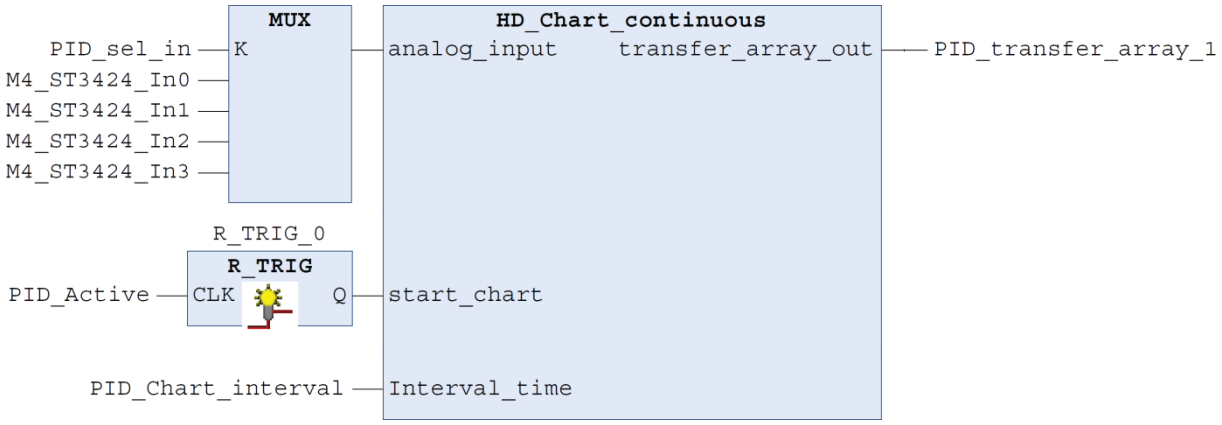
8.3 PID_regulator

Programmet PID_regulator innehåller en PID-regulator samt en funktion vilket sparar de senaste värdena från regulatorns styrvärde och processens ärvärde. Ett nätverk innehåller blocket PID vilket är kärnan i PID-regulatorn, enligt Figur 8.7. En MUX är anslutet där val av var ärsignalen kommer ifrån kan göras, därefter följer konverteringar från intervallet 0-10 V till det intervall som regulatorn arbetar med, vilket är 0-4095. Detta sker på signalerna "PID_offset_Y_value", "PID_min_Y", "PID_max_Y" och "PID_real_Y" dessa och de signaler vilket kan ses i Figur 8.7 finns beskrivna i Bilaga N.

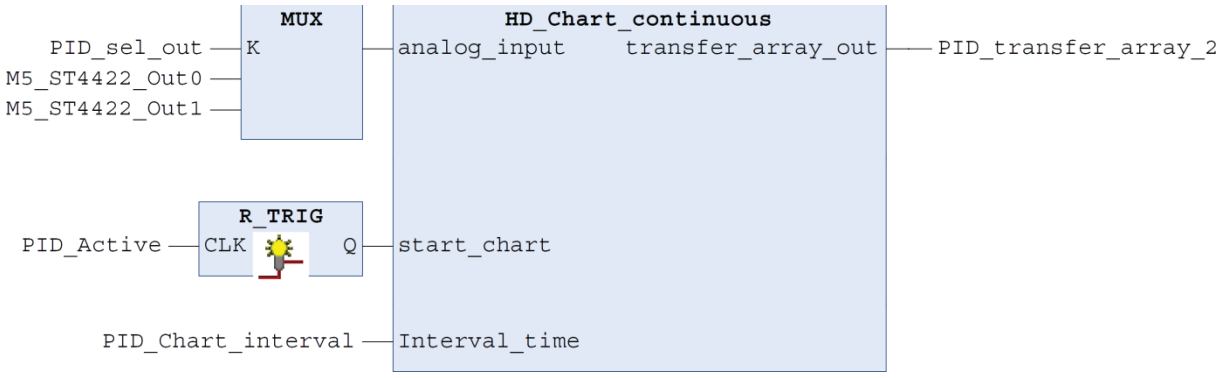


Figur 8.7 - Illustration av nätverket som innehåller PID.

Nästa nätverk innehåller i grunden blocket HD_Chart_continuous där senaste mätvärde sparar över det äldsta för att visa den senaste tidens händelser. Val av ingångskanal sker med "PID_sel_in" till en MUX där samma kanal väljs som PID-blockets ärsignal väljes enligt Figur 8.8. Det finns också ett likadant nätverk i programmet vilket istället har "PID_sel_out" som valsignal då detta används till styrsignalen som skickas från PID-blocket enligt Figur 8.9. "PID_Active" sätts igenom ett R_TRIG-block då signalen ligger hög under drift men blocket HD_Chart_continuous kör återställning i början som då inte kan brytas. Signalbeskrivningar finns i Bilaga N.



Figur 8.8 - Illustration av nätverket som innehåller HD_Chart_continuous.



Figur 8.9 - Illustration av nätverket som innehåller HD_Chart_continuous.

8.4 Source_Drain_select

För att lösa ett problem som uppstod under konstruktionen skapades denna programkod. Problemet var att de nu ovannämnda delarna måste samsas om utgångarna, vilket från början resulterade i att de olika programmen och blocken skrev över varandras värden. Lösningen är enkel men svår att implementera på större konstruktioner. Varje block får en aktiveringssignal. Beroende på vilken kanal det ska sändas på finns ytterligare ett val, se koden i Bilaga O samt variabler som används i Bilaga N.

9 IX DEVELOPER – HMI

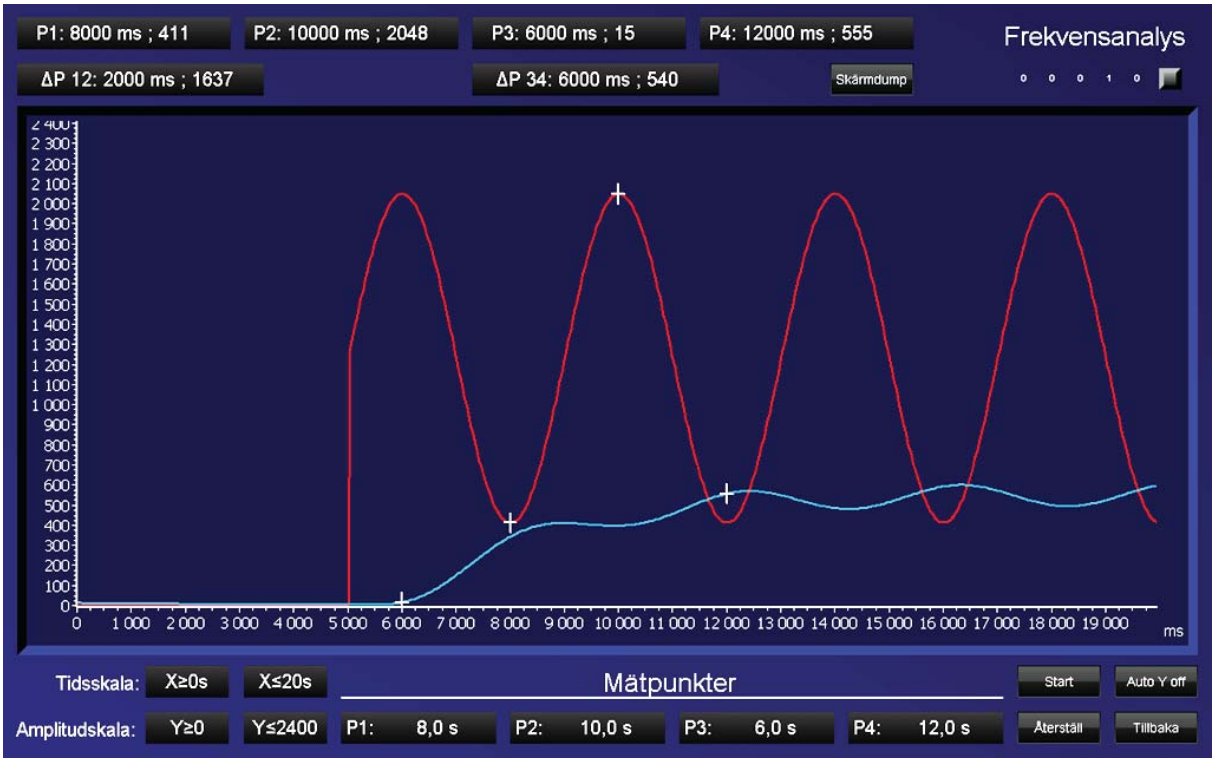
HMI-delen är uppdelad i tre delar men under arbetsgången gjordes alla tre delar parallellt. Dessa tre delar är utformningen av objekt samt färgval, kopplingen med softPLC-delen och programmering i iX Developer.

9.1 Design

Design är ett väldigt djupt område och det finns många olika teorier om hur grafiska bilder kan konstrueras. I detta projekt har dock inte designen varit huvudsyftet utan mer kraft har tilldelats funktionaliteten i de grafiska bilderna. Idén med designen är att den ska hålla sig enkel och tidlös med fokus på optimering av utrymmet men samtidigt att behålla utseendet enhetligt (6). I Figur 9.1 kan den övergripande designen skådas och viss disposition.

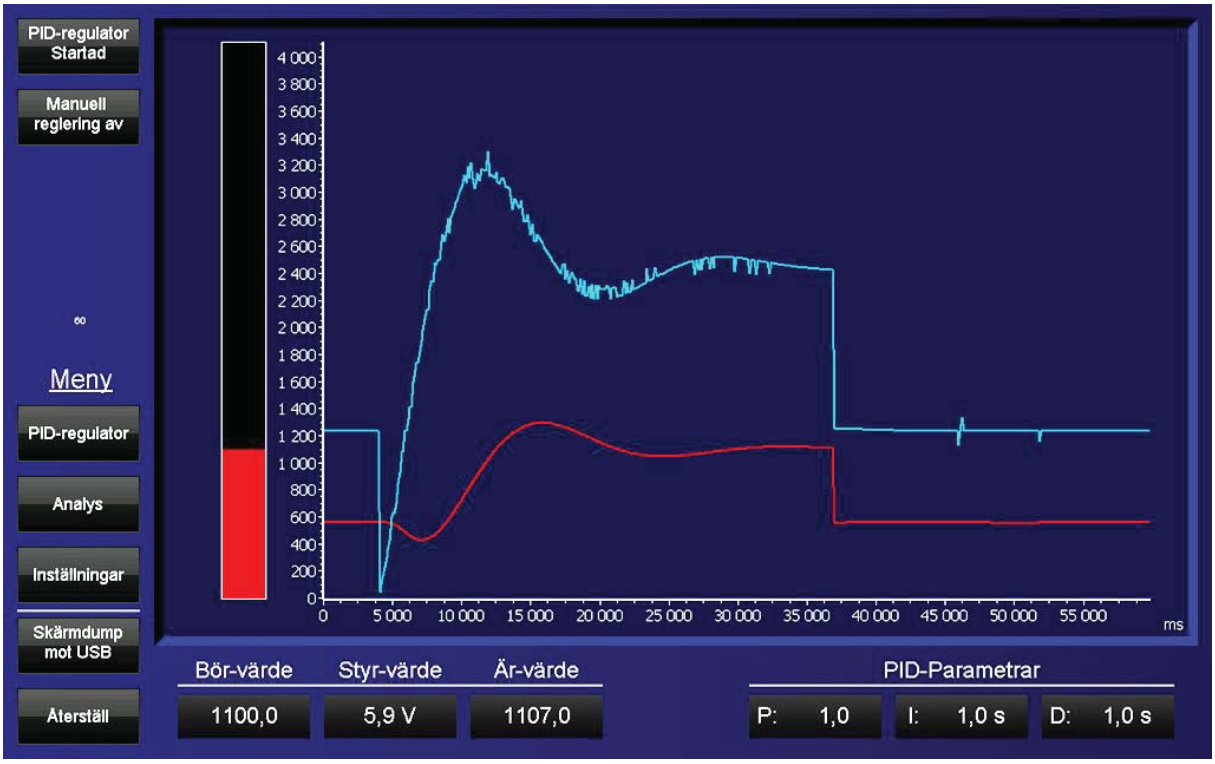
Figur 9.1 - Utseendet av sidan för inställningar i HMI-delen.

Eftersom panelen är förhållandevis liten har optimeringen varit den tyngsta delen i designen då diagrammen har störst fördel med så stor yta som möjligt vilket har gjort att knappar och nummerfält fått krympas så mycket att de inte tappar funktionaliteten i större utsträckning. På den sida där analysdiagrammet finns har även menyn plockats bort för att utnyttja utrymmet, se Figur 9.2.



Figur 9.2 - Utseendet av sidan för analys i HMI-delen.

Bakgrundsfärgen är mörkt blå som övergår till en något ljusare blå färg och alla knappar och nummerfält är svarta vilket övergår till en mörkgrå färg. All text är vit förutom varningstext som är röd. Den viktigaste informationen på alla sidor sitter innanför en ruta med en mörkare blå färg än bakgrundsfärgen så som i Figur 9.3 där sidan för PID-regulatorn kan skådas.



Figur 9.3 - Utseendet av sidan för PID regulatorn i HMI delen.

9.2 Interaktion med softPLC

De variabler som behövs importeras från CODESYS till iX Developers taglista. Dessa variabler används på följande sätt:

- Aktivera funktioner genom att trycka på en knapp som inverterar, eller håller signalen vid nedtryck och släpper signalen vid knappsläpp
- Rullgardinsmeny där olika val finns som sätter en variabel till ett motsvarande värde
- Inmatningsfält för att ge en variabel ett valfritt värde
- Utmatningsfält där ett värde presenteras från softPLC-delen
- Varningstext som visas när en variabel uppfyller ett krav
- Stapel som illustrerar värdet av en variabel

9.3 Analysdiagrammet - Script

För att kunna utföra en analys av en process är det nödvändigt att kunna se ett långt intervall med hög upplösning där mätningar kan utföras. I iX Developer finns två objekt som kan visa en graf. En av dem kallas "Trend Viewer" där signalen skriver över det äldsta värdet och grafen flyter åt vänster. Denna idé valdes bort då uppdateringstiden inte kunde ställas under en sekund. Därav konstruerades en egen grafritare där ett linjediagram i iX Developer istället användes genom att koppla denna mot arrayer vilka innehåller jämnt fördelade mätvärden av de tillkopplade källorna.

Ett försök av detta var att sända en array per kurva med 60000 element ifrån CODESYS till iX Developer som då innehöll alla mätvärden. iX Developer kan inte hantera detta på grund av en begränsning där arrayer endast kan ha typen INT med max 512 element. Istället skalades denna array ner till en annan array med 500 element av typen UINT16 då panelen dessutom inte behöver mer mätpunkter då upplösningen i bredd inte kan överstiga 1280 punkter. Detta matchades mot en array i X-led för att få punkterna att sprida ut sig jämnt. X-arrayen som visar värdet i diagrammet i tid då millisekunder skalades upp eftersom förloppstiden är fyra minuter vilket i millisekunder motsvarar 240000 där en UINT16 inte klarar mer än 65535. För att skala upp detta värde behöver en beräkning utföras i HMI-delen. Detta görs enklast i ett script som multiplicerar den array som kommer från softPLC-delen med en konstant och sparar ner de nya värdena lokalt i en ny array där elementen har typen UINT32. För att detta ska fungera måste arrayen från softPLC-delen användas i ett objekt på samma sida där den lokala arrayen av typen UINT32 ska användas. Därefter för att HMI-delen inte hela tiden ska behöva använda beräkningskraft på detta så används en signal som ändras när uppdatering av den lokala arrayen ska genomföras. Denna signal är kopplad in i scriptet och är den signal som aktiverar scriptet, se scriptet i Bilaga P. Detta gäller även för PID-regulatorns diagram, se Bilaga Q.

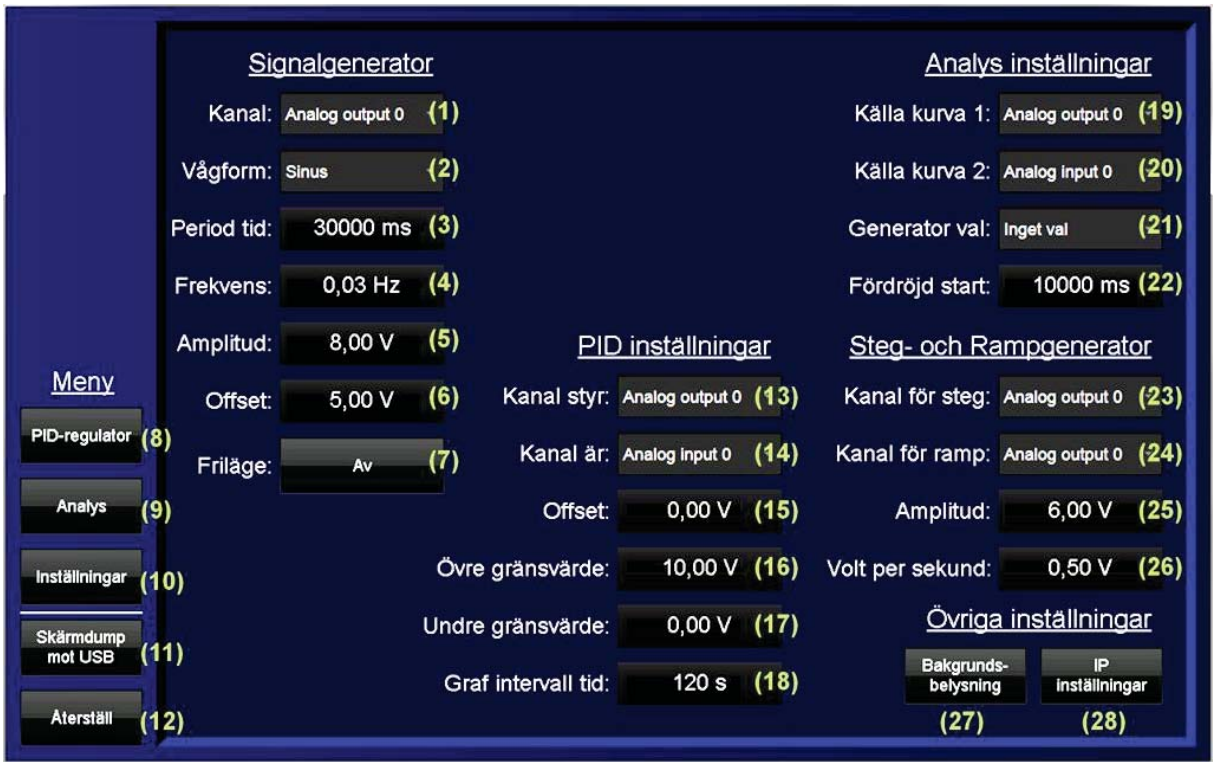
För att enkelt och exakt kunna utföra mätningar i diagrammet finns två mätpunkter per kurva. Dessa visas i diagrammet i form av ett plustecken på den punkt som Y-värdet önskas avläsas. Mätpunkterna flyttas genom att välja inmatningsvärdet på X-axeln vilket den mätpunkten ska hamna på. För att punkterna ska kunna flyttas i realtid måste dessa också uppdateras med en

signal genom scriptet eftersom de ritas upp efter samma X-axel som går genom scriptet. Se scriptet i Bilaga P.

10 HANDHAVANDEMANUAL

Här följer beskrivning av alla funktioner och handhavandehjälp för HMI-delen av systemet, alla sidor finns med som figurer.

Sidan Inställningar



Utseendet på inställningssidan med numrering av alla funktioner.

1) Val av vilken analog kanal signalen från signalgeneratorn ska sändas ut på.
2) Val av vågform för signalgeneratorn.
3) Möjlighet att ställa in periodtiden för vågformen i millisekunder.
4) Möjlighet att ställa in frekvensen för vågformen i Hertz.
5) Val av topp-till-topp värdet av signalen för signalgeneratorn.
6) Förskjutningsspänning för signalen, höjer signalen med en DC nivå för signalgeneratorn.
7) Möjlighet att kunna starta enbart signalgeneratorn och få signalen utsänd på vald kanal.
8) Knapp som tar användaren till sidan som innehåller PID-regulatorn.
9) Knapp som tar användaren till sidan som innehåller analysdiagrammet.
10) Knapp som tar användaren till sidan som innehåller Inställningarna.
11) Knapp som låter användaren ta en skärmdump som sparas på anslutet USB-minne bakom panelens svarta lucka.
12) Återställer och stänger av generatorer, PID-regulatorn, mätpunkter samt analysdiagrammet.
13) Val av vilken analog kanal styrsignalen från PID-regulatorn ska sändas ut på.
14) Val av vilken analog kanal ärsignalen till PID-regulatorn ska tas emot ifrån.
15) Offset nivå av PID-regulatorns styrsignal.

16) Övre gränsvärde för PID-regulatorn som regulatorns styrsignal inte kan överskrida.
17) Undre gränsvärde för PID-regulatorn som regulatorns styrsignal inte kan underskrida.
18) Den totala tiden som hålls kvar i diagrammet som visar PID-regulatorns styr- och ärvärde i sekunder, desto längre tid som ställs in desto sämre upplösning, kan inte ställas under två sekunder och endast jämnt antal sekunder.
19) Analog kanal som ansluts till första kurvan i analysdiagrammet, kurvan är röd.
20) Analog kanal som ansluts till andra kurvan i analysdiagrammet, kurvan är ljusblå.
21) Val av generator som ska aktiveras med start av analysdiagrammet.
22) Fördröjning av starten av generatorn så händelser före generatorns påverkan kan undersökas.
23) Val av vilken analog kanal signalen från steggeneratorn ska sändas ut på.
24) Val av vilken analog kanal signalen från rampgeneratorn ska sändas ut på.
25) Amplituden som steggeneratorn och rampgeneratorn hamnar på efter start av analysdiagrammet (startas på analyssidan).
26) Stigningshastighet för rampgeneratorn i volt per sekund.
27) Möjlighet att ändra ljusinställningar på panelen.
28) Möjlighet att ändra IP-inställningar för panelens två LAN-portar.

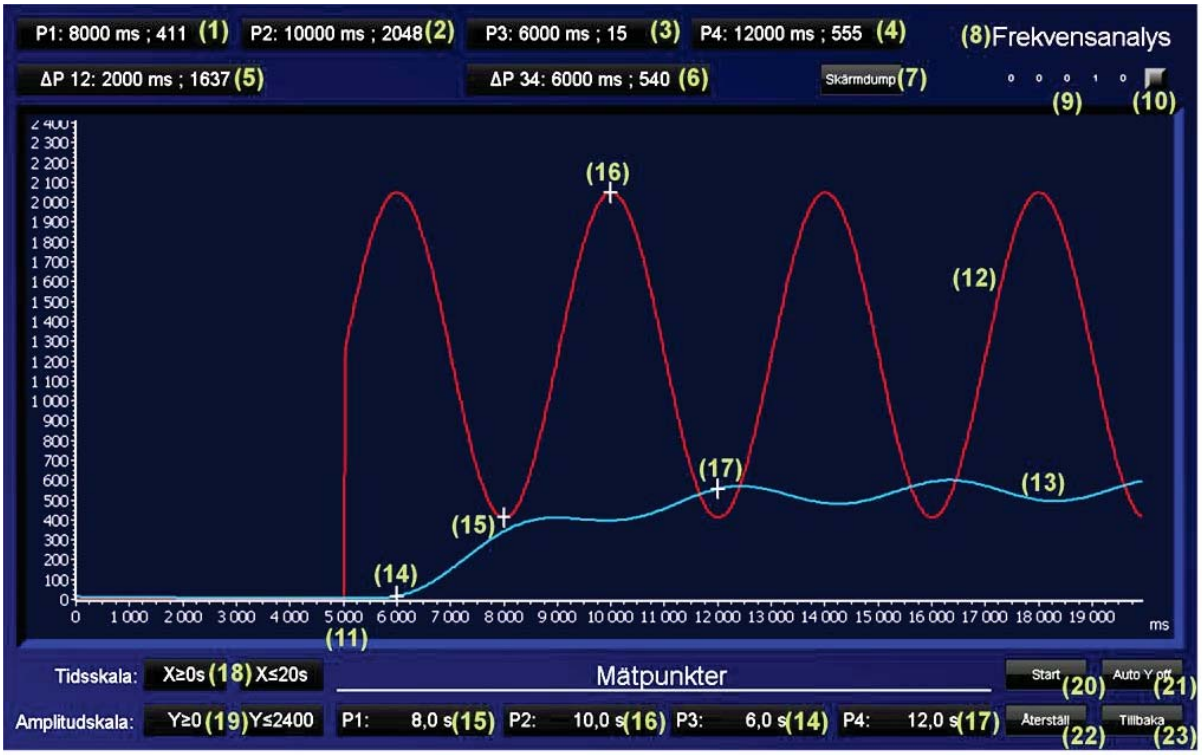
Sidan PID regulator

Utseendet av regulatorsidan med numrering av alla funktioner.

1) Knapp som släpper fram PID regulatorns styrsignal till processen.
2) Knapp som flyttar börvärdet direkt till styrvärdet utan att regulatorn kan påverka, intervallet på börvärdet 0-4095, motsvarar intervallet vid styrvärdet 0-10 V.

3) Text som syns om PID regulatorn integralverkan får overflow, annars syns inte denna text.
4) Text som syns om PID regulatorn når något av det undre eller övre inställda gränsvärdet, annars syns inte denna text.
5) Indikator där inställd intervalltid syns, används för att kunna uppdatera tidsskalan i diagrammet.
6) Stapel som indikerar börvärdet.
7) Möjligheten till att ställa in börvärdet i intervallet 0-4095 som motsvarar 0-10 V på den analoga porten.
8) Indikation av vad styrvärdet är just nu i skalan 0-10 V.
9) Indikation av vad ärvärdet är just nu i skalan 0-4095 som på den analoga porten motsvarar 0-10 V.
10) Inställbart värde för P verkan för regulatorn.
11) Inställbart värde för I verkan för regulatorn.
12) Inställbart värde för D verkan för regulatorn.
13) Positionen där graferna är just nu, kurvan till höger ritas över då värdena i detta exempel är 60 sekunder gamla.
14) Den ljusblåa kurvan är styrvärdet.
15) Den röda kurvan är ärvärdet.

Sidan analys



Utseendet av analyssidan med numrering av alla funktioner.

1) Visningsvärde för mätpunkt P1 tillhörande den röda kurvan som är kopplad till källa 1 i inställningar, första värdet är tiden i millisekunder, andra värdet är nivån vid den punkten i skalan 0-4095 som motsvarar 0-10 V på givaren.
2) Visningsvärde för mätpunkt P2 tillhörande den röda kurvan som är kopplad till källa 1 i inställningar, första värdet är tiden i millisekunder, andra värdet är nivån vid den punkten i skalan 0-4095 som motsvarar 0-10 V på givaren.
3) Visningsvärde för mätpunkt P3 tillhörande den ljusblåa kurvan som är kopplad till källa 2 i inställningar, första värdet är tiden i millisekunder, andra värdet är nivån vid den punkten i skalan 0-4095 som motsvarar 0-10 V på givaren.
4) Visningsvärde för mätpunkt P4 tillhörande den ljusblåa kurvan som är kopplad till källa 2 i inställningar, första värdet är tiden i millisekunder, andra värdet är nivån vid den punkten i skalan 0-4095 som motsvarar 0-10 V på givaren.
5) Visningsvärde för skillnaden mellan mätpunkt P1 och P2 tillhörande den röda kurvan som är kopplad till källa 1 i inställningar, första värdet är skillnaden i tid mellan mätpunkterna i millisekunder, andra värdet är nivåskillnaden mellan mätpunkterna i skalan 0-4095 som motsvarar 0-10 V på givaren.
6) Visningsvärde för skillnaden mellan mätpunkt P3 och P4 tillhörande den ljusblåa kurvan som är kopplad till källa 2 i inställningar, första värdet är skillnaden i tid mellan mätpunkterna i millisekunder, andra värdet är nivåskillnaden mellan mätpunkterna i skalan 0-4095 som motsvarar 0-10 V på givaren.
7) Knapp som sparar en skärmdump till ett inkopplat USB-minne bakom panelens svarta lucka.
8) Indikerar vilken typ av analys som är vald.
9) Dessa fem värden används för att aktivera ett skript som uppdaterar X-axeln och mätpunkterna separat. Användaren behöver inte ägna sig åt dessa.
10) Ett dolt diagram som innehåller de värden som räknas om i scriptet. Användaren behöver inte ägna sig åt dessa.
11) En fördröjd start, startar generatorn fem sekunder efter diagrammet börjat spara värden.
12) Röd kurva tillhör källa 1 i inställningar och har mätpunkterna P1 och P2 anslutna.
13) Ljusblå kurva tillhör källa 2 i inställningar och har mätpunkterna P3 och P4 anslutna.
14) Mätpunkt P3 placeras på den ljusblåa kurvan med källa 2 i inställningar.
15) Mätpunkt P1 placeras på den röda kurvan med källa 1 i inställningar.
16) Mätpunkt P2 placeras på den röda kurvan med källa 1 i inställningar.
17) Mätpunkt P4 placeras på den ljusblåa kurvan med källa 2 i inställningar.
18) Tidsskalan ställs in här med undre och övre värde, det vänstra är det lägsta och det högra är det högsta värdet som visas på tidsskalan.
19) Amplitudskalan ställs in manuellt då knappen "Auto Y" (21) indikerar "off" det vänstra värdet är det lägsta värdet och det högra värdet är det högsta värdet som visas.
20) Startar diagrammet och vald generator efter fördröjningstiden har löpt ut.
21) Val där amplitudskalan kan ställas in automatiskt beroende på vilken kurva som har högsta toppvärdet eller manuellt med hjälp av (19).
22) Knapp som återställer signaler ner till noll och förbereder nystart.
23) Knapp som tar användaren tillbaka till sidan inställningar.

11 RESULTAT

Under arbetets gång har funktioner för softPLC-delen konstruerats. HMI-delen har designats allt eftersom funktionerna i softPLC-delen har skapats, för att kunna testa dessa. En del av projektet är generatorerna som används mot analysdiagrammet, till detta diagram är de flesta delar egenkonstruerade i softPLC-delen och utnyttjar diagramfönstret i HMI-delen för att kunna presentera kurvorna. Enligt målen har en analysdel skapats med möjlighet till steg-, ramp- och frekvenssvarsanalys vilket kräver visning av två signaler i samma diagram. Dessa två kurvor är försedda med två mätpunkter vardera som dessutom beräknar skillnaden mellan punkterna för enkla mätningar. För att utnyttja panelens fulla storlek har sidan för analysdiagrammet specialutformats så diagrammet kan utnyttja mer av skärmens yta.

Den andra delen av systemet består av en PID-regulator som är kopplad till ett diagram. Regulatorns styrsignal och processens ärvärde visas under en inställbar intervalltid där kurvorna efter utlupen tid börjar skriva över de äldsta värdena på kurvan liknande en EKG-apparat med bildskärm.

HMI-delen är uppbyggd med enkel design utan många detaljer och neutrala färger som inte irriterar ögonen då starka färger avger mycket ljus som retar ögonen i längden enligt egna erfarenheter. Efter mycket testande har utseendet successivt tagits fram för att hålla funktionaliteten hög.

12 DISKUSSION

Efter att systemet blev färdigställt så uppfylldes målen som sattes och vi är nöjda med resultatet då vi med stort intresse har utfört detta projekt. De delar som är bra är bland annat designen då vi lyckats med att få objekten kompakta och enkla samt utnyttja skärmens yta till stor del. Vi valde också att inte ha flashiga utseenden för de olika objekten i HMI-delen utan valde stilar som är gångbara (6).

Vi märkte under projektets gång att utrustningens hårdvara inte riktigt är lämpad för denna typ av uppgift då beräkningstiderna kan vara något i det längsta laget, det är då den grafiska biten som hamnar mest på efterkälken. Ibland kan beräkningar behöva göras direkt i HMI-delen som vi gör på vissa ställen. Det vi upptäckte här var att dessa beräkningar verkar vara betydligt tyngre att göra än om liknande beräkning skulle göras i softPLC-delen. För att göra beräkningar i HMI-delen måste ett script skrivas med programmeringsspråket C# som är ett språk som vi tidigare inte har använt, vilket tog mycket tid från oss för att sätta sig in i språket.

Den bakomliggande PLC-koden fungerar bra och de viktigaste funktionerna finns med och de har en tydlig struktur. Vi har satt upp det yttre skalet i FBD och gjort många egna funktionsblock där vi för det mesta har skrivit i ST. Detta är en fördel under konstruktionsarbetet eftersom man enkelt kan ändra på funktionsblock utan att påverka det yttre skalet eller om man använder samma kod på flera ställen samt att koden blir översiktlig. Utifrån att ha arbetat med CODESYS under projektets gång så uppfattas det som ett kraftfullt och stabilt program med mycket bra funktioner och användarvänlighet.

Projektet är tidsbegränsat vilket har lett till att vissa önskvärda men inte nödvändiga funktioner har utelämnats då tiden löper ut. När projektet började löpa mot sitt slut var en funktion för att kunna justera in värdena från de analoga portarna till diagrammen i HMI-delen under konstruktion. Denna funktion förkastades dock eftersom lösningen av diagrammen som används i HMI-delen förlitar sig för mycket på programmeringen i HMI-delen och om denna funktion ska implementeras så skulle det krävdas ytterligare script. Huvudproblemet är att det endast går att överföra 16 bitars element av typen INT eller UINT i arrayform. Detta innebär att värdena inte kan skalas ner till vettiga nivåer och få med decimalerna i softPLC-delen. För att gå runt detta problem löste vi omskalningen med hjälp av script i HMI-delen för att kunna ha decimaler. Detta är möjligt att göra då lokala arrayer kan ha andra typer av element så som FLOAT. Tredje problemet satte dock stopp för denna lösning då visningen i diagrammet av en array med element av typen FLOAT inte gav tillräcklig upplösning mellan stegen. Ett sätt att lösa problemet på kan vara att istället lägga ett tomt diagram över det diagram som visar värdena där man endast använder skalorna och ändrar start och slutvärde på dessa och då inte behöver ändra på kurvans värden. Om denna typ av lösning fungerar skulle man kunna plocka bort scripten och låta softPLC-delen sköta alla beräkningar. Tyvärr kom denna idé för sent för att hinna realiseras.

Utrustningen vi har tagit fram för studenterna kan underlätta inlärningen och gör att de på ett lättare sätt kan ta del av reglertekniken då möjligheten till olika analyser nu betydligt mycket enklare kan utföras på kortare tid.

13 REFERENSER

1. csharpskolan.se, http://www.csharpskolan.se/ (Acc 2014-06-08)

2. Start-up: iX TxB SoftControl, Basic Setting (KI00328), http://www.beijer.se/ (Acc 2014-06-08)

3. iX TxB SoftControl, Device Description, Beijer Electronics, http://www.beijer.se/ (Acc 2014-06-08)

4. TxB_SoftControl_Setup_iX_2.0.msi, Beijer Electronics, http://www.beijer.se/ (Acc 2014-06-08)

5. TxB_SoftControl_Setup_iX_2.10.msi, Beijer Electronics, http://www.beijer.se/ (Acc 2014-06-08)

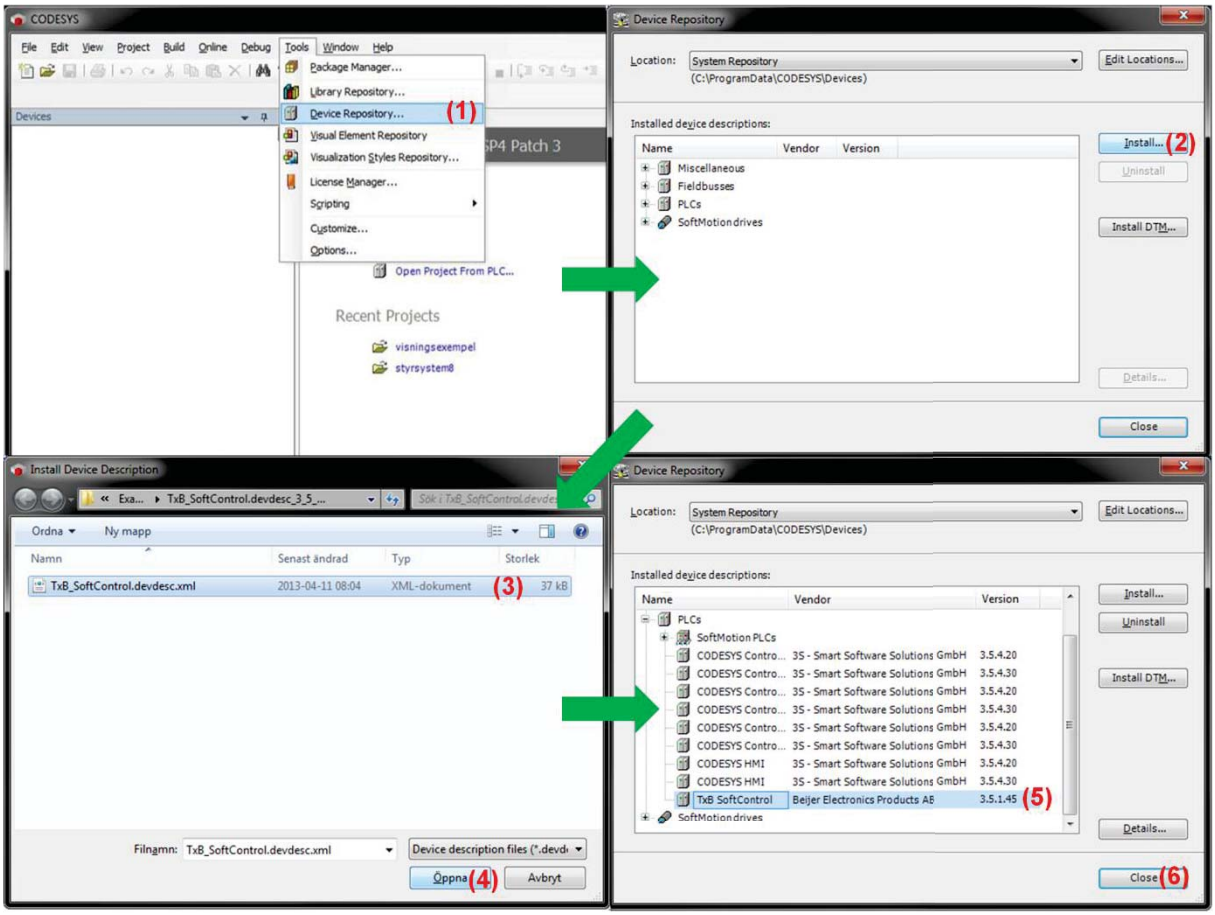
6. How to design a good HMI Display, http://www.hexatec.co.uk/ (Acc 2014-06-16)

7. Start-up: iX TxB SoftControl, NA-9186 EtherCAT (KI00329), http://www.beijer.se/ (Acc 2014-06-08)

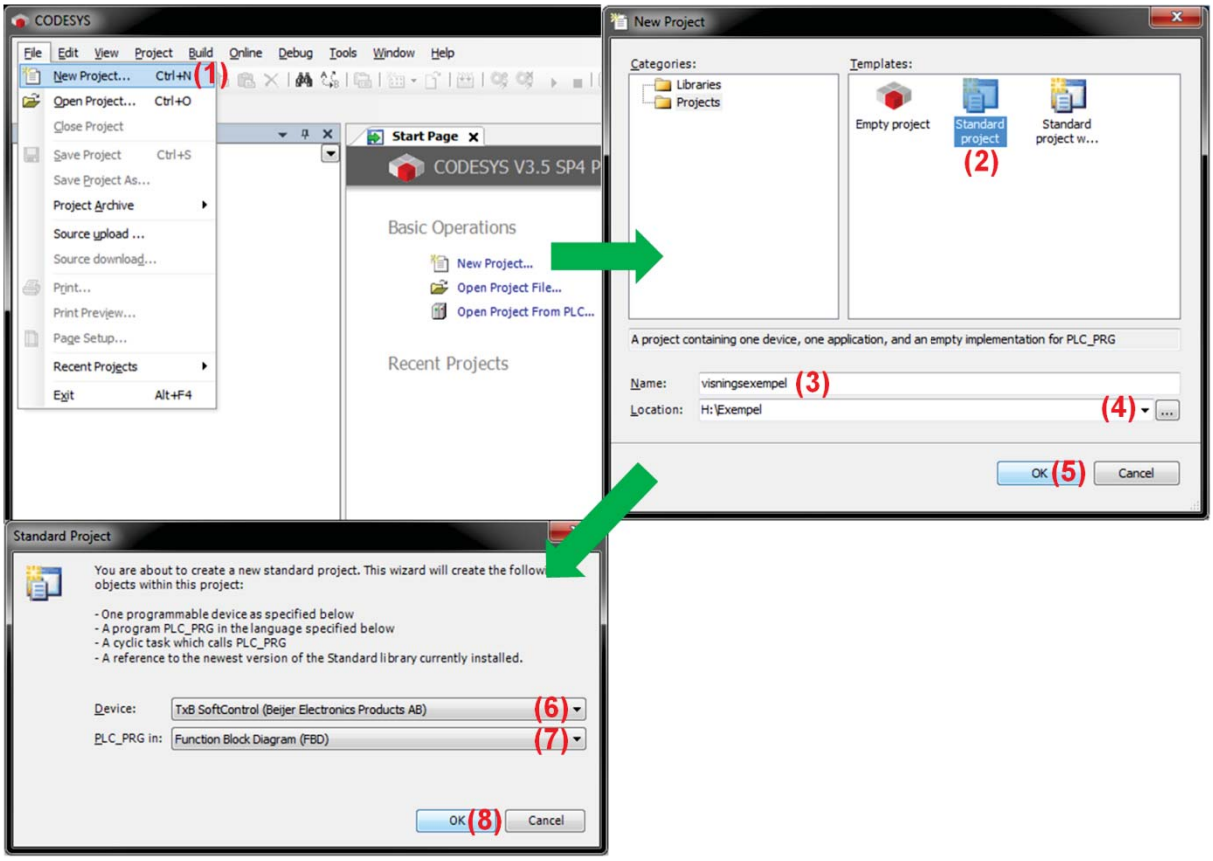
14 BILAGOR

Bilaga A Konfigurering av softPLC-delen (2)

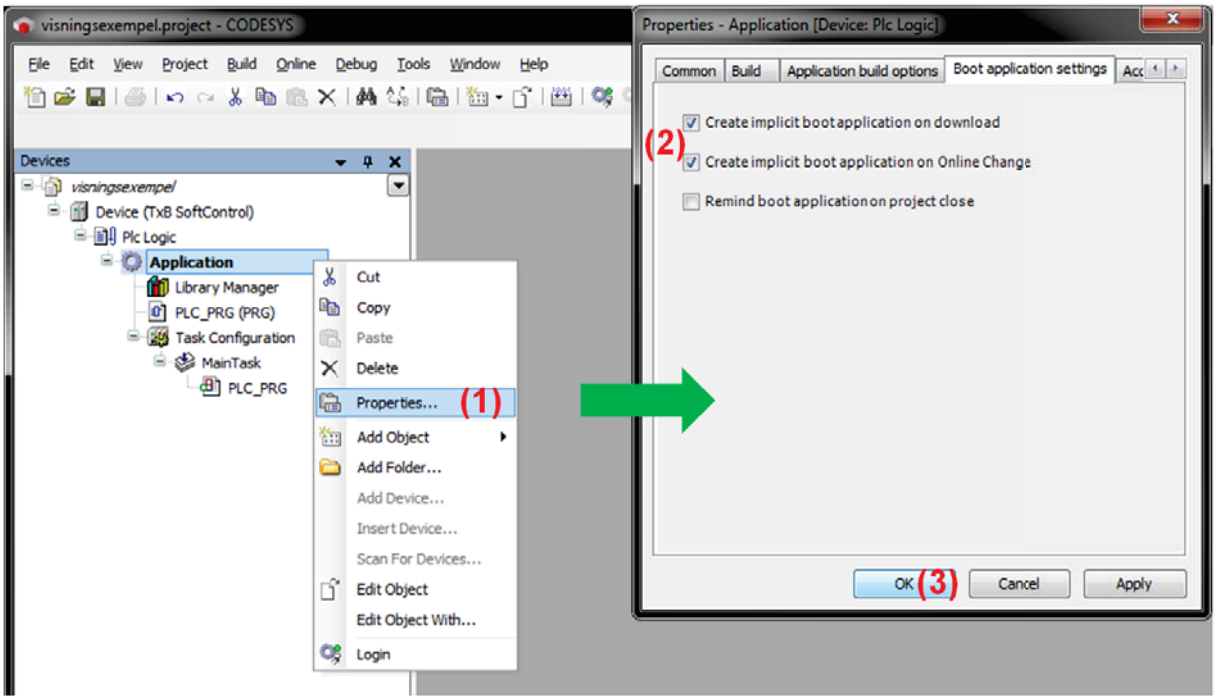
Lägg till konfigurationsfilen (TxB_SoftControl.devdesc.xml) (3) i CODESYS, vilket sköts i Device Repository.



Skapa ett projekt i CODESYS med standard bibliotek. Välj därefter "Device" för vilken typ av enhet programmet ska köras på som här valdes till "TxB SoftControl (Beijer Electronics Products AB)".



Markera "Application" och högerklicka på denna samt öppna "Properties" och gå till fliken "Boot application" och se till att de två rutorna är ikryssade.



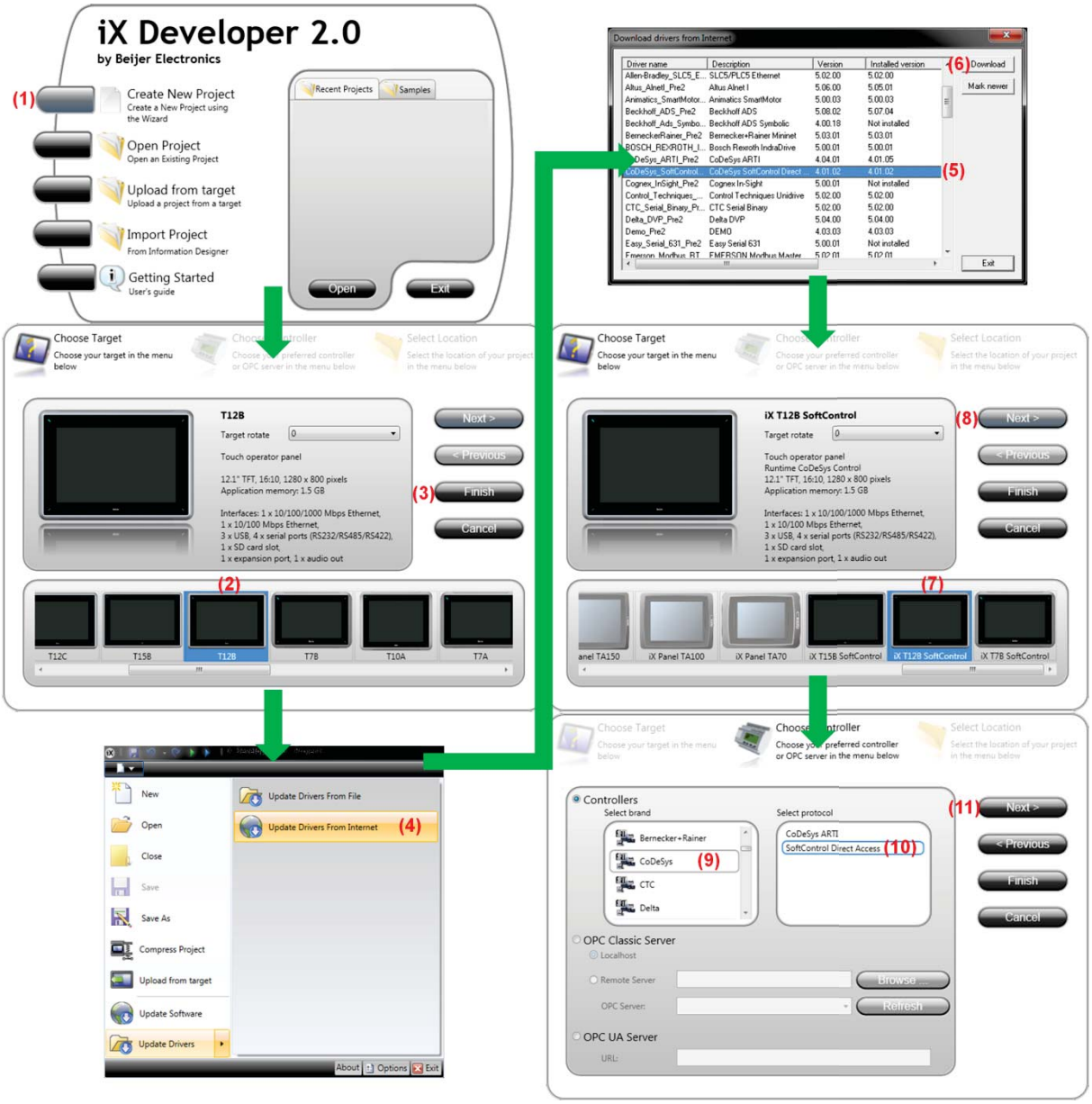
Bilaga B Konfigurering av HMI-delen (2)

Installera terminalen för SoftControl där följande fil används beroende av vilken version av iX Developer som ska användas.

iX Developer 2.0 SP1: "TxB_SoftControl_Setup_iX_2.0.msi" (4)

iX Developer 2.10: " TxB_SoftControl_Setup_iX_2.10.msi" (5)

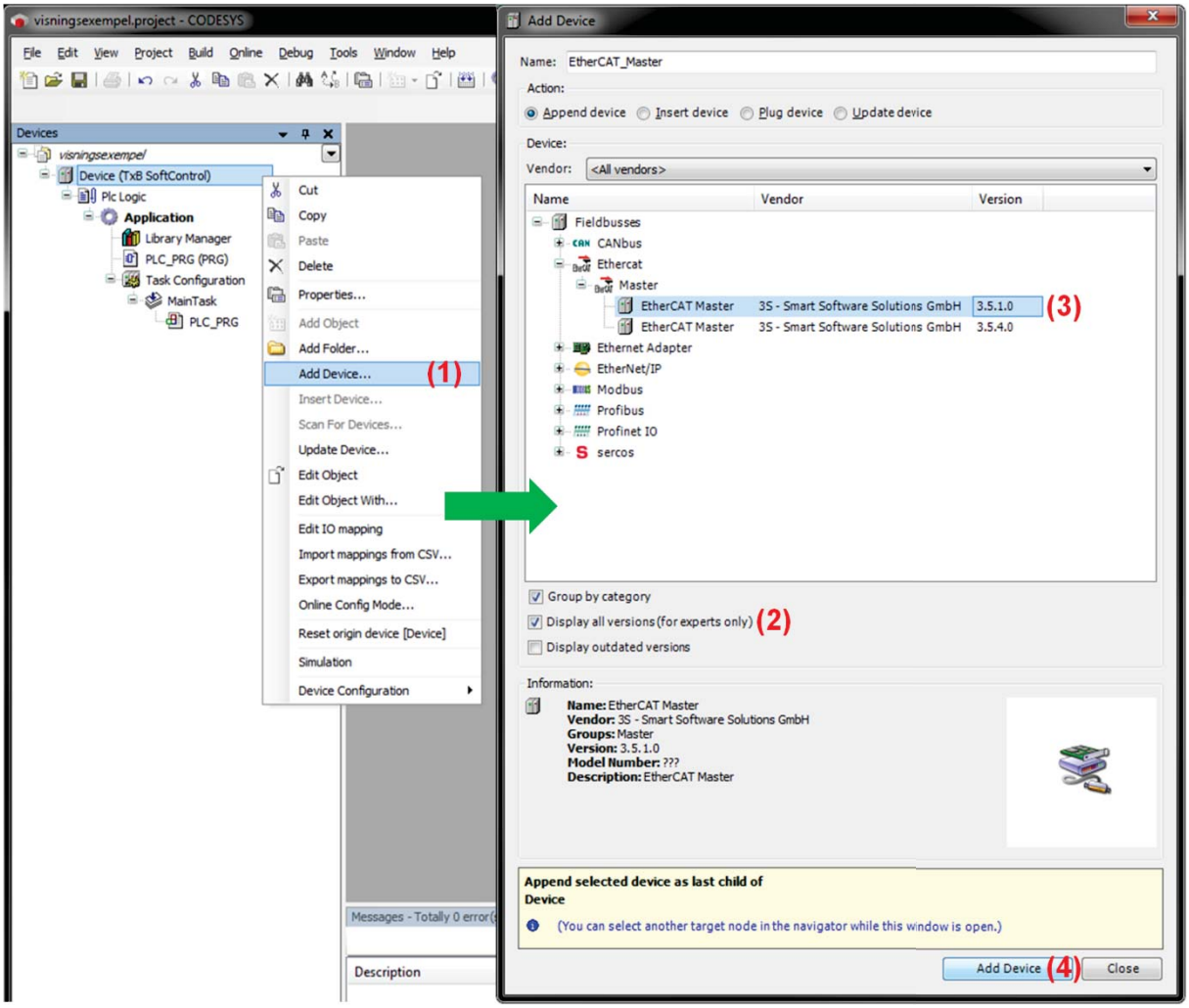
För iX Developer 2.0 användare: starta ett projekt i iX Developer som administratör utan att välja någon driver. Tryck på det tomma pappret uppe i vänstra hörnet i programfönstret. Välj "Update Drivers" och därefter "Update Drivers From Internet". Markera "CoDeSys_SoftControl_Direct_Access..." och klicka på "Download" och starta om iX Developer.



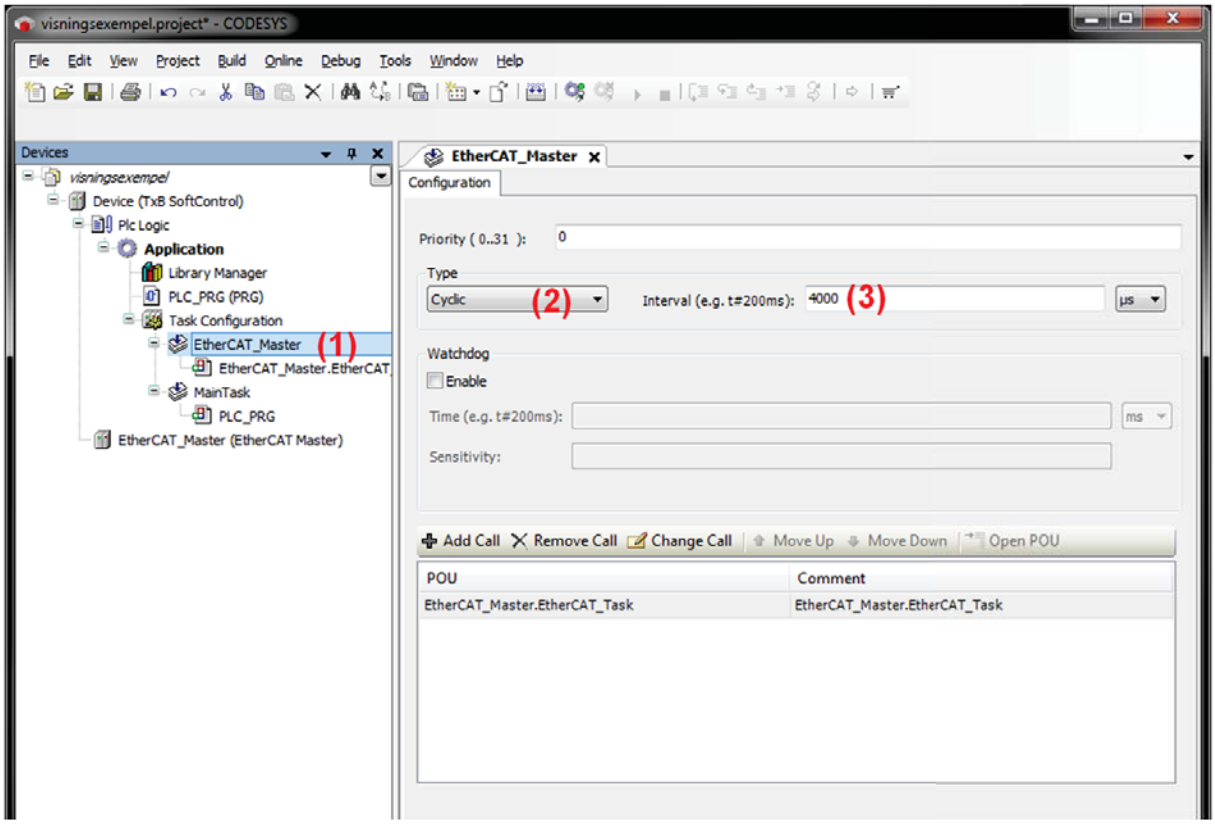
Bilaga C Konfigurering av decentraliserat I/O i CODESYS (7)

Öppna projektet som skapades i Bilaga A.

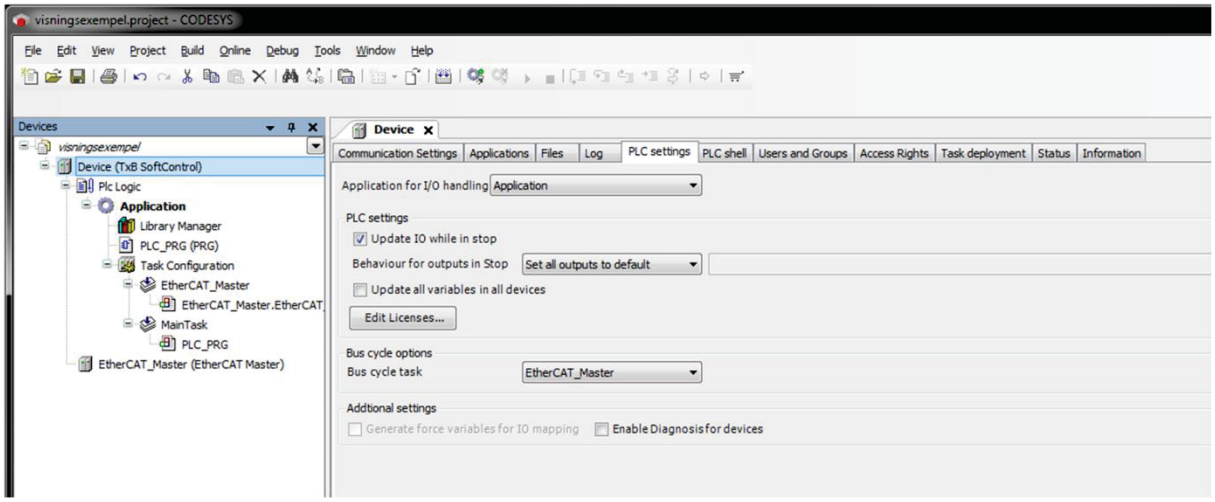
Markera och högerklicka på "Device", klicka på "Add Device" och "EtherCAT Master" med versionsnummer 3.5.1.0 ("Display all versions" måste vara ifylld).



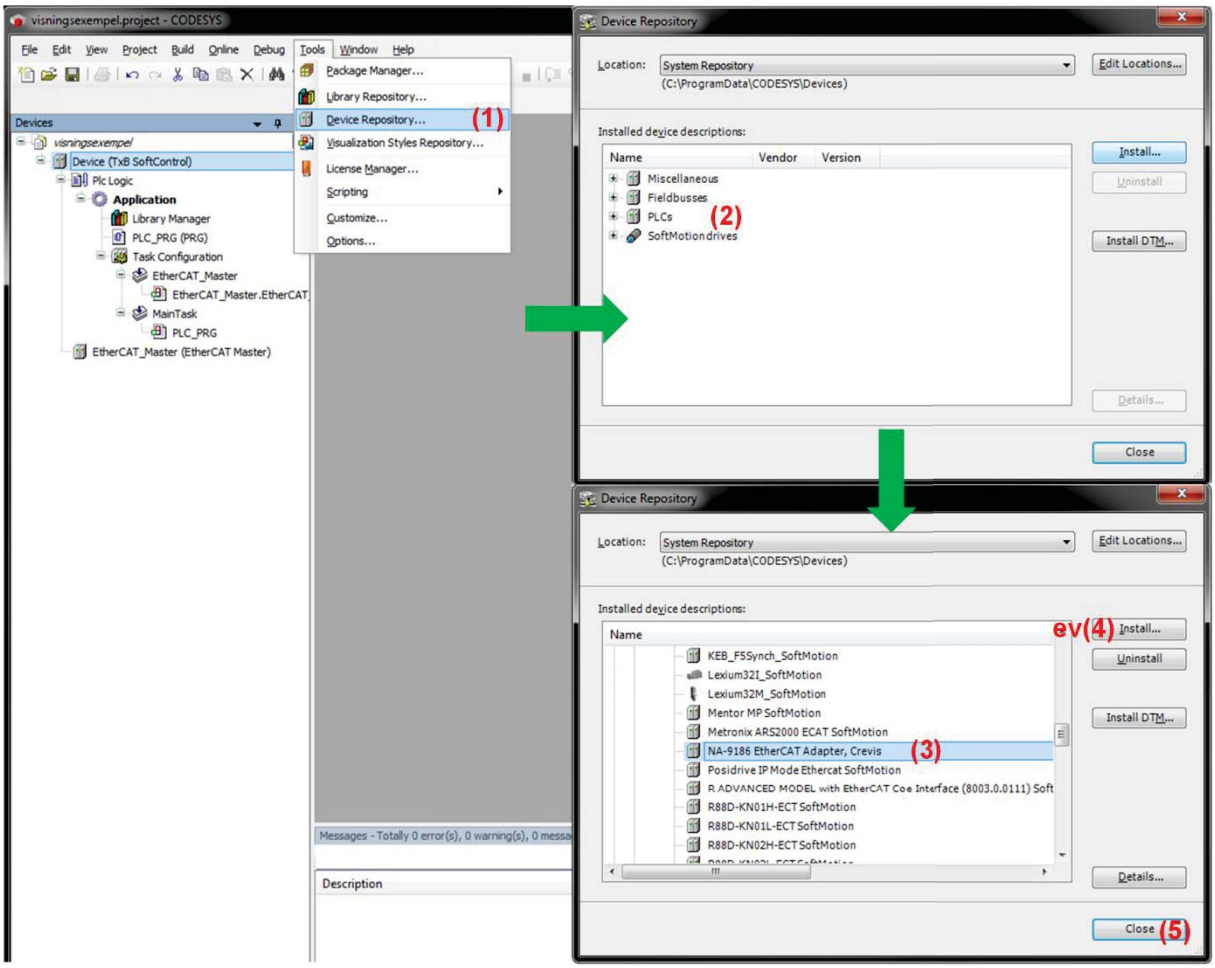
Dubbelklicka på "EtherCAT_Master" i "Task Configuration" och se till att "Type" står på "Cyclic och "Interval" står på 4000 µs.



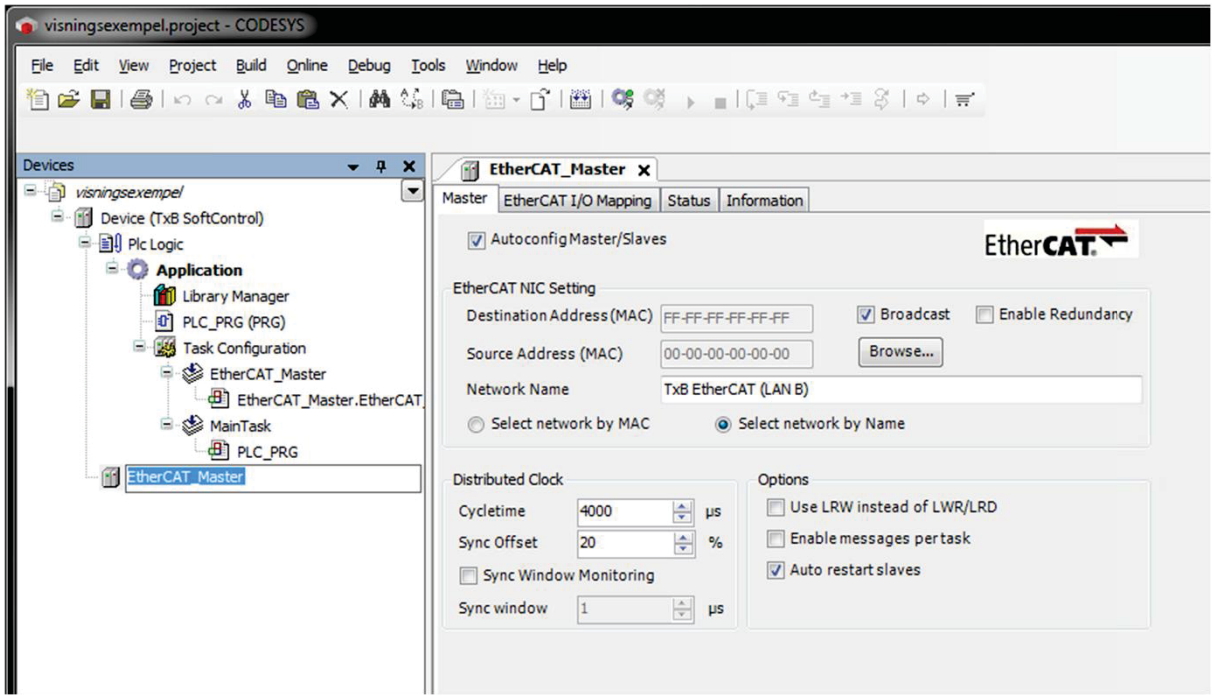
Dubbelklicka på "Device" i "Devices" och öppna fliken "PLC settings" och där se till att inställningarna som visas i bilden nedan stämmer överens.



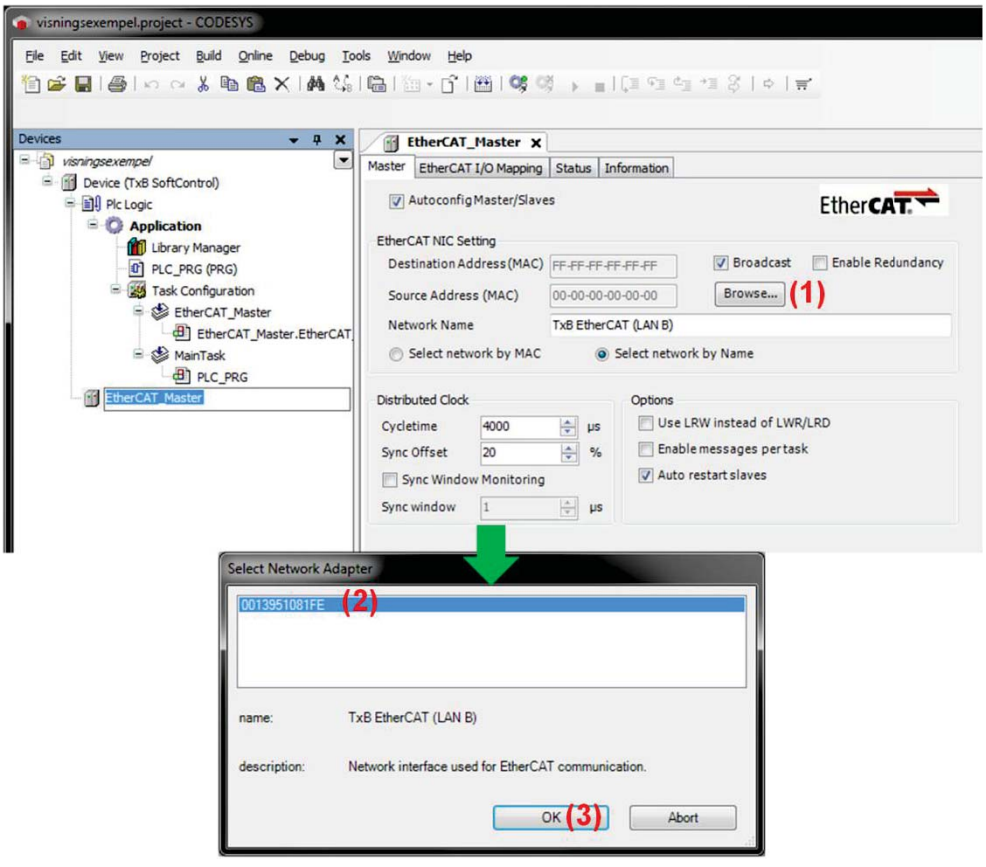
Installera "Crevis_EtherCAT_V1.xxx.xml" under "tools" > "Device Repository" om de inte redan finns.



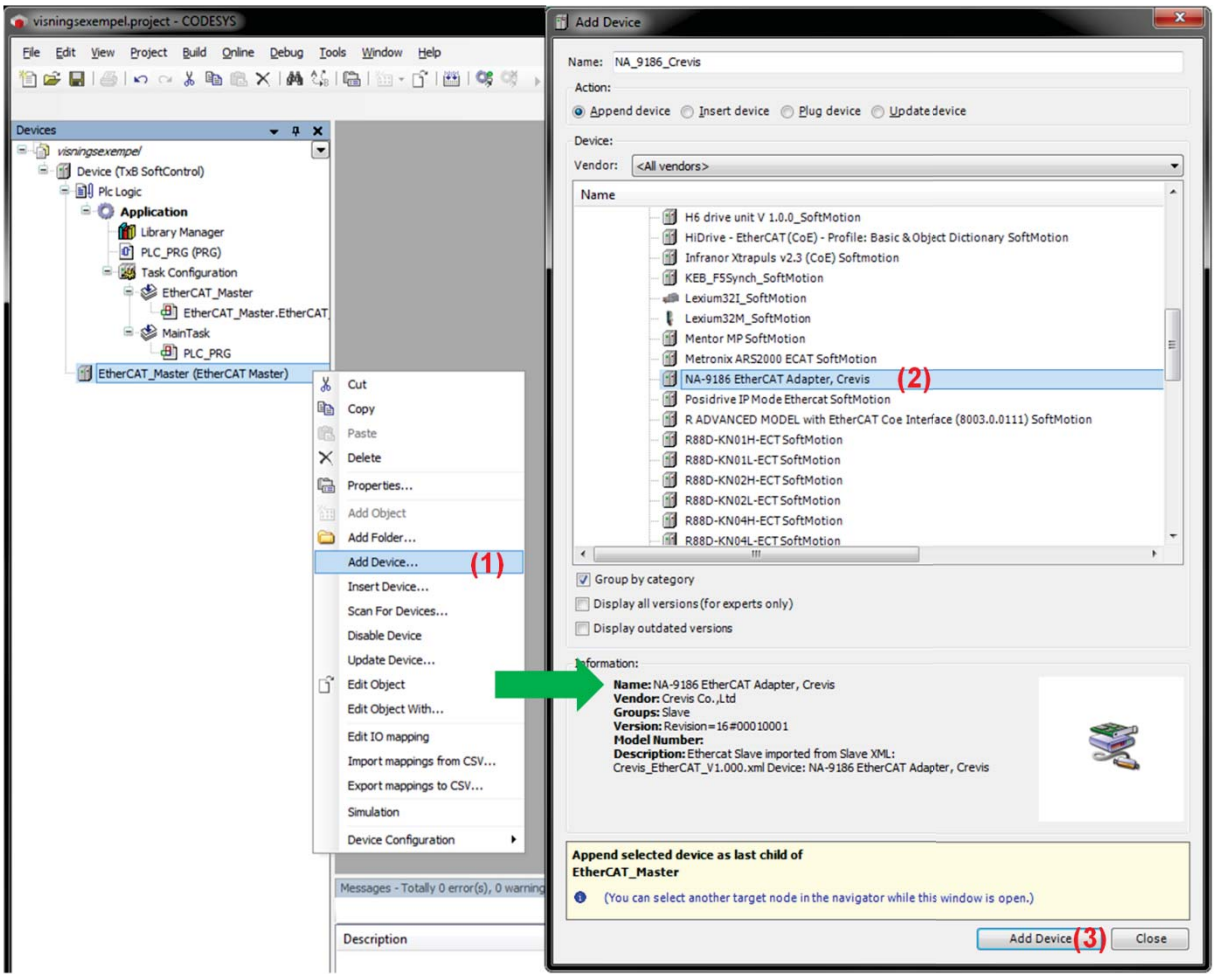
Dubbelklicka på "EtherCAT_Master" i "Devices" och fyll i informationen enligt figuren nedan. Använd LAN port B på T12B-panelen till den decentraliserade I/O-enheten.



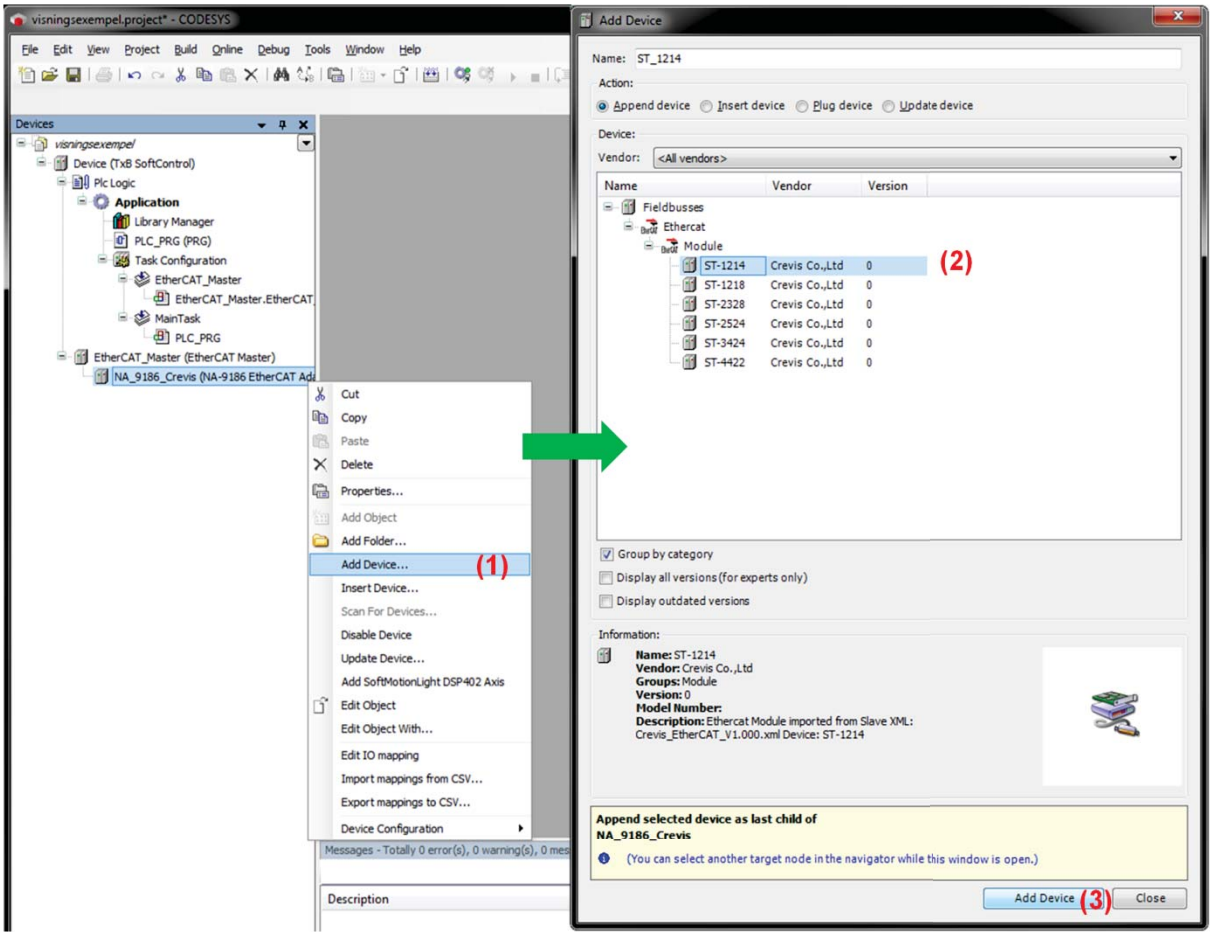
Klicka på "Browse..." där får man upp MAC-adressen till LAN port B på T12B-panelen och klicka på "OK".



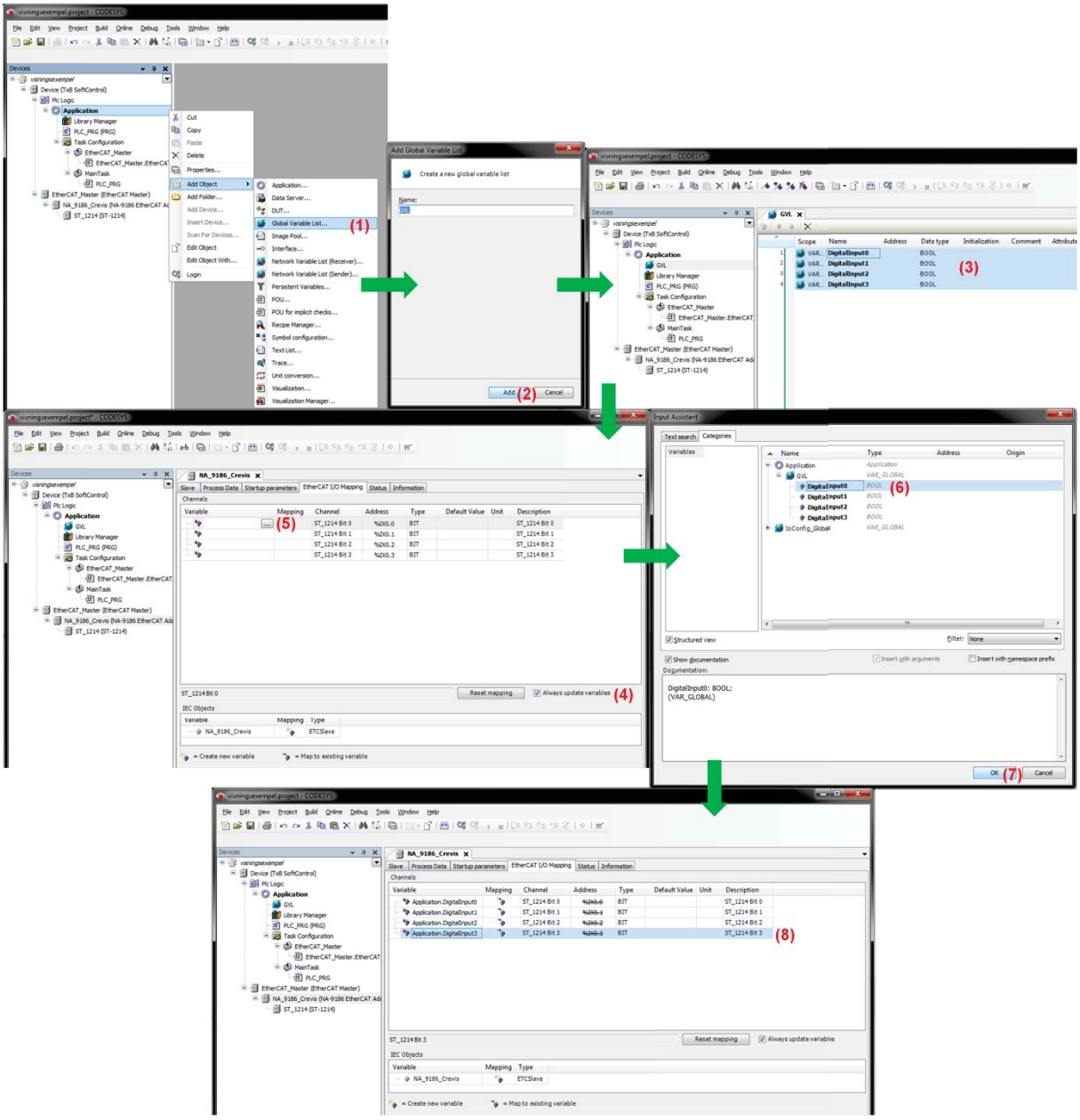
För att lägga till en slavenhet klicka på "EtherCAT_Master" i "Devices" och klicka på "Add Device" och leta upp rätt enhet enligt figuren nedan.



Lägg till modulerna genom att markera och högerklicka på den tillagda slavenheten och välj "Add Device" och i listan leta upp rätt moduler och klicka på knappen "Add Device", här är det bra att lägga in modulerna i "rätt ordning" från slavenheten och att lägga till alla moduler direkt och inte senare då det finns risk att en bugg förekommer i nästa steg.

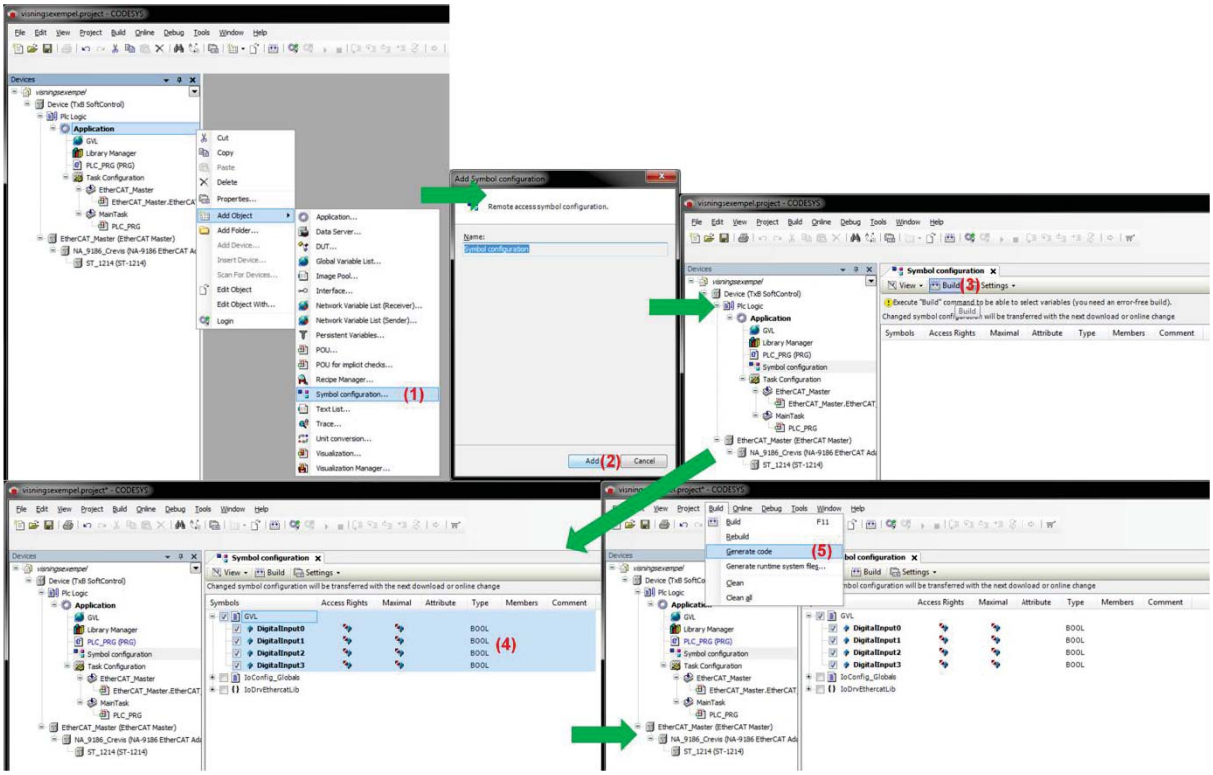


Skapa först motsvarande variabler i en globalvariabellista med bra namn och rätt datatyper för modulernas portar. Mappa om adresserna från slavenhetens expansionsmoduler till variabler genom att dubbelklicka på slavenheten i "Devices" och klicka på fliken "EtherCAT I/O Mapping" markera rutan "Always update variables". Dubbelklicka på den variabel vilket ska mappas om och klicka på knappen (...) och leta upp rätt variabel i den globala variabellistan (om många olika variabler ska mappas om, kopiera det som visas under "Variable" och klistra in i resten av rutorna och ändra den siffra som skiljer).

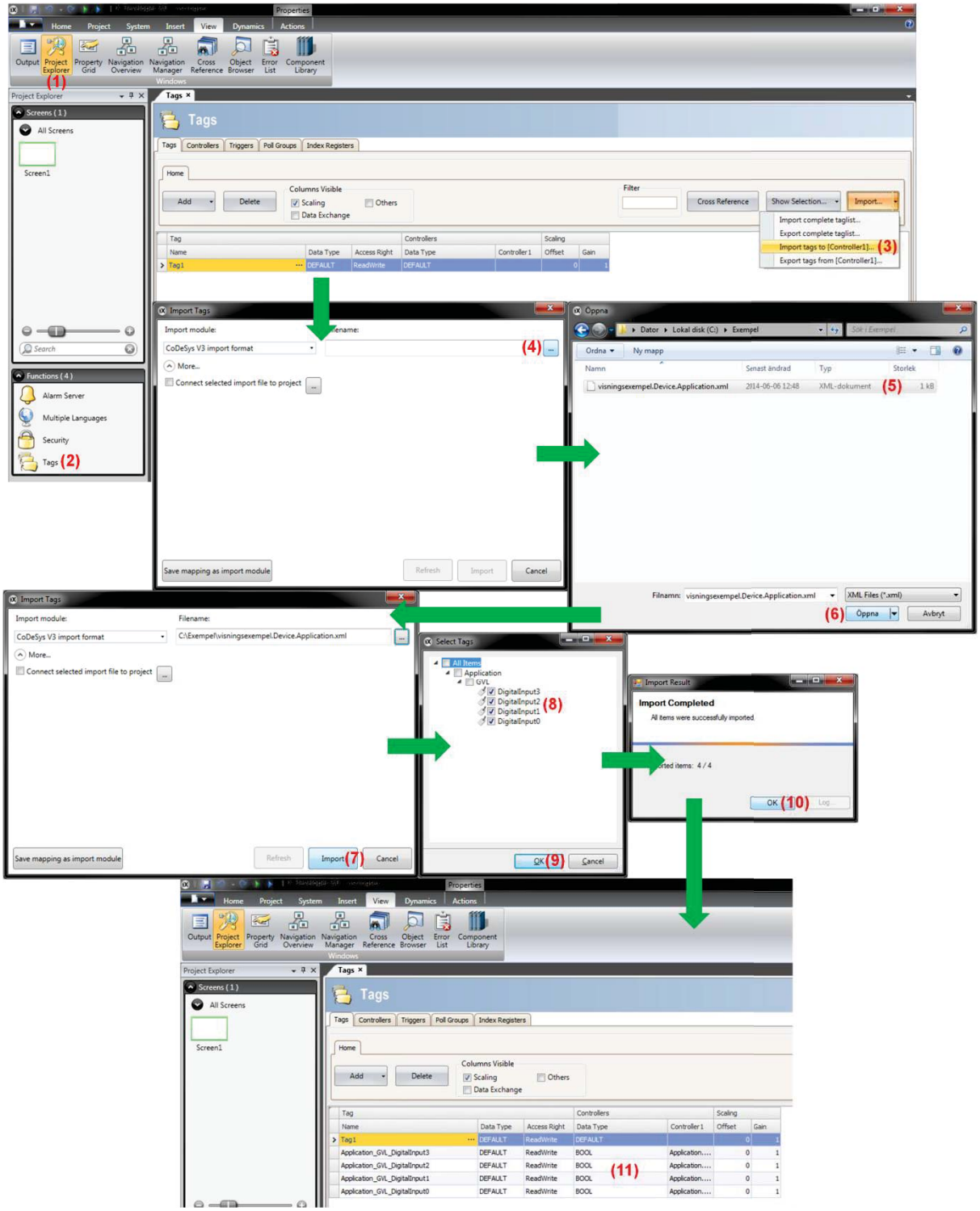


Bilaga D Överföring av variabler mellan softPLC- och HMI-del (7)

I CODESYS projektet markera och högerklicka på "Application" under "Devices" och välj "Add Object", där i välj "Symbol configuration..." utför en "build" och välj därefter önskade variabler. Gå in under "Build" och klicka på "Generate code" för att filen ska skapas.



Öppna projektet och välj "Tags" i "Functions" vilket finns i "Project Explorer" tryck på den lilla pilen bredvid knappen "Import..." och välj "Import tags to [Controller]..." se till att informationen är lika enligt bilden nedan. Filen som ska importeras hittas i projektmappen vilket CODESYS projektet skapades i.

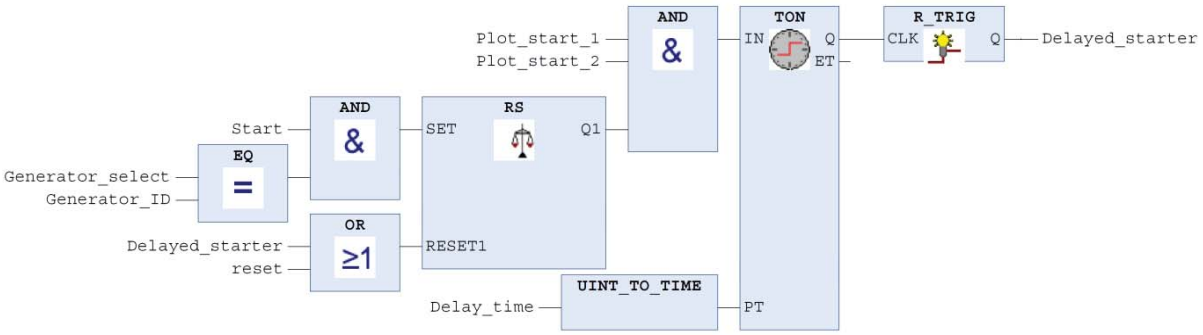


Bilaga E Koden till Freq_Period_converter

```
1 IF ( Period_time_prev <> Period_time ) THEN
//Om inmatad periodtid ändras utförs detta
2   Period_out := Period_time ;
//Periodtiden matas direkt ut
3   Period_time_converted := UINT_TO_REAL ( Period_time ) ;
//Periodtiden konverteras för att kunna beräknas om till
    frekvens
4   Freq := ( 1000 / Period_time_converted ) ;
//Periodtiden beräknas om till frekvens
5   Freq_prev := Freq ;
//Den nya frekvensen sparas
6   Period_time_prev := Period_time ;
//Den nya periodtiden sparas
7 ELSIF ( Freq_prev <> Freq ) THEN
//Annars om inmatad frekvens ändras utförs detta
8   Period_time := REAL_TO_UINT ( 1000 / Freq ) ;
//Frekvensen beräknas och omvandlas till periodtid
9   Period_out := Period_time ;
//den nya periodtiden matas ut
10  Freq_prev := Freq ;
//Den nya frekvensen sparas
11  Period_time_prev := Period_time ;
//Den nya periodtiden sparas
12 END_IF
```

Bilaga F Koden till Generator_delay

TON är blocket som fördröjer utsignalen beroende på den satta tiden, det som sker vid start är att den valda generatorn "Generator_select" jämförs med den tilldelade identifikationen "Generator_ID" om detta uppfylls väntar sedan blocket på att de två kurvorna ska starta. När dessa startar sätts fördröjningsblocket igång och på given signal skickas en puls ut som startsignal samt även som reset signal till det tillkopplade yttre blocket.



Bilaga G Koden till HD_Chart

```
1 IF start_hd_chart THEN
2   Starting := FALSE ;
3   time_is_out := FALSE ;
4   int_counter := 0 ;
5   local_reset := 0 ;
6   WHILE local_reset < 60000 DO
//Skriver in "0" på alla element i den stora arrayen
7       digital_array [ local_reset ] := 0 ;
8       local_reset := local_reset + 1 ;
9   END_WHILE ;
10  first_start := TRUE ;
//Säkerställer att inte den stora arrayen startar direkt vid
    programstart
11 ELSIF first_start THEN
12  Starting := TRUE ;
//Indikerar att värden nu börjar skrivas in i den stora
    arrayen
13  IF int_counter < 60000 THEN //
14      digital_array [ int_counter ] := analog_input ;
// Här skrivs värden in i den stora arrayen
15      int_counter := int_counter + 1 ;
16  ELSE
17      time_is_out := TRUE ;
//Den stora arrayen är full
18  END_IF ;
19 END_IF ;
20
21 IF local_counter = 500 THEN
//Återställer räknaren för den lilla arrayen som förs över
    till HMI-delen
22  local_counter := 0 ;
23 ELSE
24  out_counter := ( 500 * x_min_in + ( x_max_in - x_min_in ) *
    local_counter ) ;
//Skalningsfunktion för den stora arrayen, möjlig gör
    zoomfunktion
25  transfer_array_out [ local_counter ] := digital_array [
    out_counter ] ;
//värden från den stora arrayen överförs till den lilla
    arrayen med hjälp av skalningsfunktionen
26  local_counter := local_counter + 1 ;
27 END_IF ;
```

Bilaga H Programmet X_Scaler

```
1 IF ( x_max_prev <> x_max ) OR ( x_min_prev <> x_min ) THEN
//Aktiveras om värdet på x_min eller x_max ändras
2   x_max_prev := x_max ;
//Spara de nya värdena för att senare kunna detektera ändring
3   x_min_prev := x_min ;
4   counter := x_min_prev * 500 ;
//Startvärde ändras när x_min ändras, "500" är skalningsvärde
5   FOR counter2 := 0 TO 499 BY 1 DO
//Denna LOOP körs 500 ggr då iX Developer har en begränsning
    på 512 st element i en array
6       x_array [ counter2 ] := counter ;
//Värdena matas in i "x_array" med rätt fördelning mellan de
    punkter som får plats inom det önskade intervallet
7       counter := counter + ( x_max_prev - x_min_prev ) ;
    //Stegets storlek i X-led bestäms här
8   END_FOR ;
9   x_scale_done := NOT x_scale_done ;
//"x_xcale_done" inverteras när funktionen är klar för att iX
    Developer ska veta när värden kan tas emot
10 END_IF ;
```

Bilaga I Koden till HD_Chart_continuous

```
1 IF start_chart THEN
//Återställer innan start
2   int_counter := 0 ;
3   local_reset := 0 ;
4   Trigger_counter := 0 ;
5   Trigger_prev := ( Interval_time - 2 ) / 2 ;
//"Trigger" kommer i sekunder som räknas om med avseende på
    500 mätpunkter á 4 ms
6
7   WHILE local_reset < 500 DO
//Skriver in "0" på alla element i arrayen
8       transfer_array_out [ local_reset ] := 0 ;
9       local_reset := local_reset + 1 ;
10  END_WHILE ;
11
12  first_start := TRUE ;
//Säkerställer att inte arrayen startar direkt vid
    programstart
13
14 ELSIF first_start THEN
15  IF ( int_counter < 500 ) AND ( Trigger_counter >=
    Trigger_prev ) THEN
//Sker när samplingsräknaren kommer upp i inställt värde
16      transfer_array_out [ int_counter ] := analog_input ;
// Här skrivs värden in i arrayen
17      int_counter := int_counter + 1 ;
18      Trigger_counter := 0 ;
//Samplingsräknaren nollställs
19  END_IF ;
20
21  IF ( int_counter >= 500 ) THEN
22      int_counter := 0 ;
23  END_IF ;
24
25  Trigger_counter := Trigger_counter + 1 ;
//Samplingsräknare för att öka tidsspannet
26 END_IF ;
27
```

Bilaga J Koden till MeasurePoint

```
1 IF ( ( P_x_prev <> P_x_input ) OR ( x_max_in <> x_max_prev )
    OR ( x_min_in <> x_min_prev ) ) THEN
//Utförs när P_x_input, x_max_in eller x_min_in ändras
2   offset_y := 50 ;
//Sätter ett fast värde för storleken på plustecknet i Y-led
3   offset_x := ( ( x_max_in - x_min_in ) * 5000 ) / 1638 ;
//Sätter ett dynamiskt värde för plustecknets bredd, beroende
    på x_min och x_max
4   P_x [ 0 ] := P_x_input ;
//Placerar ut plustecknets punkter i x_led
5   P_x [ 1 ] := P_x_input ;
6   P_x [ 2 ] := P_x_input ;
7   P_x [ 3 ] := P_x_input + offset_x ;
8   P_x [ 4 ] := P_x_input - offset_x ;
9   P_y [ 0 ] := Array_in [ P_x_input ] + offset_y ;
//Placerar ut plustecknets punkter i y_led
10  P_y [ 1 ] := Array_in [ P_x_input ] - offset_y ;
11  P_y [ 2 ] := Array_in [ P_x_input ] ;
12  P_y [ 3 ] := Array_in [ P_x_input ] ;
13  P_y [ 4 ] := Array_in [ P_x_input ] ;
14  IF P_y [ 2 ] < offset_y THEN
//Löser bugg då plustecknet skrider utanför diagrammets
    gränser i y-led
15      P_y [ 1 ] := 0 ;
16  END_IF
17  IF P_x [ 2 ] < offset_x THEN
//Löser bugg då plustecknet skrider utanför diagrammets
    gränser i x-led
18      P_x [ 4 ] := 0 ;
19  END_IF
20  P_x_prev := P_x_input ;
//sparar P_x_input för att senare kunna övervaka ändring
21  x_max_prev := x_max_in ;
//sparar x_max_in för att senare kunna övervaka ändring
22  x_min_prev := x_min_in ;
//sparar x_min_in för att senare kunna övervaka ändring
23  MeasurePoint_done := NOT MeasurePoint_done ;
//Inverterar värdet på MeasurePoint_done när operationen är
    klar
24 ELSIF Reset OR ( x_min * 500 ) > P_x_input OR P_x_input > (
    x_max * 500 ) THEN
//Tar bort mätpunkten om den hamnar utanför intervallet
25  FOR Counter := 0 TO 4 BY 1 DO
//Återställer plustecknet, sätter alla punkter i samma punkt,
    tecknet syns alltså inte längre
26      P_x [ counter ] := 0 ;
27      P_y [ counter ] := 0 ;
28  END_FOR ;
29  P_x_prev := P_x_input ;
//sparar P_x_input för att senare kunna övervaka ändring
```

```
30  MeasurePoint_done := NOT MeasurePoint_done ;
//Inverterar värdet på MeasurePoint_done när operationen är
    klar
31 END_IF ;
```

Bilaga K Koden till MeasurePoint_Delta

```
1 IF P_x2 >= P_x1 THEN
//Om P_x2 är större då görs följande uträkning
2   P_x_delta := ( P_x2 - P_x1 ) * Scale_Interval ;
3   IF ( P_y2 >= P_y1 ) THEN
//Om P_y2 är större då görs följande uträkning
4       P_y_delta := P_y2 - P_y1 ;
5   ELSE
//annars görs detta
6       P_y_delta := P_y1 - P_y2 ;
7   END_IF
8
9 ELSIF P_x2 < P_x1 THEN
//Nu om P_x1 är större istället är det följande uträkning som
    utförs
10  P_x_delta := ( P_x1 - P_x2 ) * Scale_Interval ;
11  IF ( P_y2 >= P_y1 ) THEN
//Om P_y2 är större då görs följande uträkning
12      P_y_delta := P_y2 - P_y1 ;
13  ELSE
//annars görs detta
14      P_y_delta := P_y1 - P_y2 ;
15  END_IF
16 END_IF ;
```

Bilaga L Koden till Stepfunction

```
1 IF Activate THEN
//Startsignal
2   signal_out := Amplitude ;
//Flyttar och håller kvar signalnivån
3 ELSIF Reset THEN
//Återställer signalnivån vid "Reset"
4   signal_out := 0 ;
//Återställs till "0"
5 END_IF ;
```

Bilaga M Koden till Rampfunction

```
1 IF Reset THEN
2   signal_out := 0 ;
3   int_start := FALSE ;
4 ELSIF Start OR int_start THEN
5   int_start := TRUE ;
//Håller signalen så att funktionen fortsätter tills reset
    sker
6   IF signal_out < top_value THEN
//Om toppvärde inte har nåtts sker följande...
7       signal_out := REAL_TO_UINT ( ( K * time_counter ) / 250
    ) ;
//K bestämmer lutningen på rampen
8       time_counter := time_counter + 1 ;
//Räknare för att öka succesivt öka rampen
9   ELSIF signal_out >= top_value THEN
//Detta sker när signalen nått max värdet
10      signal_out := top_value ;
//Håller signalen på det inmatade max värdet
11  END_IF ;
12 END_IF ;
```

Bilaga N Global variabellista från CODESYS

Variabelnamn	Datatyp {initialvärde}	Beskrivning
M4_ST3424_In0	UINT	Analog insignal 0
M4_ST3424_In1	UINT	Analog insignal 1
M4_ST3424_In2	UINT	Analog insignal 2
M4_ST3424_In3	UINT	Analog insignal 3
M5_ST4422_Out0	UINT	Analog utsignal 0
M5_ST4422_Out1	UINT	Analog utsignal 1
Scale_Interval	UINT {4}	Intervalltiden för task ställs in här för att ge korrekta värden, fungerar dock inte överallt
plot_1_chart_sel	UINT {0}	Val av vilken kanal som ska kopplas mot den första kurvan i analysdiagrammet
plot_2_chart_sel	UINT {1}	Val av vilken kanal som ska kopplas mot den andra kurvan i analysdiagrammet
digital_out_array	ARRAY [0..60000] OF UINT	Alla mätvärden för den första kurvan i analysdiagrammet
digital_out_array2	ARRAY [0..60000] OF UINT	Alla mätvärden för den andra kurvan i analysdiagrammet
transfer_array	ARRAY [0..499] OF UINT	Jämnt utvalda mätvärden av den första kurvan beroende av "x_max" och "x_min" i analysdiagrammet
tranfer_array2	ARRAY [0..499] OF UINT	Jämnt utvalda mätvärden av den andra kurvan beroende av "x_max" och "x_min" i analysdiagrammet
x_array	ARRAY [0..499] OF UINT	X-array för analysdiagrammet
x_scale_done	BOOL {FALSE}	Värdet inverteras då X-arrayen för analysdiagrammet har uppdaterats
x_max	UINT {120}	Största värdet som visas i analysdiagrammet
x_min	UINT {0}	Minsta värdet som visas i analysdiagrammet
PID_Prop	REAL {1}	Proportionella konstanten för PID-regulatorn
PID_Integ	REAL {1}	Integreringstiden för PID-regulatorn
PID_Deriv	REAL {1}	Dervieringstiden för PID-regulatorn
PID_overflow	BOOL	Overflow från integralverkan
PID_actual_value	REAL	Ärvärdet som PID-regulatorn får som svar
PID_set_value	REAL {0}	Börvärdet som sätts till PID-regulatorn

PID_manual_active	BOOL	Om "TRUE" flyttas börvärdet direkt till styrsignalen, manuell justering av processen
PID_offset_Y_value	REAL {0}	Om "Output_reset" är "TRUE" flyttas detta värde som offset värde till PID-regulatorns Y-värde
PID_min_Y	REAL {0}	Undre gränsvärde för PID-regulatorns styrsignal
PID_max_Y	REAL {10}	Övre gränsvärde för PID-regulatorns styrsignal
PID_limits_reached	BOOL	Gränsvärdet har nåtts för PID-regulatorn om "TRUE"
PID_real_Y	REAL	Styrsignalen från PID-regulatorn omvandlad till intervallet 0-10 V
PID_Active	BOOL	Släpper fram styrsignalen från PID-regulatorn till vald utgående kanal
PID_sel_out	UINT {0}	Val av vilken kanal styrsignalen från PID-regulatorn ska sändas ut på
PID_out_bypass	UINT	Skickar styrsignalen från PID-regulatorn till kanalväljaren
PID_Chart_interval	UINT {60}	Bestämmer kurvans hela intervall tid i sekunder för PID-regulatorn
PID_transfer_array_1	ARRAY [0..499] OF UINT	Ärvärdet med historik så länge som "PID_Chart_interval" är inställt på
PID_transfer_array_2	ARRAY [0..499] OF UINT	Styrvärdet med historik så länge som "PID_Chart_interval" är inställt på
PID_sel_in	UINT	Val av vilken kanal ärsignalen ska avläsas från till PID-regulatorn
Generator_start	BOOL	Aktiverar startprocessen av generatorer med inväntan på att graferna ska starta
Generator_sel_start	UINT	Val av vilken generator som ska startas och få sända ut en signal, beroende av ett ID
Start_delay	UINT	Fördröjd starttid för generatorerna i ms
plot_1_chart_1_start	BOOL	Kurva ett har startat i analysdiagrammet
plot_2_chart_1_start	BOOL	Kurva två har startat i analysdiagrammet
Output_reset	BOOL	Återställningssignal vilket återställer samtliga block där återställning är möjligt
SGEN_Amplitude	REAL {10}	Amplitud för signalen från signalgeneratorn 0-10 V

SGEN_Freq	REAL	Frekvens för signalen från signalgeneratorn i Hz
SGEN_Period	UINT {30000}	Periodtid för signalen från signalgeneratorn i ms
SGEN_mode	UINT {1}	Val av signalform där:... 0 = Triangel 1 = Sinus 2 = Cosinus 3 = Rektangel 4 = Sågtand - stigande 5 = Sågtand - fallande 6 = Triangel över offset
SGEN_Offset	REAL {5.0}	Förskjutning av signalen från signalgeneratorn 0-10 V
SGEN_Sel_out	UINT {0}	Val av vilken kanal signalen från signalgeneratorn ska sändas ut på
SGEN_Active	BOOL	Släpper fram signalen från signalgeneratorn till vald utgående kanal, friläges start
SGEN_Active_internal	BOOL	Släpper fram signalen från steggeneratorn till vald utgående kanal, automatisk start
SGEN_out_bypass	UINT	Skickar signalen från signalgeneratorn till kanalväljaren
Step_out_bypass	UINT	Skickar signalen från steggeneratorn till kanalväljaren
Step_Active	BOOL	Släpper fram signalen från steggeneratorn till vald utgående kanal
Step_sel_out	UINT	Val av vilken kanal signalen från signalgeneratorn ska sändas ut på
Step_amplitude	REAL	Amplitud för signalen från steggeneratorn och rampgeneratorn 0-10 V
Ramp_out_bypass	UINT	Skickar signalen från rampgeneratorn till kanalväljaren
Ramp_Active	BOOL	Släpper fram signalen från rampgeneratorn till vald utgående kanal
Ramp_sel_out	UINT	Val av vilken kanal signalen från rampgeneratorn ska sändas ut på
Ramp_slope	REAL {1}	lutningen för signalen från rampgeneratorn i volt per sekund

P1_x_input	REAL {0}	Position på X-axeln av första mätpunkten från första kurvan i analysdiagrammet
MeasurePoint_done_1	BOOL	Första punkten på första kurvan är uppdaterad
P1_x	ARRAY [0..4] OF UINT	Första punkten på första kurvan, X-array för plustecknet i analysdiagrammet
P1_y	ARRAY [0..4] OF UINT	Första punkten på första kurvan, Y-array för plustecknet i analysdiagrammet
P2_x_input	REAL {0}	Position på X-axeln av andra mätpunkten från första kurvan i analysdiagrammet
MeasurePoint_done_2	BOOL	Andra punkten på första kurvan är uppdaterad
P2_x	ARRAY [0..4] OF UINT	Andra punkten på första kurvan, X-array för plustecknet i analysdiagrammet
P2_y	ARRAY [0..4] OF UINT	Andra punkten på första kurvan, Y-array för plustecknet i analysdiagrammet
P12_x	ULINT	Jämför första och andra punkten i X-led på första kurvan i analysdiagrammet
P12_y	UINT	Jämför första och andra punkten i Y-led på första kurvan i analysdiagrammet
P3_x_input	REAL {0}	Position på X-axeln av första mätpunkten från andra kurvan i analysdiagrammet
MeasurePoint_done_3	BOOL	Första punkten på andra kurvan är uppdaterad
P3_x	ARRAY [0..4] OF UINT	Första punkten på andra kurvan, X-array för plustecknet i analysdiagrammet
P3_y	ARRAY [0..4] OF UINT	Första punkten på andra kurvan, Y-array för plustecknet i analysdiagrammet
P4_x_input	REAL {0}	Position på X-axeln av andra mätpunkten från andra kurvan i analysdiagrammet
MeasurePoint_done_4	BOOL	Andra punkten på andra kurvan är uppdaterad

P4_x	ARRAY [0..4] OF UINT	Andra punkten på andra kurvan, X-array för plustecknet i analysdiagrammet
P4_y	ARRAY [0..4] OF UINT	Andra punkten på andra kurvan, Y-array för plustecknet i analysdiagrammet
P34_x	ULINT	Jämför första och andra punkten i X-led på andra kurvan i analysdiagrammet
P34_y	UINT	Jämför första och andra punkten i Y-led på andra kurvan i analysdiagrammet

Bilaga O Koden till Source_Drain_select

```
1 IF SGEN_Active OR SGEN_Active_internal THEN
//Om signalgenerator är aktiv matas värdet ut på vald kanal
2   IF ( SGEN_sel_out = 0 ) THEN
3       M5_ST4422_Out0 := SGEN_out_bypass ;
4   ELSIF ( SGEN_sel_out = 1 ) THEN
5       M5_ST4422_Out1 := SGEN_out_bypass ;
6   END_IF ;
7 END_IF ;
8
9 IF PID_Active THEN
//Om PID-regulator är aktiv matas värdet ut på vald kanal
10  IF ( PID_sel_out = 0 ) THEN
11      M5_ST4422_Out0 := PID_out_bypass ;
12  ELSIF ( PID_sel_out = 1 ) THEN
13      M5_ST4422_Out1 := PID_out_bypass ;
14  END_IF ;
15 END_IF ;
16
17 IF Step_Active THEN
//Om Steggenerator är aktiv matas värdet ut på vald kanal
18  IF ( Step_sel_out = 0 ) THEN
19      M5_ST4422_Out0 := Step_out_bypass ;
20  ELSIF ( Step_sel_out = 1 ) THEN
21      M5_ST4422_Out1 := Step_out_bypass ;
22  END_IF ;
23 END_IF ;
24
25 IF Ramp_Active THEN
//Om Rampgenerator är aktiv matas värdet ut på vald kanal
26  IF ( Ramp_sel_out = 0 ) THEN
27      M5_ST4422_Out0 := Ramp_out_bypass ;
28  ELSIF ( Ramp_sel_out = 1 ) THEN
29      M5_ST4422_Out1 := Ramp_out_bypass ;
30  END_IF ;
31 END_IF ;
32
33 IF Output_reset THEN
//Om reset är aktiv matas "0" ut på båda kanalerna
34  M5_ST4422_Out0 := 0 ;
35  M5_ST4422_Out1 := 0 ;
36 END_IF ;
```

Bilaga P Scriptet från Analysdiagrammet

```
namespace Neo.ApplicationFramework.Generated
{
    using System.Windows.Forms;
    using System;
    using System.Drawing;
    using Neo.ApplicationFramework.Tools;
    using Neo.ApplicationFramework.Common.Graphics.Logic;
    using Neo.ApplicationFramework.Controls;
    using Neo.ApplicationFramework.Interfaces;

    public partial class analys_graf
    {
        void X_scale_ValueChanged(System.Object sender,
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
//en numrerisk indikator är bunden till x_scale_done signalen från softPLC-delen. Aktiveras
vid ValueChanged, skriptet aktiveras
        {
            for (int counter=0; counter<500; counter++)
            {
                Globals.Tags.x_array1[counter].Value =
(Globals.Tags.Application_GVL_x_array[counter].Value * 4);
//arrayen skalas upp för att få rätt tidsskala
            }
        }
        void Measure_P1_ValueChanged(System.Object sender,
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
//en numrerisk indikator är bunden till MeasurePoint_done_1 signalen från softPLC-delen.
Aktiveras vid ValueChanged, skriptet aktiveras
        {
            for (int counter2=0; counter2<5; counter2++)
            {
                Globals.Tags.P1_x_local[counter2].Value =
(Globals.Tags.Application_GVL_P1_x[counter2].Value * 4);
//arrayen skalas upp för att få rätt tidsskala
            }
            Globals.Tags.P1_x_value_local.Value =
(Globals.Tags.Application_GVL_P1_x[2].Value * 4);
//visningsvärdet för mätpunkt 1 skalas upp för att få rätt tidsskala
        }
        void Measure_P2_ValueChanged(System.Object sender,
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
//en numrerisk indikator är bunden till MeasurePoint_done_2 signalen från softPLC-delen.
Aktiveras vid ValueChanged, skriptet aktiveras
```

```
        {
            for (int counter3=0; counter3<5; counter3++)
            {
                Globals.Tags.P2_x_local[counter3].Value =
(Globals.Tags.Application_GVL_P2_x[counter3].Value * 4);
//arrayen skalas upp för att få rätt tidsskala
            }
            Globals.Tags.P2_x_value_local.Value =
(Globals.Tags.Application_GVL_P2_x[2].Value * 4);
//visningsvärdet för mätpunkt 2 skalas upp för att få rätt tidsskala
        }
        void Measure_P3_ValueChanged(System.Object sender,
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
//en numrerisk indikator är bunden till MeasurePoint_done_3 signalen från softPLC-delen.
Aktiveras vid ValueChanged, skriptet aktiveras
        {
            for (int counter4=0; counter4<5; counter4++)
            {
                Globals.Tags.P3_x_local[counter4].Value =
(Globals.Tags.Application_GVL_P3_x[counter4].Value * 4);
//arrayen skalas upp för att få rätt tidsskala
            }
            Globals.Tags.P3_x_value_local.Value =
(Globals.Tags.Application_GVL_P3_x[2].Value * 4);
//visningsvärdet för mätpunkt 3 skalas upp för att få rätt tidsskala
        }
        void Measure_P4_ValueChanged(System.Object sender,
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
//en numrerisk indikator är bunden till MeasurePoint_done_4 signalen från softPLC-delen.
Aktiveras vid ValueChanged, skriptet aktiveras
        {
            for (int counter5=0; counter5<5; counter5++)
            {
                Globals.Tags.P4_x_local[counter5].Value =
(Globals.Tags.Application_GVL_P4_x[counter5].Value * 4);
//arrayen skalas upp för att få rätt tidsskala
            }
            Globals.Tags.P4_x_value_local.Value =
(Globals.Tags.Application_GVL_P4_x[2].Value * 4);
//visningsvärdet för mätpunkt 4 skalas upp för att få rätt tidsskala
        }
    }
}
```

Bilaga Q Scriptet från Regulatordiagrammet

```
namespace Neo.ApplicationFramework.Generated
{
    using System.Windows.Forms;
    using System;
    using System.Drawing;
    using Neo.ApplicationFramework.Tools;
    using Neo.ApplicationFramework.Common.Graphics.Logic;
    using Neo.ApplicationFramework.Controls;
    using Neo.ApplicationFramework.Interfaces;

    public partial class PIDregulator
    {

        void Chart_interval_ValueChanged(System.Object sender,
Neo.ApplicationFramework.Interfaces.Events.ValueChangedEventArgs e)
//en numrerisk indikator är bunden till PID_Chart_interval signalen från softPLC-delen.
Aktiveras vid ValueChanged, skriptet aktiveras
        {
            for (int counter=0; counter<500; counter++)
            {
                Globals.Tags.PID_Chart_x[counter].Value = (counter *
Globals.Tags.Application_GVL_PID_Chart_interval.Value) * 2;
//arrayen skalas upp för att få rätt tidsskala
            }
        }
    }
}
```