PLC-programmering av en pelletspanna

ZAKER TAQAWI OCH MOHAMMED ABOU NAASA
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Namn: Zaker Taqawi och Mohammed Abou Naasa

LUNDS UNIVERSITET
Lunds Tekniska Högskola

Deterministic

Examensarbete

PLC-programmering av en pelletspanna

Av

Zaker Taqawi och Mohammed Abou Naasa

Institutionen för Elektro- och informationsteknik
Lunds tekniska högskola, LTH, Lunds universitet
221 00 Lund, Sverige

Namn: Zaker Taqawi och Mohammed Abou Naasa

Sammanfattning

Detta examensarbete handlar om PLC-programmering av en pelletspanna till ett nytt fjärrvärmeverk som byggs i Landskronas stadsdel Örja i samarbete med Deterministic Control AB, för en slutkund Landskrona Energi AB. Material som P&I-schema, funktionsbeskrivning, I/O-lista levererades av ett annat företag, AKJ. All programmering görs i ett ABB 800xA-system.

De viktiga begrepp som examensarbetet vidrör, beskrivits i avsnittet teknisk bakgrund, nämligen funktionsbeskrivning, P&I-schema, KKS (Kraftwerk-Kennzeichen-System) och så vidare. Mjukvaror som har använts för programmering och uppbyggnad av processbilder i ABB 800xA-system har också beskrivits, som Control Builder, Function Designer och Plant Explorer.

Rapporten nedan beskriver processen för programmering av huvuddelar i pelletspannan, som luftsotning, roster, fläktar och så vidare, samt vilket programmeringsspråk som lämpade sig bäst för programmering av en del i pelletspannan. Därefter beskrivs kortfattat framtagning av processbilder. Examensarbetet framlägger även de erfarenheter av ett verkligt projekt som görs för en slutkund, och hur en programmerare behöver anpassa sig efter en kunds behov. Dessutom analyseras programmeringen i Function Designer enligt slutkundens önskemål samt de utmaningar och problem som uppstod under programmeringen.

Rapporten redogör dessutom för hur ett verkligt arbetsprojekt som kan utformas när flera parter är involverade, nämligen AKJ, Landskrona Energi AB och Deterministic Control AB.

Nyckelord: P&I-schema, funktionsbeskrivning, Control Builder, Function Designer, I/O lista, HMI, PLC, KKS, 800xA-system

Namn: Zaker Taqawi och Mohammed Abou Naasa

Abstract

This thesis concerns the PLC-programming of a pellet boiler for a new district heating plant being built in the Örja district of Landskrona in collaboration with Deterministic Control AB, for a final customer, Landskrona Energi AB. Material such as PI diagrams, functional descriptions, and I/O-lists were provided by another company, AKJ. All programming is carried out in an ABB 800xA system.

The important concepts related to the thesis are described in the technical background, namely functional description, PI diagrams, KKS (Kraftwerk-Kennzeichen-System), and so on. Software used for programming and building process graphics in the ABB 800xA system have also been described, such as Control Builder, Function Designer, and Plant Explorer.

The report below describes the process of programming the main components of the pellet boiler, such as air soot blowing, grate, fans, and so on, as well as the programming language that was best suited for programming a part of the pellet boiler. The creation of process graphics is also briefly described. The thesis also presents the experiences of a real project carried out for a final customer, and how a programmer needs to adapt to a customer's needs. In addition, the programming in Function Designer is analyzed according to the final customer's requirements, as well as the challenges and problems encountered during programming.

Furthermore, the report describes the experience of how a real project can look like, especially when multiple parties are involved, namely AKJ, Landskrona Energi AB, and Deterministic Control AB.

Keywords: P&ID diagrams, functional descriptions, Control Builder, Function Designer, I/O list, HMI, PLC, KKS, 800xA system

Namn: Zaker Taqawi och Mohammed Abou Naasa

Innehållsförteckning

Namn: Zaker Taqawi och Mohammed Abou Naasa

Förord

Vi vill tacka följande personer som har varit involverade i detta examensarbete och som har kunnat bidra med hjälp för att dels lyckats programmera klart pelletspannan, men också för inlärning. Medarbetarna i både Deterministic och AKJ har bidragit med att göra detta projekt lärorikt, roligt och spännande under hela arbetsprocessen.

Vi skulle vilja börja med att tacka alla som jobbat i Deterministic Control AB:

- Conny Franzon som gjorde det möjligt att utforma arbetet till ett lämpligt examensarbete och lyckades övertyga både kunden och leverantören att lita på två examensarbetare med att få arbetet gjort, med enda syftet att lära ut.
- Ett stort tack till Claes Göran Ohlson för att ha varit en otrolig handledare. Nästan inget av arbetet hade kunnat bli klar utan hans handledning när det gäller att strukturera sitt arbete, påbörja generellt, vad man bör tänka på under olika moment, hur man programmerar i olika miljöer, hur man felsöker och med mera. Vi är evigt tacksamma för de kunskaper som vi har lärt oss under vår tid med honom.
- Torulf Wilberg som har bidragit med IT support gällande uppkoppling mot kundens miljö.

Sist men inte minst vill vi tacka grabbarna från AKJ:

- Martin Olsson från AKJ som bidrog till att tydliggöra funktionerna av pannan för lättare förståelse för programmering. Han har också varit hjälpsam under driftsättningen och hjälpt till med att stämma av pannans system under flera FAT-tester.
- Johan från AKJ som hjälpte till med att leda arbetet ute i fältet och felsöka olika problem som förekom under driftsättningen.

Vi vill även tacka vår handledare Mats Lilja och examinator Christian Nyberg för deras handledning under examensarbetet och deras undervisning på LTH.

Namn: Zaker Taqawi och Mohammed Abou Naasa

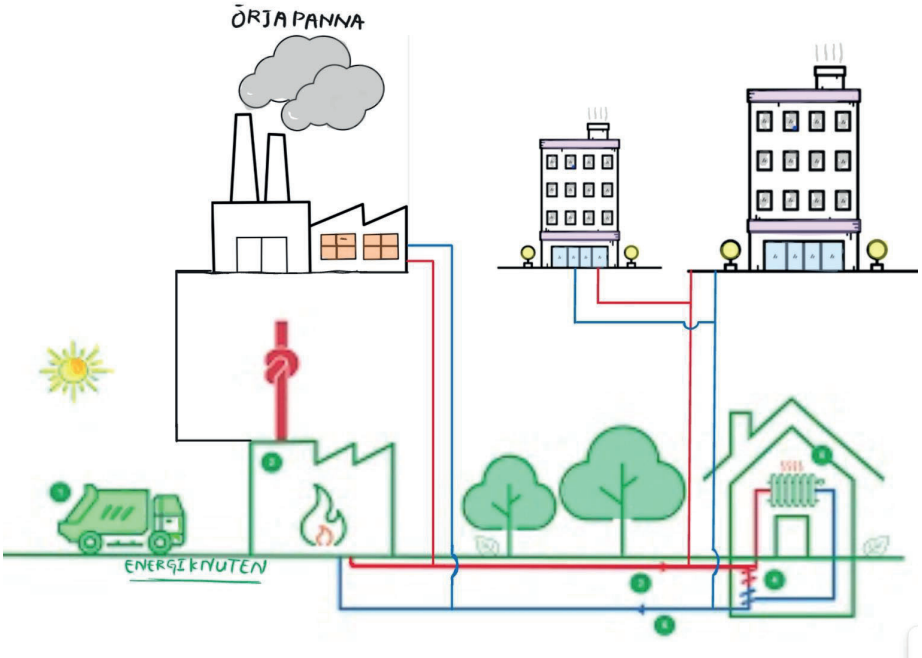
1 Inledning

I detta inledande avsnitt presenteras bakgrunden till detta examensarbete samt syftet, mål- och problemformuleringen och avgränsningen.

1.1 Bakgrund

Landskrona Energi AB är ett företag som levererar energi- och kommunikationslösningar. De producerar el och fjärrvärme genom kraftvärmeverk. Företaget är för närvarande engagerat i konstruktionen av en ny värmecentral i stadsdelen Örja i Landskrona. En pelletspanna ska monteras i fjärrvärmeverket som kommer att levereras av företaget AKJ.

AKJ energiteknik AB är ett svenskt företag som grundades 1997 och är specialiserat på att erbjuda lösningar inom energi- och miljöteknik. Företaget arbetar främst med att utveckla och sälja energieffektiva värmesystem och har som mål att minska energiförbrukningen och utsläppen av koldioxid.



Figur 1: En handritning på hur den nya fjärrvärmeanläggningen kommer se ut.

Figur 1 ovan illustrerar hur pannan ska installeras och dess syfte. Den är tänkt att kunna hjälpa till med att bidra till fjärrvärmenätet i Landskrona kontinuerligt och låta till exempel oljepannor användas som reserv ifall de skulle behövas beroende på omständigheterna. Beroende på vad temperaturen är utomhus och effektanvändningen av hushållen, så bestämmer Energiknuten, Landskrona Energi ABs kraftvärmeverk, hur mycket värme som pelletspannan i Örja ska producera för de hushåll som den är kapabel för. Örja får en order av Energiknuten att leverera en specifik mängd värme som behövs för just den dagen och det är upp till operatörerna i Örja att manuellt ställa in pannan så att den kan leverera det. Landskrona Energi AB har anlitat Deterministic Control AB för programmeringen av den nya anläggningen.

Deterministic Control AB är ett företag som erbjuder utveckling, genomförande, driftsättning och service inom industriell automation till sina kunder. Landskrona Energi AB vill att detta företag ska programmera och driftsätta pelletspannan till det nya värmeverket. Deterministic AB har tidigare programmerat klart en oljepanna i Tandådalen, som skiljer sig från pelletspannan som kommer att arbetas med i detta projektet. De har också arbetat med att uppgradera Höganäs energi, panncentralens styrsystem, servrar och nätverk med ett högre krav på IT-säkerhet. De har också utfört olika uppdrag för Svenska kraftnät, där de har programmerat ett fjärrstyrd reservkraftaggregat och för Öresundskraft där de har programmerat deras värmedrivna kylmaskiner för deras fjärrkyleproduktion och med mera.

Huvudmålet med examensarbetet är att PLC-programmera en pelletspanna och skapa processbilder/HMI för det nya fjärrvärmeverket i Örja i samarbete med Deterministic Control AB. Programmets grund kommer att vara baserade på de material som tillhandahålls av AKJ, vilka är IO-lista, P&I-schema och funktionsbeskrivning. Samtidigt under arbetets gång reda ut vilket programmeringsspråk, som exempelvis strukturerad text, sequential function chart och så vidare, vore gynnsamt att använda. Denna rapport syftar också till att utforska arbetsgången och de erfarenheter som erhållits genom användning av specifika program, såsom Function Designer och Control Builder. Dessutom undersöks hur ett verkligt arbetsprojekt kan utformas när flera parter är involverade, nämligen AKJ, Landskrona Energi AB och Deterministic Control AB.

Function Designer är en programvara som används för att utveckla och skapa automationsfunktioner för industriella processer. Det är en del av ABB 800xA-systemet och

används för att skapa anpassade funktioner för att styra, övervaka och automatisera industriella processer.

Programmets grund kommer att vara baserad på det material som tillhandahålls av AKJ, som IO-lista, P&I-schema och funktionsbeskrivning.

En IO-lista är en dokumentation som listar alla ingångar (inputs) och utgångar (outputs) som behövs för att koppla samman en enhet eller ett system inom automation och industriell styrning. Listan innehåller information om vilka sensorer, signaler, aktuatorer och I/O-moduler som används, samt vilka signaltyper och gränssnitt som behövs för att koppla samman systemet korrekt.

Ett P&I-schema (Piping and Instrumentation diagram) är en typ av teknisk ritning som används inom processindustrin för att beskriva en processanläggning och dess styrningssystem. P&I-schemat visar processflödet i anläggningen, samt vilka instrument som används för att övervaka och styra processen.

En funktionsbeskrivning är en teknisk beskrivning av hur en viss funktion ska utföras i ett system eller en process. Beskrivningen innehåller vanligtvis detaljerade instruktioner och specifikationer för hur funktionen ska utföras, vilka parametrar som ska användas, vilka krav som måste uppfyllas, samt vilka eventuella risker eller begränsningar som finns.

Inom processindustrin används styr- och reglerteknik för att styra och övervaka systemet på ett sådant sätt som säkerställer så att det fungerar korrekt och effektivt. Dessutom behövs det för att minska energiförbrukningen, uppnå hög kvalitet, öka produktiviteten och minska riskerna för driftstörningar och säkerhetsproblem.

PLC-programmeringen har utförts i ABB 800xA-system ‘Control Builder’ och ‘Plant Explorer Workplace’, som Deterministic Control AB använder sig av. Det finns ett antal färdiga typkretsbibliotek som kan tillämpas för programmeringen av olika objekt, exempelvis motor, ventil, pumpar, och m.m.

1.2 Syfte

Syftet med detta examensarbete är att PLC-programmera en pelletspanna och skapa processbilder/HMI som ska kunna styras av operatörer. Arbetet baseras på material från AKJ, inklusive IO-lista, PI-flödesschema, och funktionsbeskrivning. Rapporten kommer också att undersöka programmeringsspråk, nämligen strukturerad text, funktionsblocksdiagram och sequential function chart, som lämpar sig bäst för programmeringen av en del i pelletspannan. Samtidigt syftar detta arbete att utforska arbetsgången med flera parter involverade. Dessutom kommer även rapporten att framlägga de erfarenheter som har åstadkommits i tillämpning av specifika program, som Function Designer och Control Builder.

1.3 Målformulering

Första målet är att programmera pelletspannan rätt och säkert med den funktionsbeskrivning som blir levererad av AKJ, med hjälp av PLC-programmet och ett tydligt HMI-gränssnitt som operatören kan utnyttja. Det är en väldigt omfattande arbetsprocess som innebär mycket tid måste läggas på programmeringen, ständig avcheckning av programmet , ständig kommunikation mellan handledaren och studenterna och förståelse för pelletspannans funktioner.

Andra målet för projektet är att undersöka hur arbetet påbörjas och avslutas, vilka sorters problem som uppstod under arbetet och hanteringen av det. Denna rapport kommer analysera och framlägga de resultat som presenteras utifrån arbetet och de material som används.

Under programmeringen av pelletspannan ska programmeringsspråk, enligt IEC standarden 61131-3, undersökas, dvs vilket programmeringsspråk är lämpligt för en del i systemet. Ett exempel på en del i systemet kan vara eldningsautomatik, det är nämligen eldningen av pannan.

Ett annat mål är att göra programmeringen och processbilder så användarvänliga som möjligt för slutanvändarna.

Om allt går enligt tidsplanen för programmeringen så ska pannan eventuellt driftsättas när den är redo.

1.4 Problemformulering

Examensarbetet ska besvara följande problemformuleringar:

1. Vilka olika programmeringsspråk lämpar sig bäst för programmeringen av en del i en pelletspanna?
2. Vilka problem och utmaningar uppstod under PLC-programmeringen av pelletspannan?
3. Vilka erfarenheter kan beskrivas i samband med att använda Function Designer vid programmering av pelletspanna?
4. Är det möjligt att ha två personer arbeta med varandra med en enda Control Builder?
5. Vilka erfarenheter innebär det att delta i ett verkligt arbetsprojekt som involverar flera parter?

1.5 Motivering av examensarbetet

Intresse inom automationsteknik och bidrag till utvecklingen inom området på ett hållbart sätt ledde oss till valet av examensarbete. Det är viktigt för alla människor att ta hänsyn till miljön samtidigt som nya utvecklingar sker i världen. Landskrona Energi AB är ett företag som arbetar konstant med förnyelsebar el och värme i staden Landskrona. De bygger ett nytt värmeverk i stadsdelen Örja för att förse sina konsumenter med värme. Bränsle som företaget använder för produktion av el och värme är industriella restavfall av papper, trä och plast som återvinns. AKJ, som också är med i detta projektet, är ett företag som erbjuder hållbara och effektiva energilösningar till sina kunder. De tar hand om elkonstruktionen och produkter som behövs till den nya anläggningen.

Dessutom finns det en annan intresseväckande faktor med detta examensarbete. Nämligen att det ger en möjlighet att kunna sätta oss in i en verklig arbetssituation där kundens behov ligger i fokus. Detta kommer att hjälpa en att få en överblick över hur en arbetsprocess går till, vilket i sin tur leder till en utveckling.

1.6 Avgränsningar

Vi avgränsar vårt arbete till att programmera pannan med PLC samt bygga processbilder. Därmed ligger konstruktionen av pannan, elritningar eller mätningar av dess parametrar utanför ramen för vårt arbete. Samtidigt är vår avsikt att lägga större fokus på de program som tillämpas i det här projektet. Det är nämligen Function Designer och Control Builder.

Programmet ska både demonstrera själva driften genom grafiska processbilder. Det ska göras i form av FAT-test tillsammans med slutkunden, Landskrona Energi AB. Dessutom ska programmet utföra arbetet efter driftsättningen.

1.7 Resurser

Följande resurser tillämpades för utförandet av examensarbetet:

- ABB system 800xA
- Utbildning inom programmet ABB system 800xA
- Datorer
- Deterministic Control ABs anläggning för test och debugging
- Mentorer med goda erfarenheter
- Kontakter med AKJ och Landskrona energi AB
- Material från AKJ och Deterministic Control AB

1.8 Arbetsfördelning

Arbeten	Mohammed Abou Naasa	Zaker Taqawi
PLC-programmering	50	50
HMI och processbilder	50	50
Examensarbetesrapport	50	50
I/O lista	85	15
FAT-protokoll	10	90

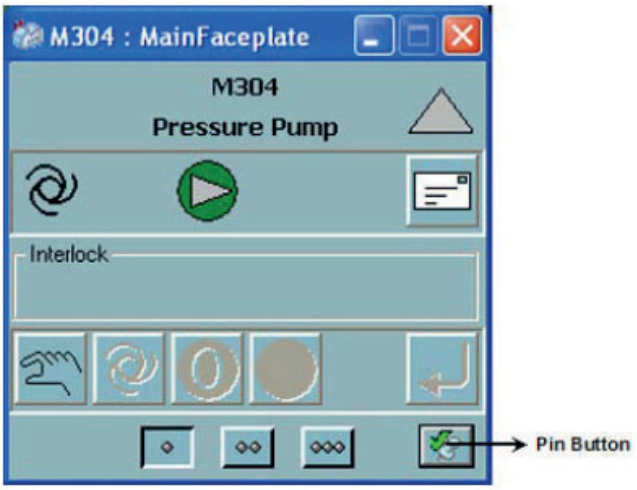
2 Teknisk bakgrund

2.1 HMI

HMI står för Human-Machine interface, där användaren kan styra ett system eller maskin via ett grafisk gränssnitt och inställningar som är visad på skärmen.Vad för slags inställningar som kan visas på HMI beror på vad programmeraren har bestämt sig för att visa på skärmen och vad som gäller för just den processen.

Valen av objekten[1] har betydelse för operatören som ska hantera HMI, därför att olika sorters objektlogik visar olika inställningsalternativ, exempelvis förmågan att köra igång en process i manuell eller automatisk styrning, eller att ha ett specifikt objekt som ska hantera en startsekvens av en process.

I ABB 800xA-system kallas HMI för varje objekt, för en ‘’faceplate’’. [3]



Figur 2: En faceplate för en pump som operatörerna kan se och styra [3]

2.2 Utvecklingsserver

Miljön som används för att programmera och simulera är en så kallad utvecklingsserver. Den gör det möjligt att både kunna programmera och simulera i en säker miljö utan att behöva

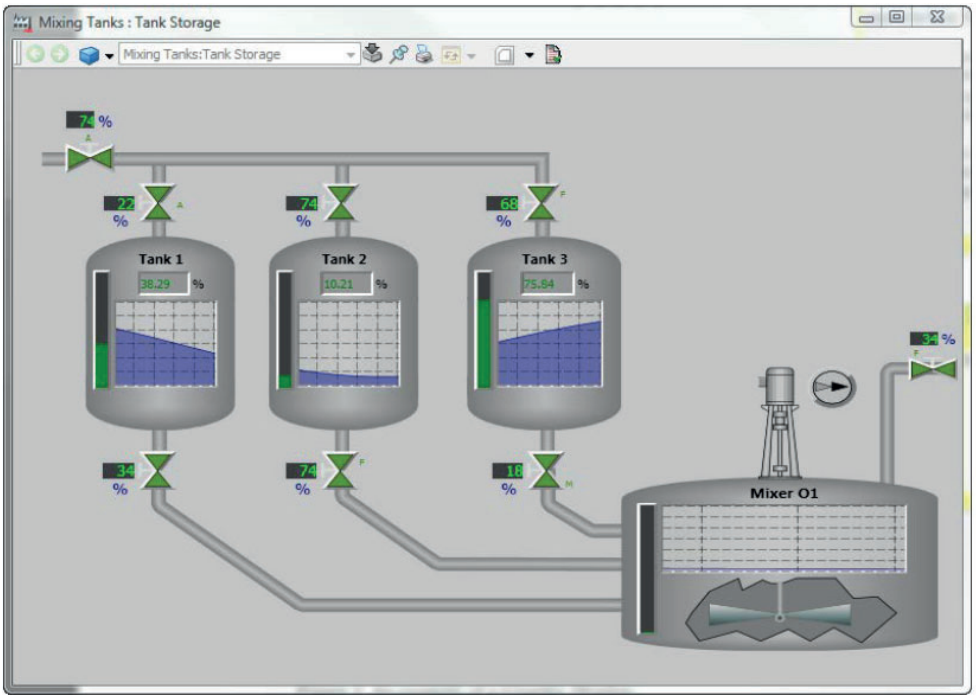
[1] Ett objekt är en färdig typkrets (funktionsblock med en inbyggd faceplate), som har en egen faceplate och som programmerare kan tillämpa.

ladda ner programmet till den fysiska hårdvaran, som den är avsedd för och på så sätt undvika risken att få fel kod som kan förstöra maskiner och system vid testning. Att kunna använda sig av en utvecklingsserver gör det enkelt för programmeraren att lätt kunna kompilera sin kod ifall justeringar behövs.

2.3 Process graphics

Process Graphics är ett program i Plant Work Explorer i ABBs 800xA system, där programmeraren och en generell användare kan grafiskt rita, infoga, m.m. grafiska display objekt med syfte att visuellt visa hur programmet fungerar på skärmen. Process Graphics är byggt för att göra det enkelt för operatören att ha kontroll över processer och system.

Mycket av Process Graphics funktionalitet bygger på att kunna göra det tydligt för operatören vad det är som sker i bilden.



Figur 3: Ett exempel på en grafisk display [3]

Figur 3 visar hur en processbild kan se ut för en operatör. I bilden syns olika objekt, exempelvis ventiler, motor och m.m. Operatören kan trycka exempelvis på något av ventil objekten så får operatören en faceplate på det objektet. Detta objekts faceplate kan se ut som i figur 2. Där syns det i vilket läge ventilen kör, dvs om den körs i autoläge eller i manuellt läge. Genom faceplate kan operatören dessutom styra ventilen genom att exempelvis stänga eller öppna ventilen.

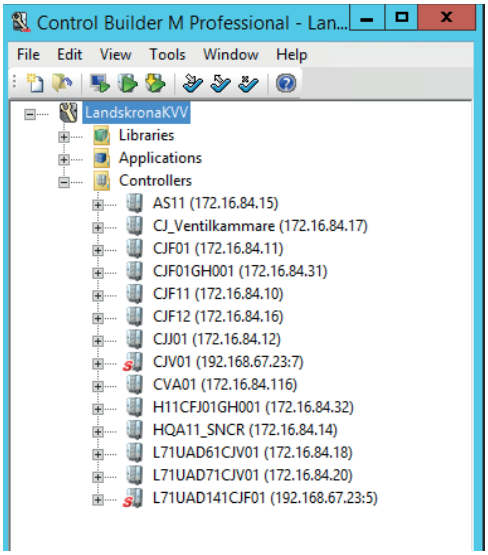
Process Graphics i ABB 800xA är utformat för att vara intuitivt och lättanvänt, vilket gör det enkelt för användare att navigera och hitta den information de behöver för att fatta beslut och styra processen på ett effektivt sätt. [3]

2.4 SoftController

Ett softcontroller är ett licensprogram från ABB 800xA som ser till att virtuellt kunna ladda ner program såsom om det vore en verklig fysisk controller med syfte att simulera, testköra, felsöka eller kompilera programkod. SoftController är kompatibelt med Control Builder och det krävs tillgång till Control Builder och en utvecklingserver, som licensen befinner sig på för att kunna aktivera SoftControllers funktionalitet, för via control buildern så förs koden ner till controllern. [1]

2.4 Simuleringsmiljö

För att kunna simulera programkoden krävs det nedladdning av programkoden till softController. Därefter kan programkoden testas i antingen Process Graphics, processbilder som har ritats för att simulera testning, eller i Control Builder. För att nerladdningen ska ske i softController behöver softController vara igång, ska vara kopplat till rätt controller via en IP-adress och dessutom ska controller vara satt i simulering (bokstaven s), som syns i figur 4 nedan.



Figur 4: Controller i simuleringsmiljön

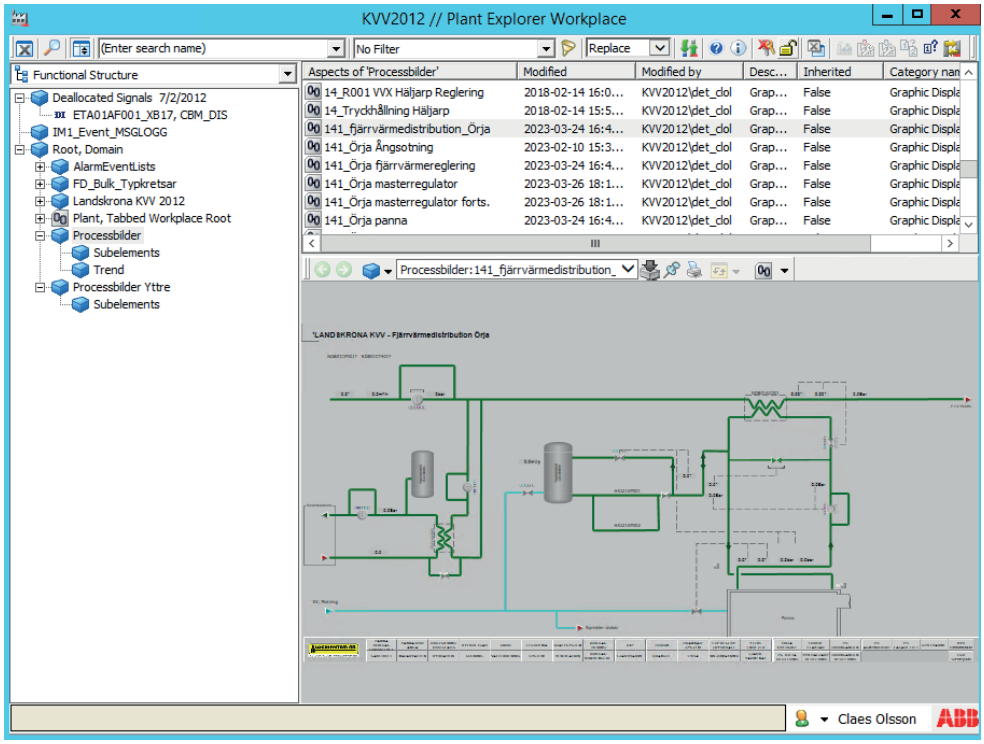
2.5 Plant Explorer Workplace

Plant Explorer används för att hantera, ordna och söka efter olika typer av objekt inom 800xA-systemet. Objekten organiseras i olika strukturer utifrån deras funktion och placering, och navigeringen och sökningen kan enkelt göras med hjälp av verktyget.

Plant Explorer är det huvudsakliga verktyget som används av ingenjörer för att bygga hierarkiskt strukturerade modeller av fabriken eller systemet. Verktyget är baserat på en strukturell hierarki som liknar den som används i Windows Explorer. De olika strukturerna representerar olika sätt att se på systemet och kan förbättras och modifieras efter behov.

Det finns flera olika strukturer tillgängliga som följande:
Functional Structure - Detta visar fabriken från ett processperspektiv.
Location Structure - Detta visar utrustningens fysiska placering i fabriken.
Control Structure - Detta visar kontrollnätverket utifrån nätverk, noder, fältbussar och stationer.

Allt som används i fabriken beskrivs av olika objekt, exempelvis motorer, ventiler, givare och controllers. Dessa objekt innehåller sedan relevant information i form av olika aspekter. Dessa aspekter kan vara processgrafik, kontrolldialoger och larmlistor. Aspekter kan visas på olika sätt och det går att filtrera vilken information användaren vill se genom att välja olika visningslägen. Verktyget har också en sökfunktion som gör det enkelt att hitta olika aspekter oavsett struktur. Slutligen är det enkelt att komma åt objektet direkt från ABB 800xA System Operator Workplace. [5]

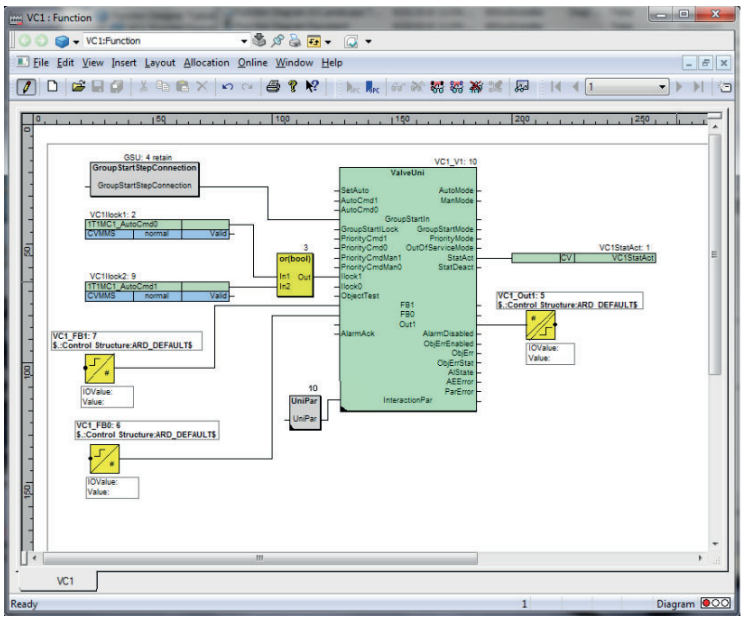


Figur 5: Struktur i Plant Explorer

2.6 Function Designer

Function Designer är en programvara som utnyttjas i plant explorer. All programmering i Function Designer sker i Plant Explorer. Precis som alla andra programvaror, som exempelvis Control Builder, används function designer för att för att skapa, konfigurera och implementera funktioner för processautomatiseringssystem. Vad som gör programvaran unik

i jämförelse med andra programvaror, är att användaren eller operatören kan lätt komma till koden via objekt i processbilder i Process Graphics. [4]



Figur 6: Function Designer i Plant Work Explorer [4]

2.7 KKS-beteckningssystem

KKS står för (Kraftwerk Kennzeichen System) beteckningssystem. KKS beteckning är en identifiering enligt KKS för elektrisk utrustning och system inom elkraftsystem. Det generella målet med en KKS-beteckning är att ge en enhetlig och systematisk metod för att identifiera och beskriva elektrisk utrustning, system och anläggningar som är kopplade. En KKS-beteckning består vanligtvis av en serie bokstäver och siffror som beskriver en specifik komponent eller system på ett unikt sätt. Bokstäverna i en KKS-beteckning representerar olika kategorier av information, såsom typ av utrustning, plats, och funktion. Siffrorna används för att skilja olika komponenter med samma beskrivning.

Till exempel kan en pump ha en KKS-beteckning som "UAD141NDB01AP001-XQ01", där “UAD141” beskriver anläggningsbeteckning, dvs vilken anläggning som den här pumpen tillhör. “NDB01” beskriver att pumpen används för returledning. “AP001” är en beteckning

för en pump och slutligen "XQ01" förklara vilken signal beteckning den har, exempelvis om det är en feedback signal eller m.m.

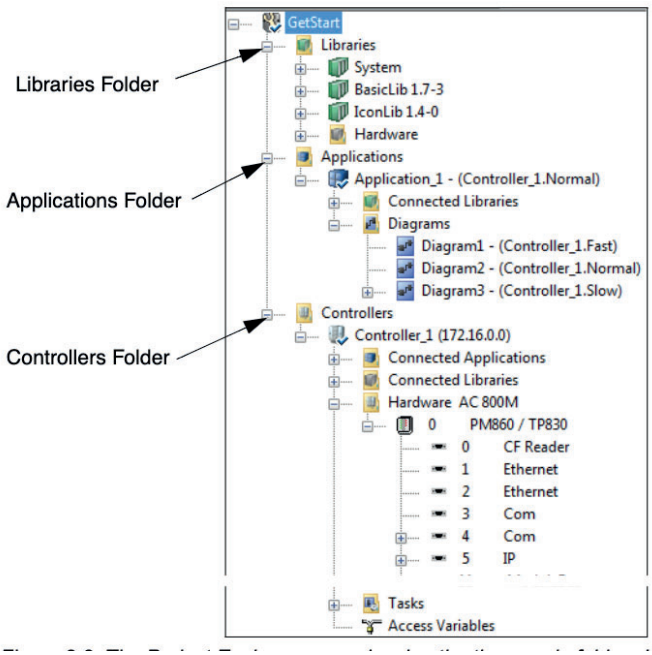
Genom att följa KKS beteckningssystem kan en elektrisk utrustning och system identifieras och beskrivas på ett enhetligt sätt. [6]

2.8 Control Builder

ABBs Control Builder är ett mjukvaruverktyg för automationsingenjörer som används för att designa, konfigurera, testa och driftsätta automations- och styrsystem. Control Builder stöder följande programmeringsspråk:

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

Dessa fem programmeringsspråk är definierade i IEC standarden 61131-3.



Figur 7: Ett exempel på strukturen i Control Builder (3BSE041880-610 A)

Project Explorer pane består av tre huvuddelar, nämligen Libraries, Applications och Controllers. I Libraries ligger alla bibliotek som en tillämpar för sitt projekt. Det finns färdiga bibliotek i Control Builder som en programmerare kan hämta och utnyttja i sin kod. I Application ligger dels de bibliotek som en programmerare har tillämpat och dels de olika filer där koden ligger. Filer som diagramtyper, functions diagram osv. I Controller ligger hårdvaran som är kopplad till applikationen. [1][5]

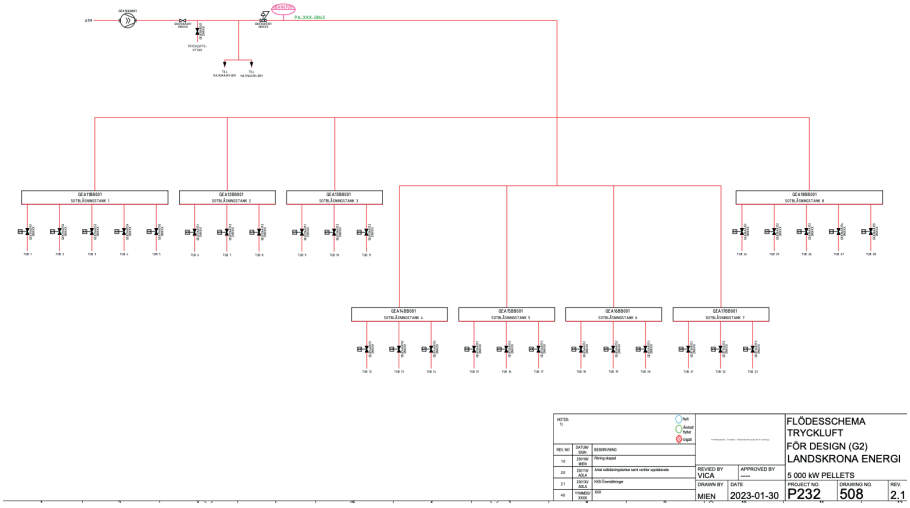
2.8 P&I-schema och funktionsbeskrivning

Ett PI-flödesschema är en visuell bild av hur ett system eller anläggning är uppbyggd och är tänkt att visa hur systemen är uppsatta, hur olika sorters utrustning är placerad, hur styrning och processer fungerar samt för att få en generell förståelse för schemat med hjälp av en funktionsbeskrivning, som beskriver varje figur och varje text i schemat i detalj.

Funktionsbeskrivningen är en dokumenterad beskrivning som är till för att förklara i detalj hur varje utrustning är tänkt att fungera, samt vilka processer som måste implementeras i PLC. Beskrivningen innehåller specifika instruktioner gällande om systemet ska ge sig från signaler för olika funktioner, till exempel att en temperaturgivare ska ge ifrån sig sin status och beroende på villkoren om det är en acceptabel status, ha möjligheten att börja elda på. Funktionsbeskrivningen är användbar för en programmerare (praktiskt taget nödvändig) för att kunna programmera något över huvud taget. Den ger också en tydlig bild för programmeraren hur programmet ska struktureras, hur det ska vara begripligt generellt, hur processorkraften inte ska påverkas negativt och hur det ska vara lätt att begripa vid felsökning. Detta kommer att beskrivas mer utförligt i Analyskapitlet.

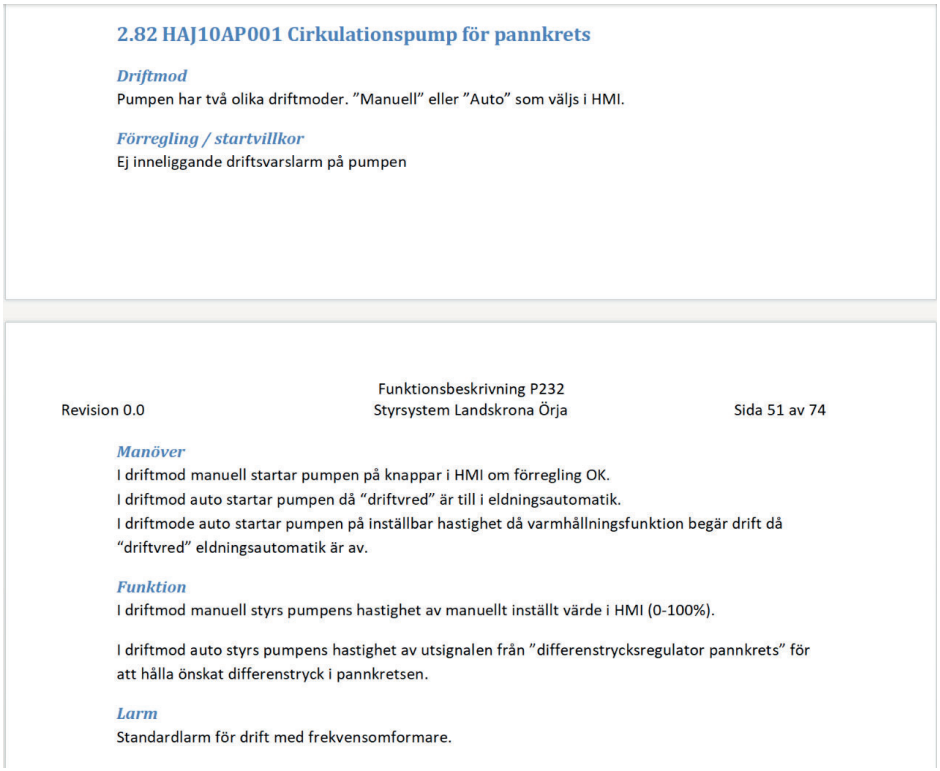
Den innehåller också specifika KKS- namnbeteckningar (se avsnitt 2.7) för varje utrustning. Det är pga. att programmeraren behöver känna till vilka utrustningar eller enheter som tillhör specifika system och på sådant vis få en bättre organisering av objekten som ska hanteras eller programmeras.

Namn: Zaker Taqawi och Mohammed Abou Naasa



Figur 8: Ett PI-flödesschema (P&I Diagram)

Namn: Zaker Taqawi och Mohammed Abou Naasa



2.82 HAJ10AP001 Cirkulationspump för pannkrets

Driftmod

Pumpen har två olika driftmoder. ”Manuell” eller ”Auto” som väljs i HMI.

Förregling / startvillkor

Ej inneliggande driftsvarslarm på pumpen

Manöver

I driftmod manuell startar pumpen på knappar i HMI om förregling OK.
I driftmod auto startar pumpen då “driftvred” är till i eldningsautomatik.
I driftmode auto startar pumpen på inställbar hastighet då varmhållningsfunktion begär drift då “driftvred” eldningsautomatik är av.

Funktion

I driftmod manuell styrs pumpens hastighet av manuellt inställt värde i HMI (0-100%).

I driftmod auto styrs pumpens hastighet av utsignalen från ”differenstrycksregulator pannkrets” för att hålla önskat differenstryck i pannkretsen.

Larm

Standardlarm för drift med frekvensomformare.

Figur 9: En funktionsbeskrivning av en cirkulationspump för pannkrets

I fig. 9 visas en funktionsbeskrivning för en cirkulationspump. Varje objekt som ska styras eller regleras har en sådan funktionsbeskrivning. I driftmod står i vilka lägen de olika objekten ska drivas. I förregling/startvillkor finns ett antal villkor som ska uppfyllas för ett objekt för att det ska kunna starta. I manöverdelen står det hur objektet ska sättas igång i de olika lägena. I funktionsdelen beskrivs hur objektet ska styras/regleras i de olika lägena. Slutligen i larmdelen står hur olika larm ska ske vid fel eller störningar.

3 Metod

3.1 Förberedelser

Efter att ha förstått vad PLC-programmering av pelletspannan i Örja handlade om, påbörjades en rad diskussioner och möten om hur arbetet skulle bedrivas med både varandra och chefen Conny Franzon. Då gällde det att skaffa information om hur arbetet skulle stegvis gå till via inläsning av de mjukvarulicenser som skulle användas, samt komma upp med en tids- och projektplan för jobbet och förbereda exjobbsrapporten. En annan bra förberedelse var att få tillgång till ett tidigare funktionsbeskrivning och P&I-schema för två kraftverk från både Knivsta och Tandådalen, vilka studerades noggrant för att få en uppfattning av hur arbetet skulle genomföras, samt vilka likheter som kunde hittas.

3.2 Utbildning

För att programmeringen skulle kunna genomföras så effektivt som möjligt, så fick det ordnas en utbildnings-workshop på Deterministic Control ABs lokal för att göra sig bekant med platsen, servern/datorer som skulle användas, mjukvaror samt process-grafik. Med hjälp av en erfaren och duktig person som heter Claes Ohlson, så kunde examensarbetarna påbörja projektet, samt visa hur en applikation ställs upp i miljön, och hur programmeringen ska påbörjas.

Under utbildningens gång introducerades Diagram Editorn Control Builder under två veckor och Function Designer i en vecka (se avsnitt 2.6 och 2.8). Under utbildningen fick studenterna möjlighet att genomföra ett mindre projekt för att bekanta sig med olika editors och Process Graphics.

3.3 Programmering

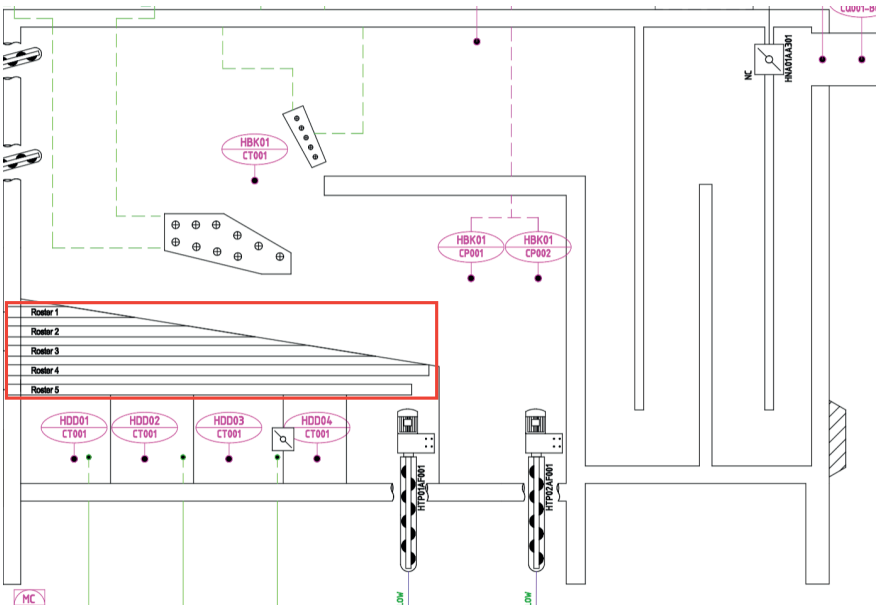
Första planen var att programmeringen skulle ske i Control Builder, men planen ändrades sedan på grund av att slutkunden föredrog Function Designer, för att det ansågs vara lättare att hantera. Kunden kan lätt komma in i koden via processgrafiken och den här egenskapen fattas i Control Builder. En annan anledning var också att kunden har haft Function Designer sedan innan och de är vana vid det här programmet. Därför bestämdes att programmera i

Function Designer i ett senare läge. Dessutom gjordes programmering på ett så tydligt sätt som möjligt, detta möjliggjordes genom att mycket kommentarer lades in i koden.

De system som presenteras här, är system som utgör stor del av pannan. Dessa system har varit de mest utmanande i hela programmeringsfasen och mycket tid har behövt läggas ner. Beskrivningen här är att visa vad som var utmanande med de olika systemen och funktionerna och hur de programmerades.

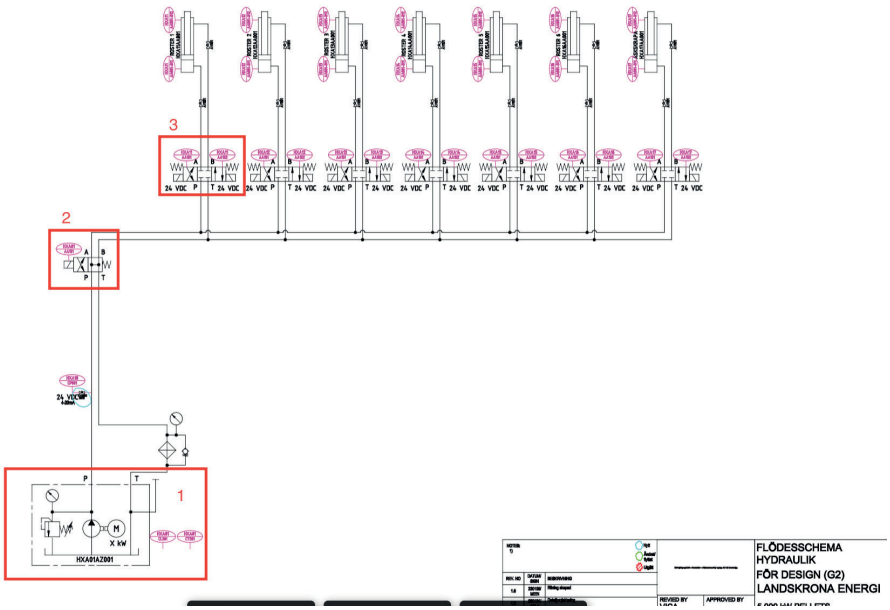
3.3.1 Roster och Askskrapa

Roster utgör en betydelsefull komponent inom pannsystemet. De är belägna vid ugnens botten (se figur 12), och har som främsta funktion att successivt föra fram bränslet in i ugnen, vilket möjliggör en effektiv förbränning av bränslet till aska. Efter förbränningen tas askan om hand av särskilda askskruvar. I den här pelletspannan används 6 roster och en askskrapa och varje roster och askskrapan utnyttjar dubbelverkande hydraulcylinder[2].



Figur 12: Illustration av en roster (rödmarkerad) i en ugn

[2] En cylinder som överför hydrauliska krafter i båda riktningarna. Sådana cylindrar har både dragande och tryckande kraft.

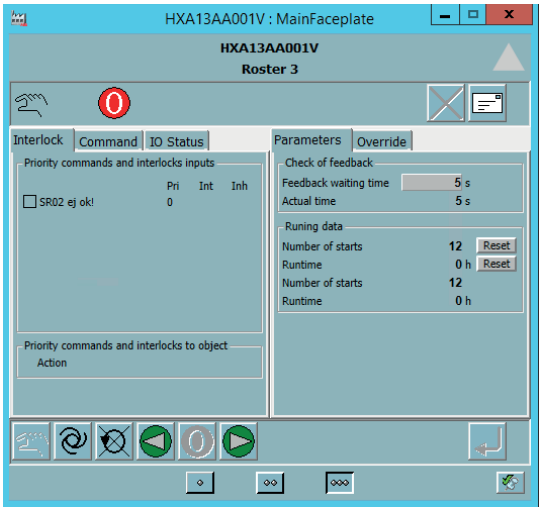


Figur 13: P&I-schema för roster och askskrapa

1. Hydraultank med hydraulmotor och några nivågivare. Hydraulmotorn ska drivas varje gång någon roster eller askskrapan begär drift, enligt funktionsbeskrivning.
2. Enkelriktad cylinder som endast ska aktiveras när någon roster eller askskrapan begär drift.
3. Dubbelverkande hydraulcylinder som kan drivas i bägge riktningar, dvs fram och bak.

Varje roster har en egen KKS, roster 1 (HXA11AA001), roster 2 (HXA12AA001) och så vidare. Askskrapan har en egen KKS (HXA17AA001). Dessa KKS ska tillämpas som namn för det objekt som tillämpas för roster och askskrapa, så att när operatörerna tar fram faceplate så att de kan se KKS:en.

Namn: Zaker Taqawi och Mohammed Abou Naasa



Figur 14: Faceplate på ett av roster objekt

2.60 HXA11AA001 Roster 1

Driftmod

Roster 1 kan vara i två olika driftmoder. "Manuell" eller "Auto" som väljs i HMI.

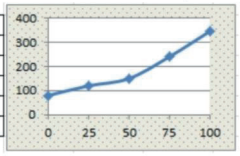
Funktion

I driftmod manuell styrs magnetventiler för roster med hjälp av manuella knappar i HMI för "mot vändläge" och "mot hemmaläge". Om man kör den i läge "manuellt kontinuerligt" så fungerar den som i auto-läget men med ett fast inställbart antal slag per timme.

I driftmod auto styrs roster med X antal slag per timme beroende av mastersignal. Ett slag med roster innebär att den går framåt i inställd tid, dvs roster gör inte hela slag utan den pulsas framåt. När den

når sitt främre gränsläge så vänder den och kör ända hem i ett svep.
Vid riktningsväxling fördröjs magnetventiler 500ms för att undvika slag i hydraulsystem.
Aktuellt antal slag/timme skalas fram beroende av inställda värden i HMI för antal slag vid 0, 25, 50, 75 och 100% master (master-värden är låsta till dessa värden):

Master (%)	*slag/timme*
0	80
25	120
50	150
75	240
100	345



Aktuellt antal slag per timme skalas fram linjärt beroende av aktuell mastersignal och inställda värden. Detta värde används sedan för att beräkna fram "pausetid mellan slag". Paustid mellan slag beräknas kontinuerligt och jämförs med aktuell pausetid som räknas i systemet. Då aktuell pausetid är större än beräknad pausetid sätts minne för att "slag önskas" och timer för pausetid nollställs och börjar om. När hydraulen är "ledig" dvs inga andra drifter håller på med ett slag, utförs slaget.

Om panna går in i läge fyrhållning, går roster med speciellt inställt värde för "Antal slag per timme vid fyrhållning".

Larm

B-Larm erhålls vid max gångtid riktning mot vändläge.
B-Larm erhålls vid max gångtid riktning mot hemmaläge.

B-Larm erhålls vid max antal tryckvändningar mot vändläge.
B-Larm erhålls vid max antal tryckvändningar mot hemmaläge.

Figur 15: Funktionsbeskrivning av en roster

Enligt funktionsbeskrivningen ska roster kunna köras både manuellt och i auto. I auto ska masterregulatorn[3] reglera antal slag som roster ska utföra, samtidigt ska rostern drivas i manuellt kontinuerligt läge, dvs rostern är i auto men operatören matar in antal slag per timme. Ett annat läge är fyrhållningsläge[4] som alltid ska få prioritet när den aktiveras.

Följande färdiga objekt i ABBs bibliotek användes för programmeringen av roster utom:

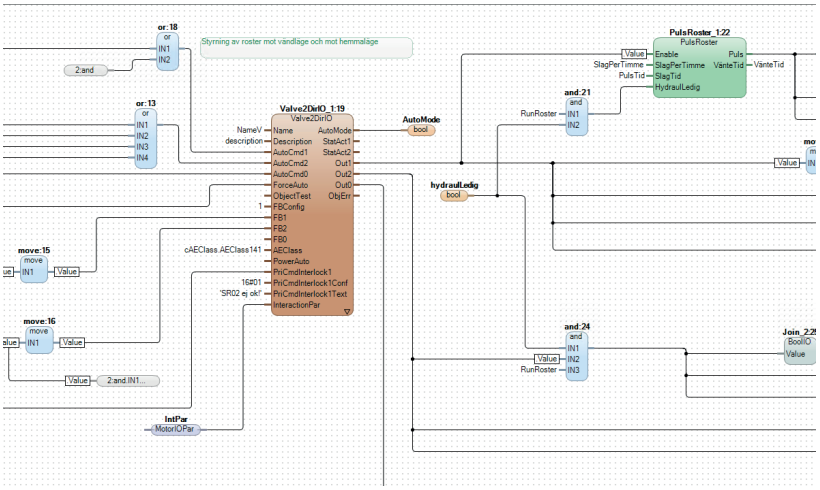
- Valve2DirIO : Ett dubbelriktad ventilobjekt, öppna 1, öppna 2 och stäng.
- ValveIO : enkelriktad ventil, öppen och stäng.
- MotorOnOffIO : Motorobjekt för hydraulmotorn.
- Piecewiselinear : Ett XY-skalningsblock.
- SelCC : Selector som väljer de olika driftlägen (ex. manuell, auto och m.m).
- SignalInBoolMLogik : Ett signal block som används för att generera larm.

Enligt funktionsbeskrivningen ska rostern pulseras när den kör framåt och köra tillbaka på ett enda svep. För detta gjordes ett funktionsblock som programmerades med strukturerad text (se figur 17 nedan). Funktionsblocket tar in ett realvärde från SelCC, som sedan utnyttjas för att beräkna paustid. Blocket kontrollerar också om hydrauliken är ledig, dvs om inte någon annan roster kör framåt eller bakåt. Om den är ledig så pulsar rostern framåt och sedan väntar på sitt nästa puls tills paustiden löper ut. Det finns dessutom ett antal larm för att undvika fel och störningar i systemet.

Programmeringen av roster har varit en av de svåraste delarna i detta projekt. En av anledningarna kan vara att funktionsbeskrivningen inte var så pass tydlig som den skulle ha varit. Dessutom var det alltid någon del i koden som inte fungerade korrekt, exempelvis enligt funktionsbeskrivningen ska en roster utföra ett slag åt gången när hydrauliken var ledig, men istället möjliggjorde koden att alla 6 roster kunde ta över hydrauliken och utföra ett slag samtidigt. Vilket var fel eftersom funktionsbeskrivningen vill endast att en roster utför ett slag åt gången. Detta på grund av att hydraulmotorn bara har kraft att hantera en roster i taget. En annan anledning var Objektet Valve2DirIO betedde sig som en motor och inte som en två riktningsventil. Det var felkonstruerat och ingen programmerare som hade använt detta objekt i något föregående projekt. Slutligen löstes problemet med hjälp av handledaren Claes

[3] En regulator som aktiveras i eldningsatuomatik steg 7 (se avsnitt 3.3.1) och som reglerar antal slag per timme som roster ska utföra sitt slag.
[4] Fyrhållningsläge är ett min-last-läge för att förhindra övertemperatur.

Olsson. Lite fel i logiken som behövde åtgärdas i både pulsRoster objektet och i diagrammet där koden fanns.



Figur 16: En del av kod på roster (valve2DirIO, PulsRoster och lite logik)

	Name	Data Type	Attributes	Direction	FD Port	Initial Value	Description
1	Enable	bool	retain	in	yes		Enable pulsfunktionen
2	SlagPerTimme	real	retain	in	yes		Slag per timme (sekunder)
3	SlagTid	real	retain	in	yes		Pulstid
4	HydraulLedig	bool	retain	in	yes		Hydaul är ledig att användas
5	Puls	bool	retain	out	yes		Puls som skickas ut till rostern

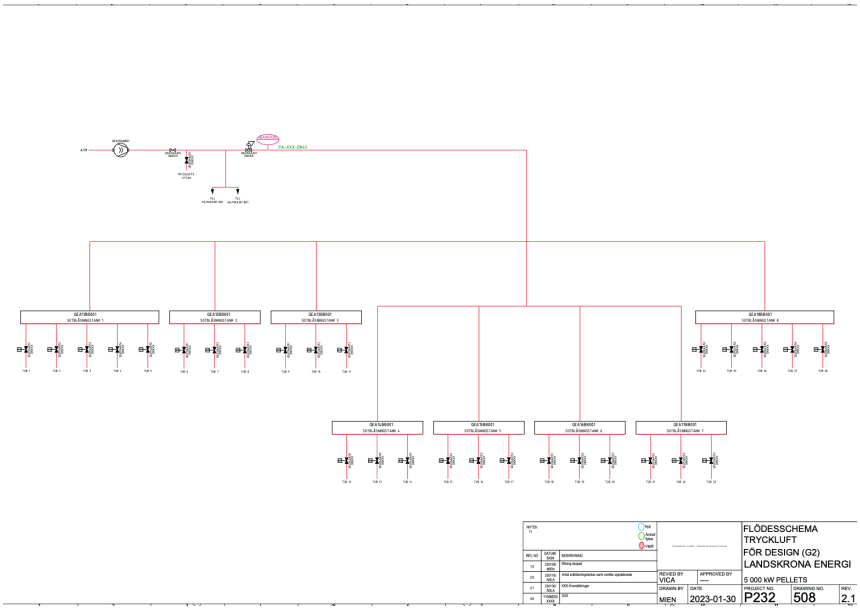
Parameters | Variables | External Variables | Function Blocks

```
FirstScan := FirstScanAfterApplicationStart();

(* Startpulsen aktiveras endast en gång varje gång out från ventil2DirIO blir aktiv *)

StartRisingEdge( Clk := Enable );

SRMinne( S1 := StartRisingEdge.Q,
         Reset := Puls );


StartPuls( Clk := SRMinne.Q1 and HydraulLedig );


(* Beräkning av paustid mellan slag *)
pausTid := real_to_time(_3600s / SlagPerTimme * 1000); (* omvandling i ms *)


(* Mätning av paustid *)
timerMät( In := not FirstScan and not timerPuls.Q and Enable and not SRMinne.Q1,
          PT := _1h,
          ET => timerMät_ET );

(* Flank paustid uppnådd *)
paustidUppnådd( Clk := timerMät.ET >= pausTid and Enable);


(* Aktivera minne Slag önskas *)

SlagÖnskas1( S1 := paustidUppnådd.Q,
             Reset := Puls );

pulsTrigg( Clk := SlagÖnskas1.Q1 and HydraulLedig );

timerPuls( In := pulsTrigg.Q or StartPuls.Q,
           PT := real_to_time(SlagTid * 1000) );


Puls := timerPuls.Q and Enable and HydraulLedig;

(* vänteTid i sekunder, begränsad till positivt värde *)

vänteTid := abs( Min( ( time_to_real(timerMät.ET) - time_to_real( PausTid ) ) / 1000.0, 0.0 ) );
```

Figur 17: PulsRoster funktionsblock som programmerades med strukturerad text för pulsning av roster

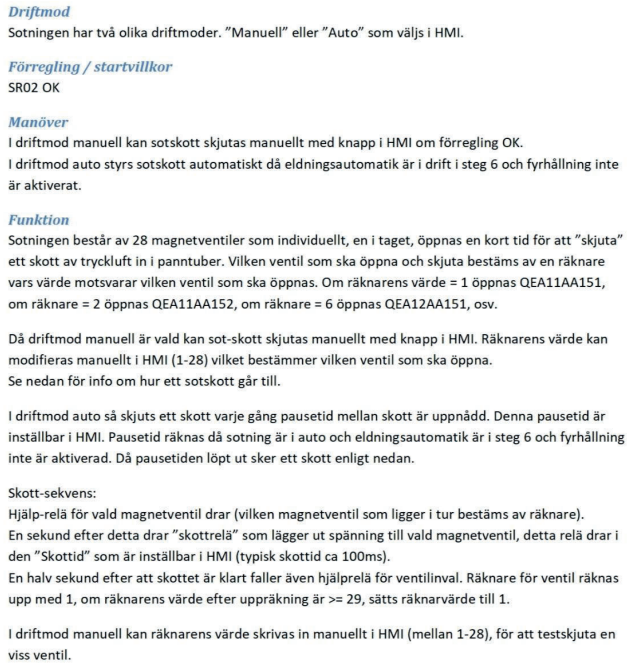
3.3.2 Luftsotning panna

Luftsotning är en effektiv metod för att minska kväveoxider (NOx-utsläpp) från fjärrvärmeverk och används alltmer i moderna fjärrvärmeverksanläggningar. Luftsotning består av 8 sotblåsningstankar, tank 1 och 8 har 5 ventiler vardera, medan 2-7 har 3 ventiler vardera. Totalt 28 ventiler som individuellt, en i taget, ska öppna en kort tid för att skjuta ett skott av tryckluft i panntuber, enligt funktionsbeskrivningen nedan.



Figur 18: P&I-schema för luftsotning panna

Namn: Zaker Taqawi och Mohammed Abou Naasa



Driftmod
Sotningen har två olika driftmoder. ”Manuell” eller ”Auto” som väljs i HMI.

Förregling / startvillkor
SR02 OK

Manöver
I driftmod manuell kan sotskott skjutas manuellt med knapp i HMI om förregling OK.
I driftmod auto styrs sotskott automatiskt då eldningsautomatik är i drift i steg 6 och fyrhållning inte är aktiverat.

Funktion
Sotningen består av 28 magnetventiler som individuellt, en i taget, öppnas en kort tid för att ”skjuta” ett skott av tryckluft in i panntuber. Vilken ventil som ska öppna och skjuta bestäms av en räknare vars värde motsvarar vilken ventil som ska öppnas. Om räknarens värde = 1 öppnas QEA11AA151, om räknare = 2 öppnas QEA11AA152, om räknare = 6 öppnas QEA12AA151, osv.

Då driftmod manuell är vald kan sot-skott skjutas manuellt med knapp i HMI. Räknarens värde kan modifieras manuellt i HMI (1-28) vilket bestämmer vilken ventil som ska öppna.
Se nedan för info om hur ett sotskott går till.

I driftmod auto så skjuts ett skott varje gång pausetid mellan skott är uppnådd. Denna pausetid är inställbar i HMI. Pausetid räknas då sotning är i auto och eldningsautomatik är i steg 6 och fyrhållning inte är aktiverad. Då pausetiden löpt ut sker ett skott enligt nedan.

Skott-sekvens:
Hjälp-relä för vald magnetventil drar (vilken magnetventil som ligger i tur bestäms av räknare).
En sekund efter detta drar ”skottrelä” som lägger ut spänning till vald magnetventil, detta relä drar i den ”Skottid” som är inställbar i HMI (typisk skottid ca 100ms).
En halv sekund efter att skottet är klart faller även hjälprelä för ventilinval. Räknare för ventil räknas upp med 1, om räknarens värde efter uppräkning är >= 29, sätts räknarvärde till 1.

I driftmod manuell kan räknarens värde skrivas in manuellt i HMI (mellan 1-28), för att testskjuta en viss ventil.

Figur 19: Funktionsbeskrivning av luftsotning

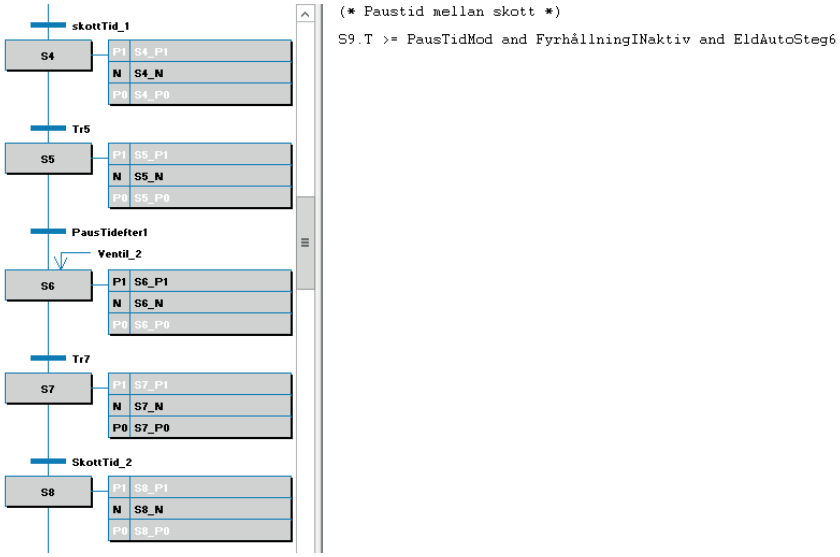
Luftsotningen ska kunna köras i både manuellt och autoläge. I autoläge aktiveras luftsotningen när eldningsautomatik är i steg 6 och fyrhållningen är inaktiverad. Dessutom ska operatörerna kunna testskjuta en viss ventil i manuellt läge beroende på vad de matar in.

Det finns ett antal färdiga objekt i ABBs bibliotek som användes för programmeringen av luftsotning, nämligen följande:

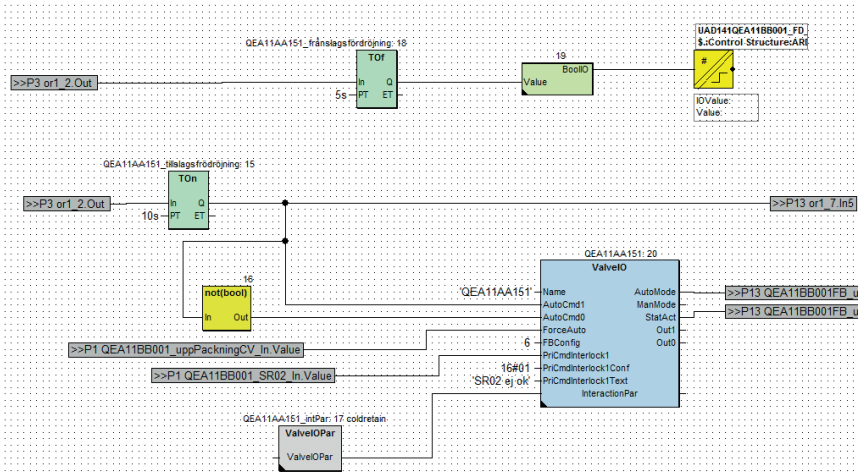
- ValveIO : Ett ventilobjekt som enbart kan öppnas och stängas.
- SetValue : Ett inmatningsfält som operatörerna kan knappa in mellan 1-28, beroende på vilken ventil som de vill, ska starta från i autoläge eller som de vill testskjuta i manuellt läge.

Vid implementeringen av styrningen för luftsotning har det rekommenderats av handledare att använda Sequential Function Chart (SFC). Detta rekommenderas på grund av att luftsotningsprocessen sker i en följd av steg där ventiler skjuter ett skott (Öppna och stänga en kort tid) i ordning. Det är nämligen om operatörerna har valt att starta sotningen från ventil 1 i autoläge, kommer ventil 2 inte öppna förrän ventil ett har skjutit ett skott, sedan avvaktar ventil 2 en kort paustid innan det utför ett skott och så vidare.

Programmeringen av sekvensen har strukturerats på ett sådant sätt att alla objekt sätts i autoläge redan vid steg 1. Detta görs för att säkerställa att ventilen reagerar på kommandon som ges från sekvensen. Detta säkerställs i stegvillkoret om allting har satts i autoläge före steg 2. Därefter beroende på vilken ventil som har valts hoppar sekvensen till det valda steget för att påbörja förskjutningen av ett skott. I figur 19 ovan beskrivs hur ett skott ska ske. Ventilen öppnar ca 100 ms när båda hjälprelä och skottrelä har dragits.



Figur 20: SFC programmering av luftsotning för ventil 1



Figur 21: ValveIO objektet som är kopplad till SFC:en ovan i figur 20

Luftsotningen var utmanande eftersom det var 28 ventiler som skulle tas hand om i en enda sekvens. Det blev sammanlagt 113 steg. Det som var mest utmanande med den här delen av pelletspanna var att i ett 800xA-system är det endast tillåtet att ha 32 kommunikationsvariabler[5] i ett enda diagram. Sekvensen i varje steg skickar ett kommando till ventilen att öppna och sedan avvaktar sekvensen ett svar från ventilen innan sekvensen ska hoppa till nästa steg. Det innebär att i det diagram där sekvensen kommer att befinna sig att kommandot ska skickas via kommunikationsvariabler till alla 28 ventilerna och dessutom få 28 svar från dessa ventiler. Totalt blir detta 56 kommunikationsvariabler, vilket överskrider gränsen i 800xA-systemet. För att motverka detta gjordes två datatyper per tank (se P&I-schema i figur 18). En datatyp för att skicka kommandot och en annan för att ta emot ett svar (feedback om ventilen är stängd eller öppen). Sammanlagt blev det 16 datatyper. Ett diagram skapades per tank och där låg alla ventiler som tillhörde tanken, exempelvis tank 1 hade 5 ventiler, ett diagram skapades för tank 1 och i det diagrammet lades in alla 5 ventiler som tillhörde tank 1. Nu kunde sekvensen kommunicera med varje tank diagram via två kommunikationsvariabler kopplade till datatyper som låg i tankdiagrammet. Den ena datatypen överförde kommandot från sekvensen vidare till alla ventiler som tillhörde tanken. Den andra datatypen överförde svar (feedback) från alla ventiler till sekvensen. På detta sätt undveks alltför många kommunikationsvariabler i ett enda diagram. Från sammanlagt 56 kommunikationsvariabler minskades detta till 16, dvs 2 per tank diagram.

3.3.3 Skruvar

Det finns ett antal skruvar i pelletspannan som har olika roller. Det finns bränsleskruvar som ser till att kunna mata in pellets till stokerskruvar, stokerskruvar som för vidare pellets rakt in i ugnen för förbränning och till sist askskruvar som matar ut askan efter förbränning. Vad skruvarna har gemensamt är att varje skruv har en liten låda ute i fältet för lokalstyrning[6], med en knapp för Start order och en driftvred där det går att vrida till framkörning, back körning eller sätta den i fjärrläge.

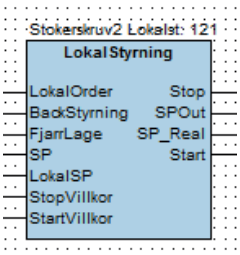
Minst sagt så har programmeringen av lokalstyrningen för dessa skruvar varit den mest

[5] **Kommunikationsvariabel:** Detta innebär två objekt som befinner sig i två olika diagram men de är kopplade till varandra. De kopplas via en kommunikationsvariabel så att båda objekten kommunicerar med varandra.
[6] **Lokalstyrning:** Detta innebär att det finns en växellåda ute i fältet där drifttekniker kan gå fram och lokalt styra processen via knappar eller vred.

utmanande delen för dessa system. Motorobjekten som ska symbolisera frekvensomriktare ute i fälten, har inställningar för att kunna indikera eller aktivera lokalstyrning av motorn. Problemet är att ABB stödjer inte funktionaliteten för lokalstyrningen, trots att inställningar för det existerar. Det innebär att följande problem måste åtgärdas:

- På egen hand måste det tillämpa logik som tar hand om indikeringen att någon har aktiverat lokala styrningen ute i fältet och skicka signalen till operatörens skärm, t.ex ett larm via en SignalInBoolMLogik (Ett objekt i ABBs bibliotek).
- Då den lokala styrningen är aktiverad, måste styrningen ta över kontrollen av processen så att operatören som sitter i operatörsrummet inte har förmågan att styra processen under tiden som någon är ute i fältet.

När logik väl tillämpades och testades, så blev koden väl fyra sidor lång. Det är för många sidor och det hjälper inte operatören att kunna navigera sig smidigt genom kod om de skulle behöva göra det. Därför löstes problemet genom att koda in all logik via en egen typkrets objekt. Se figur nedan.

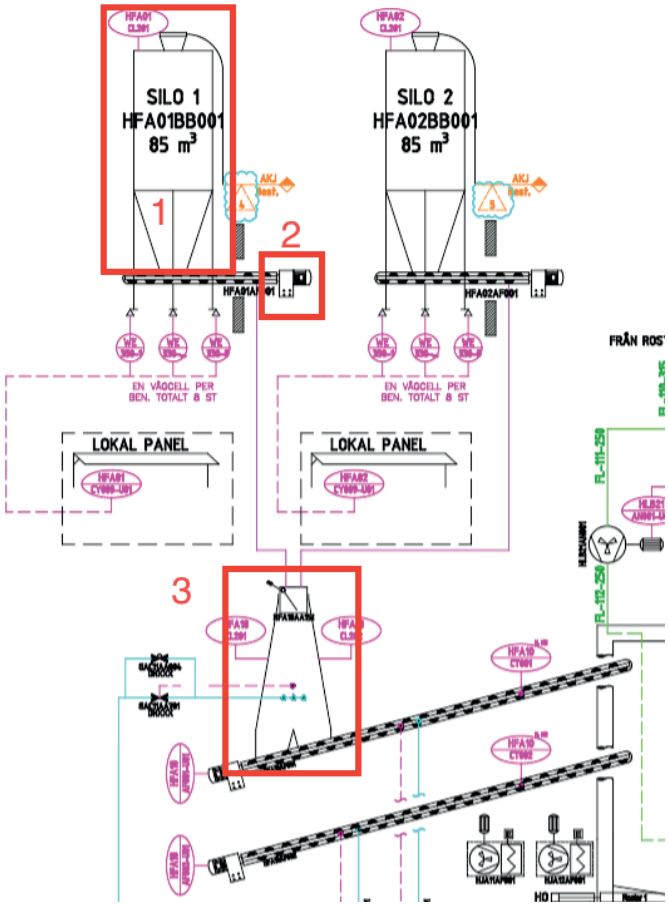


Figur 22: Exempel på hur en typkretsobjekt ser ut.

Detta objektblock ersatte fyra sidor kod. Dessa objekt kan bara skapas via Control Builderns bibliotek och importeras in till Function Designer. Detaljbeskrivning av objektblocket kommer inte att beskrivas.

3.3.3.1 Bränsleskruvar

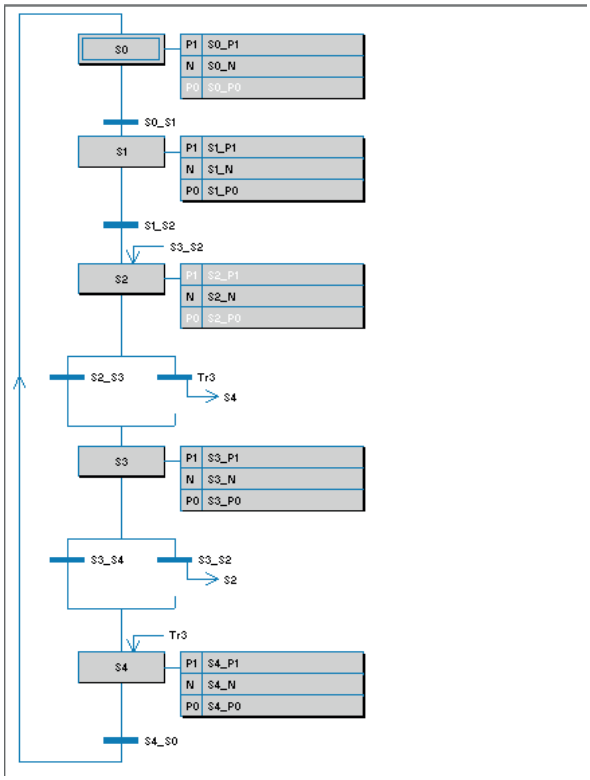
Bränsleskruvar används för att förse stokerbehållare med pellet från silos och det styrs i sin tur av en annan sekvens. Det är nämligen en bränsleinmatningssekvens som också har programmerats med SFC.



Figur 23: P&I-schema visar hur silo, bränsleskruvar och stokerbehållaren är anknutna med varandra

Bränsleskruvar har till uppgift att fylla på stokerbehållare med bränslet som sedan förs vidare via stokerskruvar till ugnen. Bränsleinmatningssekvensen tar hand om styrningen. Den består av 4 steg. Bränsleinmatningsekvens aktiveras när eldningsautomatiken är i steg 6. Bränsleinmatningsekvensen ställer alla tillhörande objekt, som bränsleskruvar, i autoläge. Därefter avvaktar sekvensen i steg 2 tills nivåvakterna i stokerbehållaren indikerar låg nivå. När sekvensen är i steg 2, ligger ett annat stegvillkor parallellt med stegvillkoret för att hoppa in i steg 3, som hela tiden kontrollerar så att eldningsatuomatiken är i steg 6. Om det inte är fallet, hoppar sekvensen till steg 4 och där stoppas hela sekvensen. Detta i sin tur stoppar bränsleskruvarna. Däremot om eldningsautomatiken är i steg 6 och så fort nivåvakterna indikerar låg nivå hoppar sekvensen i steg 3, där skickas en kommando till någon av

bränsleskruvarna beroende på vilken som är primär och sekundär, som sedan börjar mata in stokerbehållare med bränsle tills nivåvakter indikerar ej låg nivå. (Se figur 24 nedan)

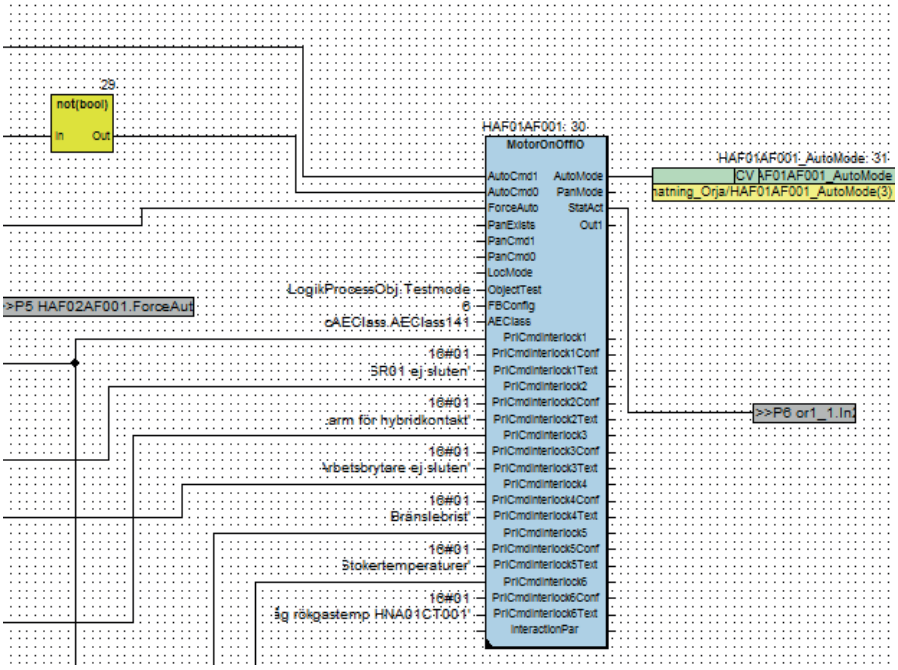


Figur 24: Bränsleinmatningssekvensen som programmerades med SFC och består av 4 steg

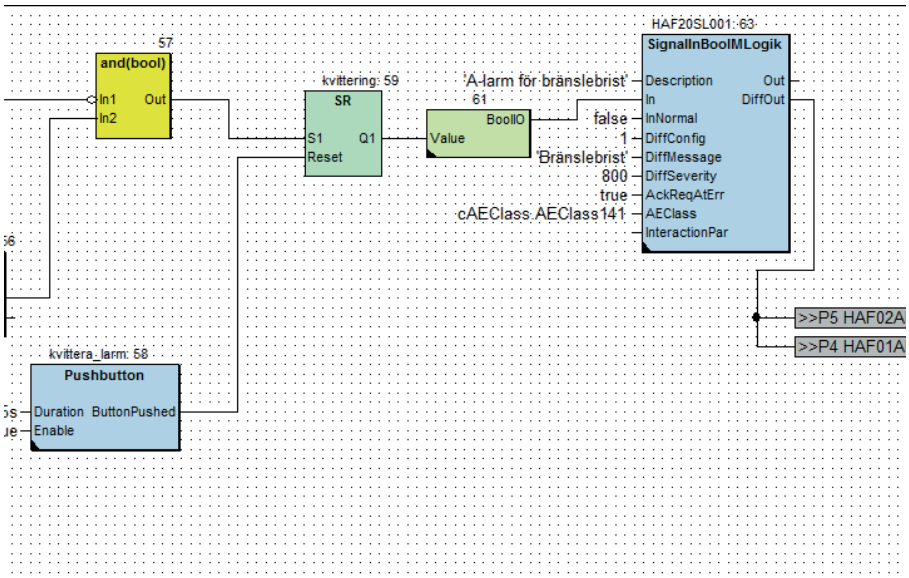
För bränsleskruvar som har en KKS (HFA01AF001 och HFA02AF001) tillämpades följande objekt från ABBs bibliotek:

- MotorOnOffIO : Ett on/off motorobjekt som tillämpas för motorer, pumpar och m.m.
- SignalInBoolMLogik : Ett signalblock som tillämpas för att generera larm vid fel och störningar.

I figur nedan visas ett motorOnOffIO-objekt som får in kommandon från bränsleinmatningssekvens beroende på om det ska köra eller ej. Dessutom syns ett antal PriCmdInterlock som tar hand om startvillkor, som måste vara uppfyllda, för att motorn ska driva bränsleskruven.



Figur 25: Motorbjektet för bränsleskruv 1 och som är kopplad till bränsleinmatningssekvensen



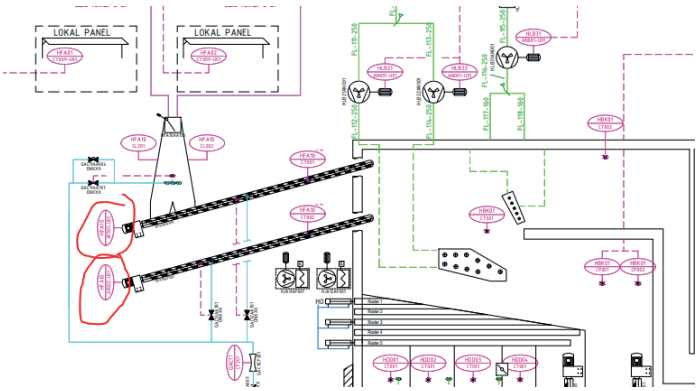
Figur 26: Logik för larmgenerering vid fel (Larm block SignalInBoolMLogik)

Det var som utmanande med bränsleinamtningen var autoväxlingen, det vill säga automatisk växling mellan bränsleskruvar. Det finns 2 stora silos som är fyllda med pellets (se figur 23).

Varje silo har en egen bränsleskruv som för över pellets till stokerbehållaren. Endast en av bränsleskruvarna kan vara i drift åt gången. För detta programmerades en separat logik som tar hand om autoväxlingen. Två knappar tillämpades för att välja någon silo som primär. Två ytterligare knappar utnyttjades för att aktivera autoväxlingen eller inaktivera. Om bränsleskruv för silo 1 är i drift eller valt som primär, samtidigt är autoväxlingen också aktiverad och det blir låg nivå i silo 1, startas bränsleskruv för silo 2 och bränsleskruv för silo 1 stängs. Om autoväxlingen är inaktiverad och det blir låg nivå i silo 1 som har valt som primär av en operatör, kommer motorn att fortfarande vara i drift enligt funktionsbeskrivning. (se figur 31)

3.3.3.2 Stokerskruvar

Mycket av bränsleinmatningen sker tack vare stokerskruvar, som för vidare pellets rakt in i ugnen för att förbrännas.



Figur 27: Exempelbild av Stokerskruv motorer

Skruvarna styrs av frekvensomriktare med objekttypen MotorVarSpeedBus[7] och har beteckningarna HFA10AF001 respektive HFA10AF002. Skruvarna har inkopplade temperaturgivare som mäter temperaturen på stokern och är säkerhetsklassade. De har också inbyggda stora kylfläktar som körs simultant med omriktare för att kyla ner motorerna.

De styrs utifrån masterregulatorns utsignal, som har gränser att köra drifter mellan 0 till 100%. För stokerskruvarna så används signalen till att bestämma vilken hastighet som ska

[7] Motor Objekt som resemblar verklig omriktare med Bus koppling.

köras i kontinuerlig drift och vilka paus och puls tider som ska finnas vid intermittent drift. Detta kan åstadkommas via en Piecewiselinear objektblock.[8]

Det finns ett inställbart gränsvärde som jämförs med masterregulatorns insignal. Om signalen understiger gränsvärdet, så kör skruvarna i intermittent drift. Om signalen överstiger värdet, drivs skruvarna kontinuerligt. Denna funktionaliteten kan lösas med hjälp av ett Selector objekt, ett SetValue objekt och ett GreaterThanValue objekt. Se figur 4.1.2 för funktionsbeskrivning.[9]

Vid andra driftfall där till exempel fyrhållning är aktiverad eller att temperaturgivarna för skruvarna visar alltför hög temperatur, så behövdes det kodas specialfall där special paus och pulstiden körs intermittent vid dessa situationer. Den största svårigheten var att försöka lista ut vilket utav drift fallen som ska först prioriteras och i så fall, sätta korrekt paus och puls tider. Vid diskussioner med kollegor så bestämdes det att följande prioritet skulle tillämpas:

1. Hög temperatur för skruvarna har högst prioritet
2. Fyrhållning har näst högst prioritet
3. Normal driftfall har lägst prioritet

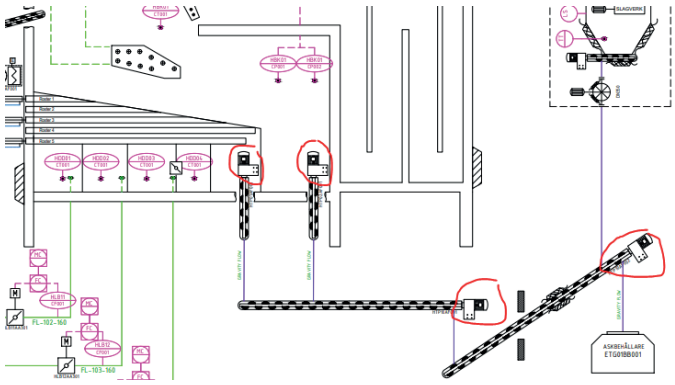
Slutsaten kommer från att fyrhållning inte har så stor betydelse för skruvarna, medan hög temperatur kan leda till brandfara. Med den slutsatsen så behövdes flera grindar av typen AND och OR som såg till att prioriteten uppehölls.

3.3.3.3 Askskruvar

Askskruvar ser till att mata ut askan till en askbehållare då antingen den sjätte rostern eller askskrapan har utfört ett antal slag och det blir läge för att mata ut askan från ugnen. System-beteckningen för dessa skruvar är HTP och primärt används HTP10AF00-1 och 2 skruvar. Beroende på om askan matas ut från askdiket eller från under rostern så aktiveras

[8] Ett objekt som skapar linjära X och Y värden i en börvärdeskurva.
[9] Selector - ett objekt som väljer mellan två värden beroende på vilka villkor som gäller.
Setvalue - objekt som ger möjlighet att ställa in värden i grafik.
GreaterThanValue - objekt som kontrollerar vilket inkommande värde är högre.

antingen skruv HTP01AF001 eller HTP02AF001.



Figur 28: Exempelbild av Askskruvar. Längst till höger är HTP10AF001- och 2. Längst till vänster är skruvar från ugnen.

Askutmatningen sker i en sekvens där HTP10AF002 startas och sedan HTP10AF001 och till sist något utav ovanstående skruvar beroende på var askan kommer ifrån. Naturlig lösning för att åstadkomma denna utmatningssekvensen är att använda sig av SFC som programmeringsspråk. Det största utmaningen med den här systemet var att implementera logik som såg till att sekvensen:

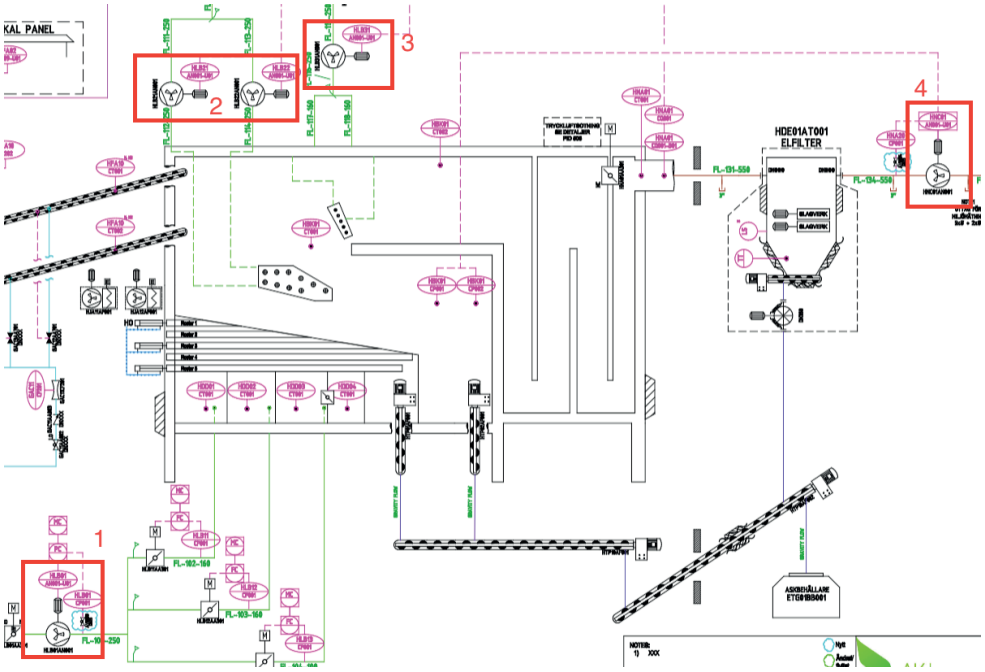
- Kunde köras manuellt flera gånger vid start knappar och inte köra på egen hand.
- Såg till att sekvensen faktiskt kunde stoppas vid behov och inte köras på egen hand.
- Logiken som implementerades, inte stör annan implementerad logik.

3.3.4 Fläktar

Det finns fyra olika sorters fläktar som hjälper till med förbränningen av pellets:

1. Primärluftsfläkt - HLB01AN001
2. Två sekundärluftsfläktar - HLB21(22)AN001
3. Tertiärsluftsfläkt - HLB31AN001
4. Rökgasfläkt - HNC01AN001

Namn: Zaker Taqawi och Mohammed Abou Naasa



Figur 29: 1 primärluftsfläkt, 2 sekundärluftsfläkt, 3 tertiärluftsfläkt och 4 rökgasfläkt

Primärluft innebär att luften från denna fläkt går rakt in i brasan under elden. Sekundärluften kommer från en särskild kanal som leds in i ugnen för att se till att ångan eldas på och blir tillräckligt varmt för att kunna hetta vatten. Sekundärluftsfläktar styrs dels utifrån masterregulatorns insignal, och dels också av en syrehalts regulatorn.

Syrehaltsregulatorn ser till att reglera sekundärluftsfläktens hastighet beroende på hur mycket syre som tillförs till brasan. Signalerna från både masterregulatorn och syrehaltsregulatorn adderas och skickas till fläkten. För en utav sekundärluftsfläkten så finns det en inställbar multiplikationsfaktorn som kan justera hastigheten. Regleringen av luftflödet till ugnen sker med dessa fläktar då luften inte strömmar rakt in i brasan som det gör vid primärluftsfläkten.

Tertiärluften används vid slutförbränningen av pellets för att säkerställa att förbränningen har gått bra vid slutskedet. Här används en fläkt och ett spjäll och har en gemensam insignal som hanteras på samma sätt som för sekundärluftsfläkten. Det intressanta med tertiärobjekten är att insignalen till objekten hanteras lite annorlunda, där det finns en brytgräns för hur spjället och fläkten ska styras.

Denna utsignal delas sedan upp till tertiärluftspjäll (HLB31AA301) och till tertiärluftsfläkt (HLB31AN001) beroende av inställd gräns för ”Utsignal då spjäll fullt öppet och fläkt börjar varva upp”. Dvs om utsignalen är under denna gräns går fläkten med min utsignal och spjället regleras 0-100%, om utsignalen är över denna gräns är spjället fullt öppet och fläkten regleras 0-100%.

Figur 30: Funktionsbeskrivning av Tertiärluftsfläkt.

Hanteringen av denna funktion var lite knepig och det var inte helt uppenbart hur det skulle åtgärdas. Handledaren i Deterministic Control AB, Claes Ohlsson, föreslog att splitRange kan tillämpas och en setValue för att kunna ställa in en brytgräns när funktionen i bilden skulle ske.[10]

Rökgasfläkten ser till att gasen som har bildats efter förbränningen, används till att överföra gaserna till en skorsten. Gasen går via en mätare som används för miljömätning och kontrollerar att gaserna är inom rätt nivåer för utsläpp.

3.3.5 Eldningsautomatiken

Eldningsautomatiken är en sekvens av uppstartsfunktioner som är utformad för att initiera eldningsprocessen i en panna. Denna sekvens består av nio olika steg som vart och ett aktiverar en specifik del av pannan, såsom roster, fläktar, bränsleinmatning och andra relevanta komponenter. I funktionsbeskrivningen står alla 9 steg, stegvillkor samt kommentarer. Sammantaget utgör dessa steg en väsentlig del av den automatiserade processen för att på ett effektivt och säkert sätt initiera eldning i pannan. (se figur 10 nedan)

[10] Splitrange - ett objekt som delar en signal till två utsignaler.

Namn: Zaker Taqawi och Mohammed Abou Naasa

3.1 Eldningsautomatik

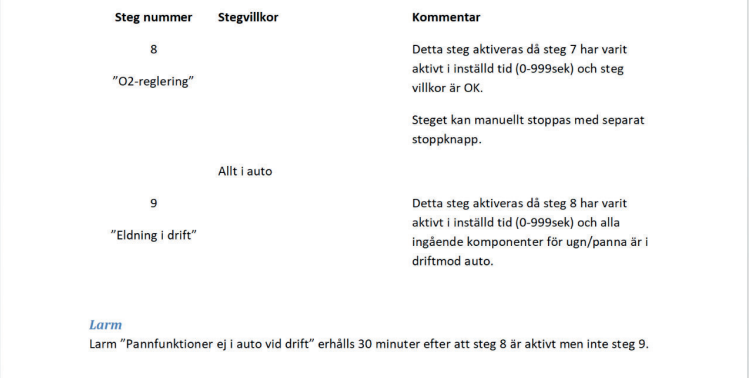
Funktion

Eldningsautomatik är en ”uppstartssekvens” för förbränning i ugnen.
Denna sekvens har 9 steg som var och ett startar upp olika delar på pannan. Tekniskt sett är det inte en sekvens i ordets rätta bemärkelse på grund av att föregående steg hela tiden ligger aktiva tills de stoppas. Dock måste underliggande steg vara aktiva för att nästa ska kunna aktiveras. Dvs om alla steg är aktiva och steg 4 bryts, bryts momentant även steg 5, 6, 7, 8 och 9.
Om eldningsautomatik avbryts till något steg pga av att stegvillkor faller för nästa steg stannar eldningsautomatik tills man återstartar med separat knapp i HMI.

Steg nummer	Stegvillkor	Kommentar
1 ”Driftvred till”		Detta steg aktiveras genom att man aktiverar knappen ”Driftvred till” i HMI. Detta steg är remanent och stoppas endast genom att trycka ”Driftvred från”.
	Säkert stopp rökgasfläkt OK (från säkerhets-PLC).	
2 ”Rökgasfläkt”		Detta steg aktiveras då steg 1 har varit aktivt i inställd tid (0-999sek) och steg villkor är OK. Steget kan manuellt stoppas med separat stoppknapp.
	Driftsvar rökgasfläkt OK. Inga larm avseende rökgasfläkt. Säkert stopp tertiärluftsfläkt OK (från säkerhets-PLC).	
3 ”Tertiärlufts-fläkt”		Detta steg aktiveras då steg 2 har varit aktivt i inställd tid (0-999sek) och steg villkor är OK.

Steg nummer	Stegvillkor	Kommentar
4 ”Sekundärlufts-fläkt”		Detta steg aktiveras då steg 3 har varit aktivt i inställd tid (0-999sek) och steg villkor är OK. Steget kan manuellt stoppas med separat stoppknapp.
	Säkert stopp primärluftsfläkt OK (från säkerhets-PLC). Ej larm Låg-Låg O2-halt HNA01CQ001. Ej larm Hög-Hög ugnstemperatur. Ej larm hög temp framledning (HAJ10CT001/HAJ10CT002)	
5 ”Primärlufts-fläkt”		Detta steg aktiveras då steg 4 har varit aktivt i inställd tid (0-999sek) och steg villkor är OK och O2-halt i ugn är större än inställt värde för ”Min O2 för start primärluft” (ca 8%). Steget kan manuellt stoppas med separat stoppknapp.
	Saknas	
6 ”Bränsle mm”		Detta steg aktiveras då steg 5 har varit aktivt i inställd tid (0-999sek) och steg villkor är OK. Steget kan manuellt stoppas med separat stoppknapp.

Namn: Zaker Taqawi och Mohammed Abou Naasa



Steg nummer	Stegvillkor	Kommentar
8 "O2-reglering"		Detta steg aktiveras då steg 7 har varit aktivt i inställd tid (0-999sek) och steg villkor är OK. Steget kan manuellt stoppas med separat stoppknapp.
	Allt i auto	
9 "Eldning i drift"		Detta steg aktiveras då steg 8 har varit aktivt i inställd tid (0-999sek) och alla ingående komponenter för ugn/panna är i driftmod auto.

Larm
Larm "Pannfunktioner ej i auto vid drift" erhålls 30 minuter efter att steg 8 är aktivt men inte steg 9.

Figur 10: Funktionsbeskrivning av eldningsautomatik

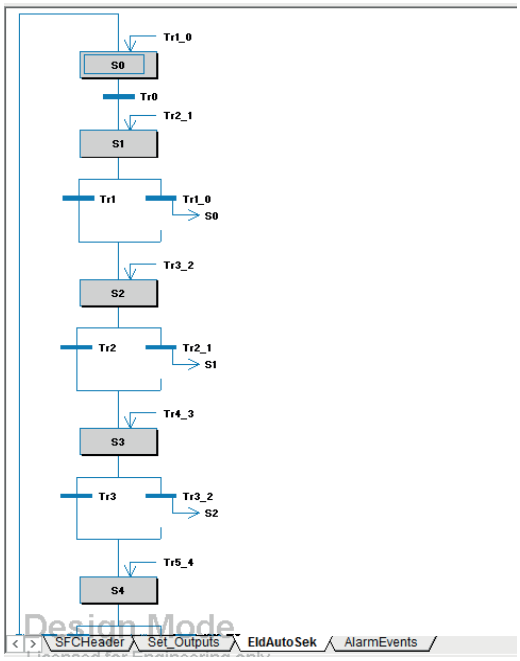
I den beskrivna eldningsautomatiken utgör var och en av de nio stegen en specifik funktion för att säkerställa en effektiv och kontrollerad eldningsprocess. För det första, i steg 1, sätts alla objekt i autoläge för autostyrning. Därefter, i steg 2, aktiveras rökgasfläkten för att påbörja cirkulationen av gaser genom pannan. I steg 3, aktiveras tertiärluftsfläkten, som tillsätter luft i slutet av förbränningszonen för att allt skall förbrännas klart. I steg 4, aktiveras sekundärluftsfläktarna 1 och 2, för att säkerställa optimal syrehalt och tryck. I steg 5, sätts primärluftsfläkten igång för att öka luftflödet ytterligare.

I steg 6 aktiveras bränsleinmatning, roster (se avsnitt 3.3.1), luftsotning (se avsnitt 3.3.2) och stokerskruvar (se avsnitt 3.3.3.2), som är kritiska för att försörja ugnen med pellets. Därefter, i steg 7, aktiveras masterregulatorn för att reglera roster, stokerskruvar, ventiler och andra relevanta komponenter. I steg 8, aktiveras syreshaltsregulatorn, som är en central del av förbränningsprocessen och säkerställer en optimal syrehalt. Slutligen, i steg 9, indikeras att eldningen är igång och processen är klar för fortsatt användning.

Eldningsautomatiken implementeras som en sekvens med hjälp av sequencer function chart (SFC) programvara. I figur 10 ovanför, framgår i kommentarsektionen att varje steg är försedd med en manuell stoppknapp, och att efterföljande steg ska stoppas om en knapp trycks. Till exempel, om steg 6 stoppas genom att en knapp trycks, ska även steg 6, 7, 8 och 9 avbrytas. I figuren 11 nedan visas en SFC-kodning av eldningsautomatiken.

I Tr0 ligger stegvillkoret för att komma in steg 1. Tr1 finns villkoret för att komma in i steg 2 och Tr1_0 finns villkor för att stoppa steg 2. Om knappen är tryckt för steg 2 eller om

driftvred är från, hoppar sekvensen i steg 1 tills villkoret är uppfyllt eller om knappen har återställts via “Reset” knapp. Detta gäller för alla steg 1-9 enligt funktionsbeskrivning.

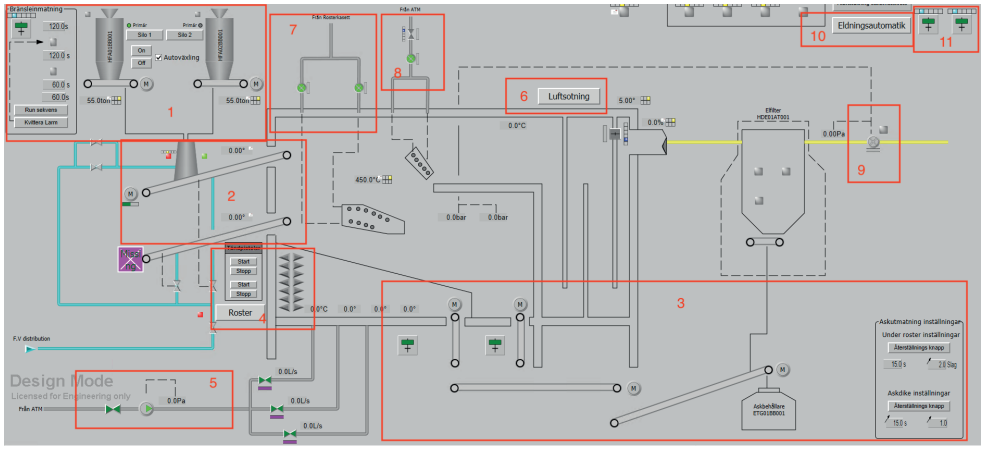


Figur 11: SFC-kodning av eldningsautomatik

3.3 Framställning av processbilder

Efter färdigställandet av programmeringen var det lämpligt att producera flera processbilder genom användning av Process Graphics, vilket nämns i avsnitt 2.3. P&I-schemat utnyttjades som en grundläggande referens vid konstruktionen av dessa processbilder, vilket underlättade skapandet av dessa bilder. Samtliga objekt som styrdes hämtades på bild. Två bilder av dem kan ses i figurerna nedan. Landskrona Energi AB blev ombedd om tillstånd att inkludera dessa två figurer i rapporten. I ABBs Process Graphic manual finns ett steg-för-steg procedur att bygga en processbild. [3]

Namn: Zaker Taqawi och Mohammed Abou Naasa

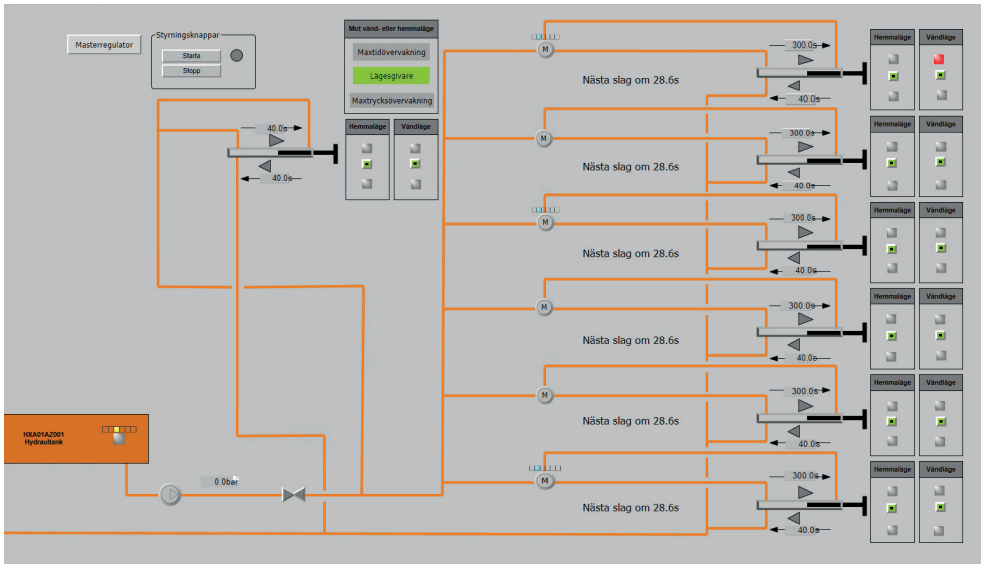


Figur 31: En översiktsprocessbild för pelletspannan i Örja

1. Faceplate på bränsleinmatningsekvens och bränsleskruvar med logik för autoväxling.
2. Stokerskruvar som matar ugnen med pellets.
3. Faceplate på askutmatningssekvens och askskruvar.
4. Rosterknapp och pilarna indikerar åt vilket håll rostern kör.
5. Primärluftsfläkt.
6. Luftsotning är en knapp som för vidare användaren till en annan processbild där 8 tank med alla 28 ventiler finns.
7. Sekundärluftsfläktar 1 och 2.
8. Tertiärluftsfläkt.
9. Rökgasfläkt
10. Eldningautomatik är en knapp som för vidare användaren till en annan processbild.
11. Faceplate för Eldningsautomatikssekvens och luftsotningssekvens.

Översiktsbilden av pelletspannan i Örja visar flera knappar, såsom Roster, Luftsotning och Eldningsautomatik. När någon av dessa knappar trycks på, överförs användaren till en annan sida som är specifik för den valda funktionen. Till exempel, om Roster-knappen väljs, leder det till en annan sida som ser ut, som i Fig.32.

Namn: Zaker Taqawi och Mohammed Abou Naasa



Figur 32: Processbild för 6 roster och en askskrapa

3.4 Källkritik

Referenserna [1][2][3][4][5] utgörs av ABB:s egna manualer som är tillgängliga online. Vi har även tillgripit ABB-manualer som anses vara pålitliga eftersom ABB är tillverkaren av de produkter som vi har använt. Dessa manualer kan endast redigeras av ABB, och de nedladdade PDF-filerna kan inte modifieras, vilket ökar trovärdigheten.

Referens [6], som avser KKS-beteckningssystemet, är en PDF-fil som tillhandahållits av Conny Franzon, chef på Deterministic Control AB. Erfaren personal på Deterministic har i många år utgått från denna fil för att arbeta med KKS, vilket anses vara en pålitlig källa.

4 Analys

Detta kapitlet behandlar diskussion och analys av ingående delar för projektet under programmeringens gång, samt de problem som uppstod och deras lösningar.

4.1 Utmaningar med arbetet

4.1.1 Förståelse för funktionsbeskrivningen

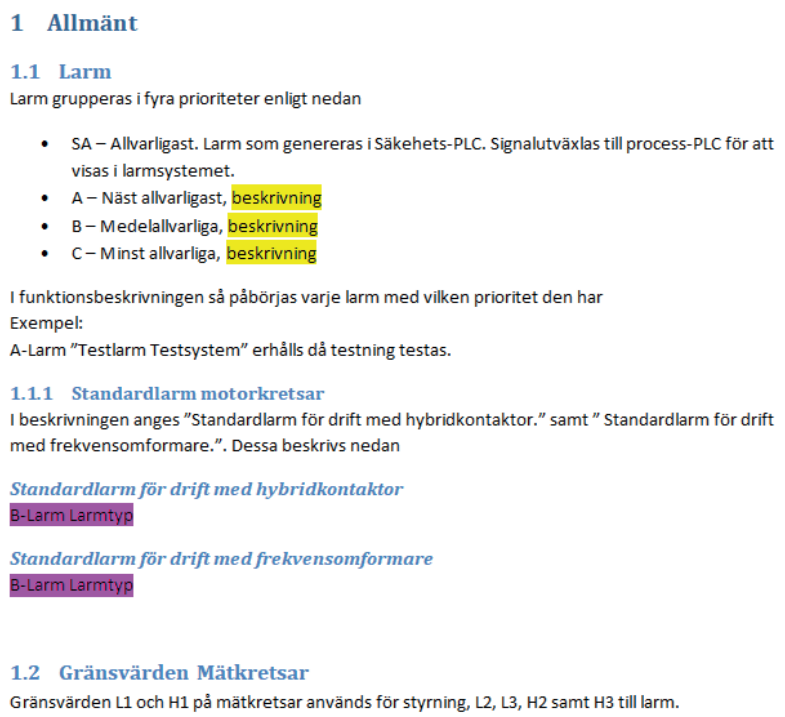
Det första som sker innan programmeringen av pannan startas är instudering av funktionsbeskrivning, om hur pannan är tänkt att fungera och hur en programmerare kan använda beskrivningen som en mall för att se till att funktionerna fungerar. Det allra vanligaste sättet är att gå igenom innehållsförteckningen, för att se vilka KKS-namn varje system har.

Det underlättar för programmeraren att först kunna strukturera upp sin applikation genom att döpa de olika ‘’areor’’ med de KKS-namn som kommer ifrån funktionsbeskrivningen och sedan lägga till enheter som hör ihop med de relevanta systemen. T.ex har vi en systembeteckning för framledning av en panna med beteckningen ‘’NDA’’ och vi har en temperaturgivare som heter 10CT001 som mäter temperaturen vid framledningen, då kommer den att hamna i ‘’Arean’’ NDA med namnet ‘’NDA10CT001’’.

Något annat som var intressant med funktionsbeskrivningen var att för de flesta givare som mätte någonting, tydliga standardnivåer för vilka nivåer som skulle larma operatören (eller varna) och vilka nivåer som ska användas för styrning (t.ex styrning av regulatorer, ventiler, pumpar mm). Dessutom så fanns också standard för hur allvarliga larmen får vara i en skala från 1 till 1000, där 1 är minst allvarligt och 1000 som mest allvarligt.

Notis: Den första versionen av funktionsbeskrivningen har ingen tydlig beskrivning för vilka allvarlighetsgrad som ska gälla för olika larmnivåer, vilket fick oss att improvisera lämpliga allvarlighetsnivåer. Se bild nedan.

Namn: Zaker Taqawi och Mohammed Abou Naasa



1 Allmänt

1.1 Larm

Larm grupperas i fyra prioriteter enligt nedan

- SA – Allvarligast. Larm som genereras i Säkehets-PLC. Signalutväxlas till process-PLC för att visas i larmsystemet.
- A – Näst allvarligast, beskrivning
- B – Medelallvarliga, beskrivning
- C – Minst allvarliga, beskrivning

I funktionsbeskrivningen så påbörjas varje larm med vilken prioritet den har
Exempel:
A-Larm "Testlarm Testsystem" erhålls då testning testas.

1.1.1 Standardlarm motorkretsar

I beskrivningen anges "Standardlarm för drift med hybridkontaktor." samt " Standardlarm för drift med frekvensomformare.". Dessa beskrivs nedan

Standardlarm för drift med hybridkontaktor

B-Larm Larmtyp

Standardlarm för drift med frekvensomformare

B-Larm Larmtyp

1.2 Gränsvärden Mätkretsar

Gränsvärden L1 och H1 på mätkretsar används för styrning, L2, L3, H2 samt H3 till larm.

Figur 33: Typiskt bild för hur olika larm kan se ut för olika enheter

För följande larm så tilldelades varje nivå enligt allvarlighetsskalan:

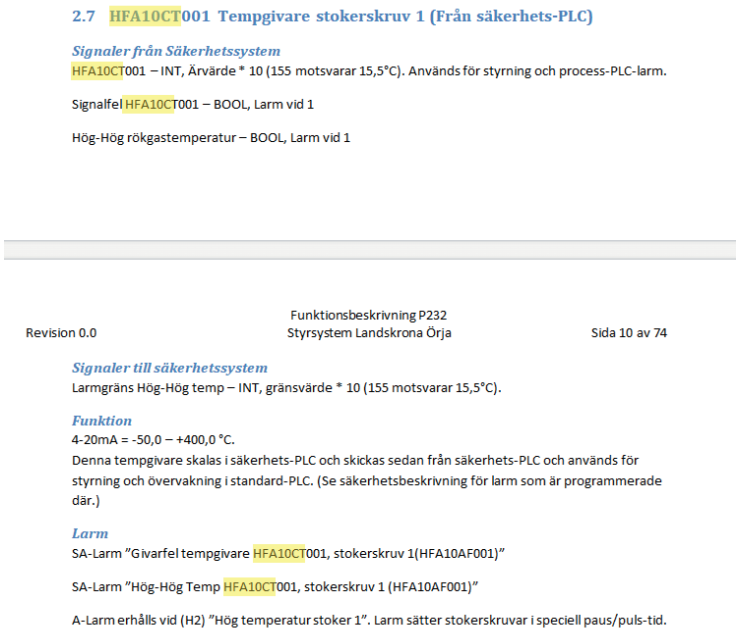
- SA-larm - 1000
- A-larm - 800
- B-larm - 600
- C-larm - 400

Gränsvärden för olika sorters mätkretsar är också värt att prata om, med tanke på hur inkonsekvent funktionsbeskrivningen har varit gällande det. Som texten lyder så ska objekttyper som har L2 (Låg-Låg-nivå), L3 (Låg-Låg-Låg-nivå) eller H2/H3 (hög-hög-nivå etc) indikera larmnivåer för någon specifik parameter (t.ex temperatur,tryck mm) och L1/H1 (Låg/hög-nivå) användas för styrning.

Vid detaljläsning av funktionsbeskrivningen, så fanns många givare och mätare där denna

princip inte tillämpas, speciellt för givare som är avsedda för säkerhets-PLC. Detta är värt att påpeka, eftersom det kan förvirra både operatörer som ska kontrollera mätvärden och för själva programmeraren. Detta förblir en tendens som förekommer alltför ofta.

Ett tydligt exempel som kan nämnas gäller för stokerskruvar i anläggningen. En temperaturgivare som ska mäta temperaturen för skruven ska larma operatören för hög temperatur och sätta special paus och puls tider när skruvarna körs intermittent.



2.7 HFA10CT001 Tempgivare stokerskruv 1 (Från säkerhets-PLC)

Signaler från Säkerhetssystem

HFA10CT001 – INT, Ärvärde * 10 (155 motsvarar 15,5°C). Används för styrning och process-PLC-larm.

Signalfel HFA10CT001 – BOOL, Larm vid 1

Hög-Hög rökgastemperatur – BOOL, Larm vid 1

Signaler till säkerhetssystem

Larmgräns Hög-Hög temp – INT, gränsvärde * 10 (155 motsvarar 15,5°C).

Funktion

4-20mA = -50,0 – +400,0 °C.
Denna tempgivare skalas i säkerhets-PLC och skickas sedan från säkerhets-PLC och används för styrning och övervakning i standard-PLC. (Se säkerhetsbeskrivning för larm som är programmerade där.)

Larm

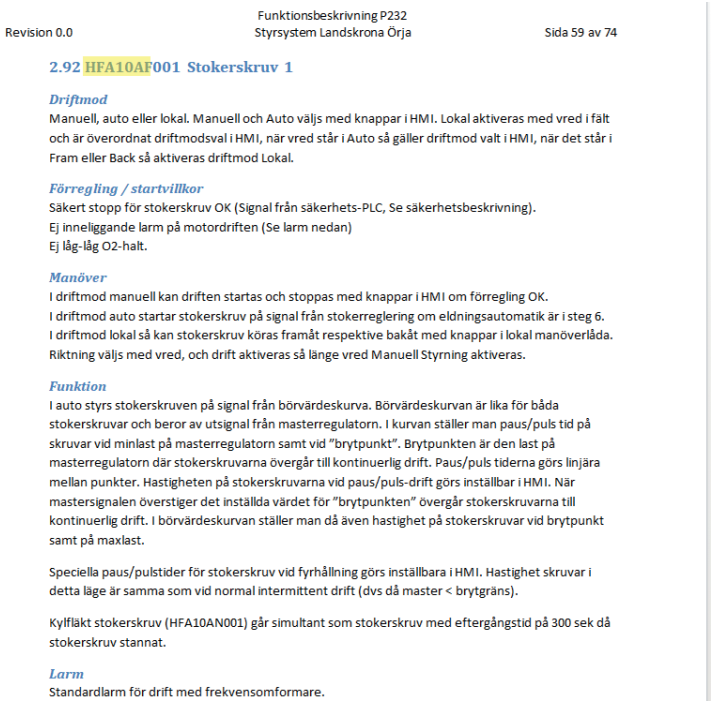
SA-Larm "Givarfel tempgivare HFA10CT001, stokerskruv 1(HFA10AF001)"

SA-Larm "Hög-Hög Temp HFA10CT001, stokerskruv 1 (HFA10AF001)"

A-Larm erhålls vid (H2) "Hög temperatur stoker 1". Larm sätter stokerskruvar i speciell paus/puls-tid.

Figur 34: Ett exempelbild på hur en funktionsbeskrivning kan se ut, bilden visar säkerhetsgivare för stokerskruvar

Namn: Zaker Taqawi och Mohammed Abou Naasa



Funktionsbeskrivning P232
Revision 0.0 Styrsystem Landskrona Örja Sida 59 av 74

2.92 HFA10AF001 Stokerskruv 1

Driftmod
Manuell, auto eller lokal. Manuell och Auto väljs med knappar i HMI. Lokal aktiveras med vred i fält och är överordnat driftmodsval i HMI, när vred står i Auto så gäller driftmod valt i HMI, när det står i Fram eller Back så aktiveras driftmod Lokal.

Förregling / startvillkor
Säkert stopp för stokerskruv OK (Signal från säkerhets-PLC, Se säkerhetsbeskrivning).
Ej inneliggande larm på motordriften (Se larm nedan)
Ej låg-låg O2-halt.

Manöver
I driftmod manuell kan driften startas och stoppas med knappar i HMI om förregling OK.
I driftmod auto startar stokerskruv på signal från stokerreglering om eldningsautomatik är i steg 6.
I driftmod lokal så kan stokerskruv köras framåt respektive bakåt med knappar i lokal manöverlåda.
Riktning väljs med vred, och drift aktiveras så länge vred Manuell Styrning aktiveras.

Funktion
I auto styrs stokerskruven på signal från börvärdeskurva. Börvärdeskurvan är lika för båda stokerskruvar och beror av utsignal från masterregulatorn. I kurvan ställer man paus/puls tid på skruvar vid minlast på masterregulatorn samt vid "brytpunkt". Brytpunkten är den last på masterregulatorn där stokerskruvarna övergår till kontinuerlig drift. Paus/puls tiderna görs linjära mellan punkter. Hastigheten på stokerskruvarna vid paus/puls-drift görs inställbar i HMI. När mastersignalen överstiger det inställda värdet för "brytpunkten" övergår stokerskruvarna till kontinuerlig drift. I börvärdeskurvan ställer man då även hastighet på stokerskruvar vid brytpunkt samt på maxlast.

Speciella paus/pulstider för stokerskruv vid fyrhållning görs inställbara i HMI. Hastighet skruvar i detta läge är samma som vid normal intermittent drift (dvs då master < brytgräns).

Kylfläkt stokerskruv (HFA10AN001) går simultant som stokerskruv med eftergångstid på 300 sek då stokerskruv stannat.

Larm
Standardlarm för drift med frekvensomformare.

Figur 35: Beskrivning av funktionen för stokerskruv

I texten finns det ingen information gällande för att skruven ska sättas i speciella paus/puls tider för larm vid temperaturgivare. Detta är viktigt därför att beskrivningen ovan beskriver ett specialfall när skruvarna får lov att sättas i speciella paus/puls tider (gällande när fyrhållning gäller). Den beskriver inget om specialfallet för temperaturgivaren, som har högst prioritet pga. att den larmar operatören att temperaturen är för hög och kan medföra risk för skruvarna att brinna upp. Dessa problem känner inte en programmerare till och då ska kommunikationen ske ständigt med leverantören, i detta fall AKJ.

4.1.2 Inlärning av arbetssätt

Att kunna påbörja projektet är inte alltid helt klart för hur det ska gå till, speciellt när flera parter är inblandade. I Utbildnings kapitlet (se 3.2 Utbildning) så diskuteras hur workshopen bidrog till att lära ut funktionerna av Control Builder och Function Designer i god omfattning att programmeringen kunde påbörjas på egen hand.

Men projektet innebar mycket mer än att förstå programmeringsmiljöer. Den omfattade bra förståelse av funktionsbeskrivningen och kunna förstå innebörden med KKS-namnen, samt

funktionerna till systemen som är beskrivet. Att kunna hitta typkretsobjekt som passar till det som ska realiseras från funktionsbeskrivningen var inte heller alltid självklart, utan ständig kommunikation med handledaren i Deterministic Control AB var nödvändigt för att komma vidare med projektet.

En annan stor fördel som drogs nytta av var ständig dokumentation och kommunikation för att se till hur mycket av arbetet som faktiskt var avklarat och dokumenterat och hur mycket som fanns kvar i Örja pannans anläggning. Med en (eller flera) Excel-filer så kunde anläggningen mappas i flera delsystem för att få en klar bild av vad som behövdes göras. Detta sparade mycket tid och underlättade arbetet signifikant.

5	GAC						
6							
7	GAC10AA301-MS01	☐	☐	Ställdon nödkylning on/off	Ej i funktionsbeskrivning		
8	GAC10AA301-B01	☐	☐	Lägesindikering Ställdon nödkylning on/off	Ej i funktionsbeskrivning / FB signal		
9	GAC10AA301	☑	☐	Reglerventil nödkylning inlopp on/off			
10	GAC11CF201	☑	☑	Flödevakt tillbakabrand			2023-02-20
11	GAC30CL001	☑	☑	Nivågivare expansionskärl			2023-02-20
12	GAC20AA151	☑	☑	Magnetventil påfyllning exp.kärl - kallvatten			2023-02-20
13	GAC11AA004	☐	☐	Shut-off valve	Ej i funktionsbeskrivning		
14	GAC11AA002	☐	☐	Shut-off valve	Ej i funktionsbeskrivning		
15	GAC10AA601	☐	☐	Shut-off valve Kallvatten	Ej i funktionsbeskrivning		
16	GAC11AA001	☐	☐	Shut-off valve Kallvatten	Ej i funktionsbeskrivning		

Figur 36: Exempel bild på en del av Excelfilen för dokumentation.

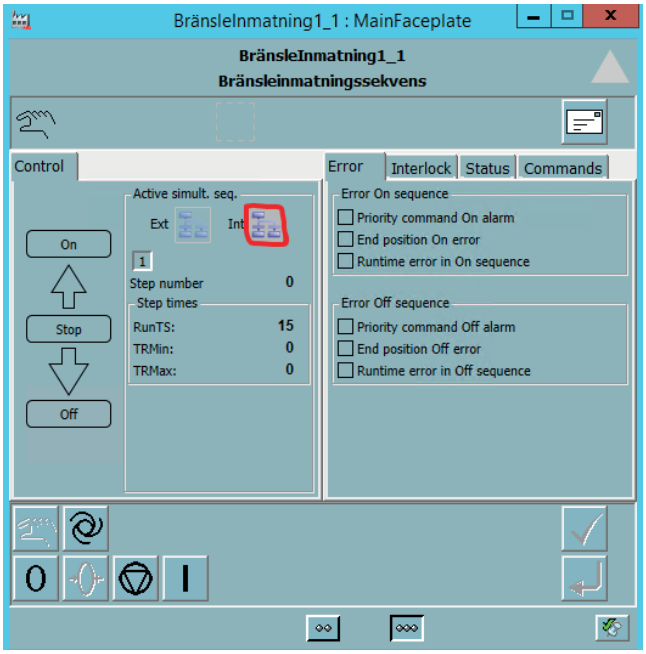
4.2 Val av Programmeringsspråk

Samtliga styrbara objekt som beskrivs i funktionsbeskrivningen programmerades för pelletspannan. Under hela programmeringsprocessen användes tre programmeringsspråk, nämligen funktionsblockdiagram (FBD), strukturerad text (ST) och sekventiellt funktionsdiagram (SFC). FBD användes för att programmera nästan alla delar av pelletspannan, såsom pumpar och fläktar, med flera. ABB:s bibliotek innehöll färdiga typkretsar som var enkla att använda. handledaren Claes Olsson rekommenderade att SFC:en skulle användas för allt som rör sekvensen, medan ST skulle användas för all beräkning. Detta tillämpades på vissa delar av pelletspannan, såsom bränsleinmatning, askutmatning, luftsotning och eldningsautomatik. Detta var dessutom enklare än att programmera varje del med andra programmeringsspråk som ST, FBD, och så vidare.

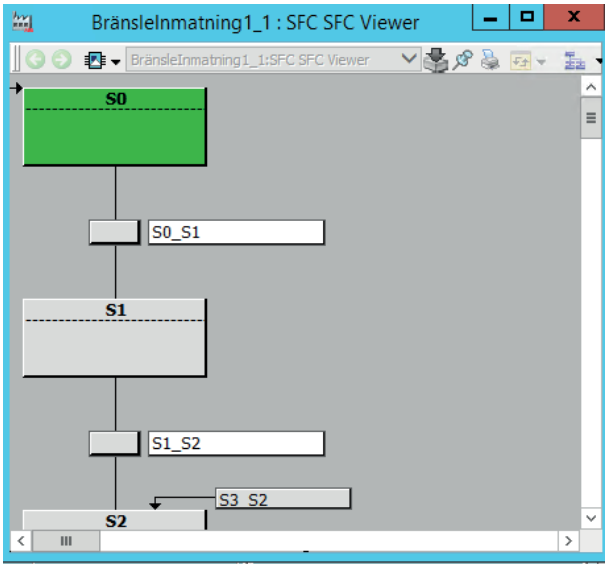
För att rostern skulle kunna pulseras behövde ett funktionsblock användas. Ett sådant block fanns i ABB:s bibliotek och kallades pulsGeneratorn, men det var inte lämpligt för roster-fallet eftersom roster endast skulle pulseras när hydrauliken var ledig samtidigt som

paustiden hade löpt ut. PulsGenerator-objektet hade en fast paus- och pulstid, vilket inte var användbart i detta fall. Därför skapades ett nytt funktionsblock som programmerades med ST. Blocket programmerades så att det endast pulserade när det ovan nämnda villkoret var uppfyllt, och en paustid beräknades och utfördes inuti blocket (se figur 17).

Ett av målen var att göra programmeringen så enkel som möjligt för slutanvändarna, genom att använda enkel logik och skriva kommentarer så att det blir lättare att felsöka om något fel uppstår och dessutom göra det enkelt för operatörerna. En av fördelen med att använda SFC:en är att när operatörerna tar fram faceplate har de även möjlighet att se igenom sekvensen, nämligen vilket steg i sekvens ligger processen i ett system som är i drift. Dessutom har operatörer möjlighet att se stegvillkoret. Om på grund av något fel fastnar sekvensen i något steg kan de ta fram stegvillkoret och se om alla villkoren är uppfyllda.



Figur 37: Faceplate på bränsleinmatningssekvens med SFC viewer (rödmarkerad)



Figur 38: SFC-viewer som operatörer kan se och följa sekvensen utan att behöva gå in i koden

4.3 Programmering i Function Designer

Den ursprungliga planen var att programmera pelletspannan i Control Builder som sedan ändrades till Function Designer enligt slutkundens önskemål. Control Builder är relativt sett enklare, eftersom detta är oberoende av något annat program. Däremot Function Designer sker i Plant Explorer, vilket innebär vid nedladdningen eller kompileringen av något diagram måste Control Builder vara igång, eftersom diagrammet går till softController via Control Builder. Det vill säga Function Designer är beroende av Control Builder. Lite av koden programmerades dock i Control Builder och lades in i ett bibliotek, som exempelvis pulsRoster. Detta för att kunna använda sådana typkretsar till andra projekt i framtiden.

4.3.1 För- och nackdelar med Function Designer

4.3.1.1 Fördelar

- Function Designer ger möjligheten att användaren kan gå in i koden via grafiken genom att högerklicka på ett objekt, exempelvis ett motorobjekt och sedan välja Function. Detta för användaren vidare till det diagram där koden befinner sig. På så sätt kan användaren felsöka i just det diagrammet om något fel har skett eller om användaren vill se vilka diagram och objekt är det här motorobjektet uppkopplad. Den

här funktion saknas i Control Builder, vilket tvingar användaren att gå in i applikationen och börja leta efter KKS:en som det här motorobjektet tillhör.

- En annan fördel Function Designer har är att användaren kan hoppa till det diagram som objektet är kopplad till via kommunikationsvariabler genom att dubbelklicka på kommunikationsvariabler. Exempelvis förreglas ett motorobjekt av en flödesvakt och bägge ligger i var sitt diagram men dock är de kopplade via en kommunationsvariabel. Genom att dubbelklicka på kommunationsvariabeln kan användaren komma åt dessa diagram. Den här funktionen fattas dock i Control Builder. I Control Builder behöver användaren se till vilket/vilka diagram är kommunationsvariabeln kopplad till och sedan manuellt öppna det diagrammet.

4.3.1.2 Nackdelar

- Kan ej skapa egna typkretsar jämfört med Control Buildern
- Mycket buggar eller krasch som gör Function Designer väldigt tungarbetad.
- Function Designer har inte stöd för globala variabler. Istället kan det bara förlita sig på kommunikationsvariabler för att kommunicera mellan diagram. Dessa variabler har en tidsfördröjning vid dataöverföring mellan två diagram som kommunicerar via kommunikationsvariabeln.
- I många fall så kan Function Designer lämna obegripliga felmeddelanden som gör det svårt att förstå vilken del av diagramkoden som den syftar på.
- Då nerladdning till SoftControllern är redo och den befinner sig online, så är det inte möjligt att göra kod ändringar under tiden , detta kan dock göras i Control Buildern.
- För att hitta objekttyper som behövs vid användning för kodning , så krävs det bra förståelse för i vilket bibliotek som objektet befinner sig i och manuellt söka sig till det rätta biblioteket för att hämta in objektet. I Control Buildern så har den en sökningsfunktion som automatiskt hämtar in objektet förutsatt att namnet på objektet är korrekt.
- I både Function Designer och Control Builder finns möjlighet för att skapa flera sidor i ett diagram och koppla objekt som ligger i olika sidor via pagelinks[11]. I Function Designer går det ej att navigera användaren till den sidan som objekt är kopplad till genom att dubbelklicka. Däremot finns den här funktionen i Control Builder,

[11] **Pagelinks:** Det innebär två olika objekt som befinner sig i ett enda diagram men på två olika sidor och de är kopplade till varandra via pagelinks.

dubbelklickar användaren på pagelinks och navigeras sedan användaren till den sidan som objektet är kopplad till.

- Det går inte att döpa om sidonamnet i Function Designer, dock finns det möjlighet att lägga kommentar för varje sida. Däremot i Control Builder finns möjligheten att döpa om sidonamn.

4.3.2 Ovanligt många buggar och krasch

En av de mest användbara funktionerna som Control Builder har, är funktionen att kunna ge tydliga felmeddelanden om ett fel har uppstått. Antingen vid nedladdning av applikationen eller kompilering av kod, så kommer det tydligt felmeddelande med navigeringsmöjlighet till källan av problemet.

Function Designer har begränsade möjligheter gällande detta område. Det största problemet gällande felmeddelanden är hur otydliga meddelanden kan vara. Detta tvingar programmeraren att felsöka logiken i hela diagrammet, trots Function Designers stöd för att kunna navigera programmeraren till felkällan. Detta kan vara problematiskt om diagrammet som bearbetas är för stort, vilket är vilseledande. Desto värre är det om det finns flera felmeddelanden som inte visas samtidigt, utan bara visas upp ett i taget beroende på hur allvarliga felen är. Dessutom kan det finnas fel som Function Designer inte klagar på vid kompileringen, däremot vid nedladdningen eftersom programmet laddas ned till softController via Control Builder, upptäcker Control Builder felen och förhindrar detta i sin tur nedladdningen. Dock finns det en tydlig information var och i vilket diagram felen finns.

Ett annat exempel på felmeddelandet som var vilseledande var "Linked into another structure”. Detta meddelande dyker upp när antingen namn på två olika objekt är samma eller insignalen till ett objekt har råkat kopplas till utsignal av ett annat objekt. Namn krockar brukar ske när ett objekt kopieras och klistras in, och sedan glömmer programmeraren att ändra namn. Sådana felmeddelanden kan vara svåra att hantera för någon som inte har sysslat med Function Designer. Problemet kan lösas genom att hitta objektet och ändra namn på det eller eventuellt granska om kopplingen är rätt. Dessa felmeddelanden kan ibland krascha hela Plant Explorer Workplace.

4.4 Fördröjning av arbetet

Detta avsnitt syftar till att belysa de fördröjningar som förekom under arbetets gång och de konsekvenser de medförde för både individer och företag.

4.4.1 Funktionsbeskrivningen

Att skriva en funktionsbeskrivning för en pelletspanna och själva anläggningen i sig, tar en lång tid att göra. En avstämning krävs för att säkerställa att rätt funktioner ska ingå. Dessutom krävs korrekta KKS-beteckningar för den utrsutning som ska levereras för att lätt identifiera systemen och instrumenten i systemet. Sedan behöver det göras tydligt för programmeraren av funktionsbeskrivningen, vilka säkerhetsinstrument som finns och hur de ska hanteras med. Dessutom är det viktigt att veta vilka larm som behövs för olika system.

Deterministic AB fick en uppskattning i slutet av november 2022 om att funktionsbeskrivningen skulle vara klar för leverans till programmeraren. Då arbetet hade officiellt börjat från och med den 11 januari 2023 var det värt att spendera värdefull tid på att undersöka en annan funktionsbeskrivning för en anläggning i Knivsta och förstå hur det fungerade, under tiden som den aktuella funktionsbeskrivningen inväntades.

Arbetet hade dock fördröjts till den 27 januari 2023, då funktionsbeskrivningen blev levererad. Tiden som gick förlorad under dessa 12 arbetsdagar (med 6-8 arbetstimmar per dag) var betydande.

För att förstå varför detta sker så måste parternas prioriteringar i projektet belysas. Generellt så är programmerarna i princip lägst prioriterade inom projektet därför det är viktigare att säkerställa vad som ska ingå för pannan innan programmeringen av den kan ske. Vilket sorts utrustning som ska levereras, hur den ska levereras, dimensioner på varje utrustning, kostnaden mm. är bara några punkter som måste specificeras. Dessa prioriteringar gäller först och främst, för att säkerställa att Örja-pannan kommer att byggas upp och att utrustningen för pannan bli levererad inom rimlig tid.

Därför är det inte så svårt att förstå varför programmerare hamnar sist i kedjan när det gäller att de ska få informationen de behöver, för att programmera klart pannan. Det finns ingen poäng att leverantören ska spendera sin tid med att först skriva klart en funktionsbeskrivning

för utrustning för en anläggning som ännu inte är levererad.

Det innebär då att som en programmerare, då gäller det att tidsplanera ordentligt och försöka dels uppskatta tidsplanen för arbetet samt driftsättning, men dels också försöka inkludera en god marginal med tid , för att kompensera för möjliga förseningar från andra parter.

4.4.2 I/O Lista och elritningar

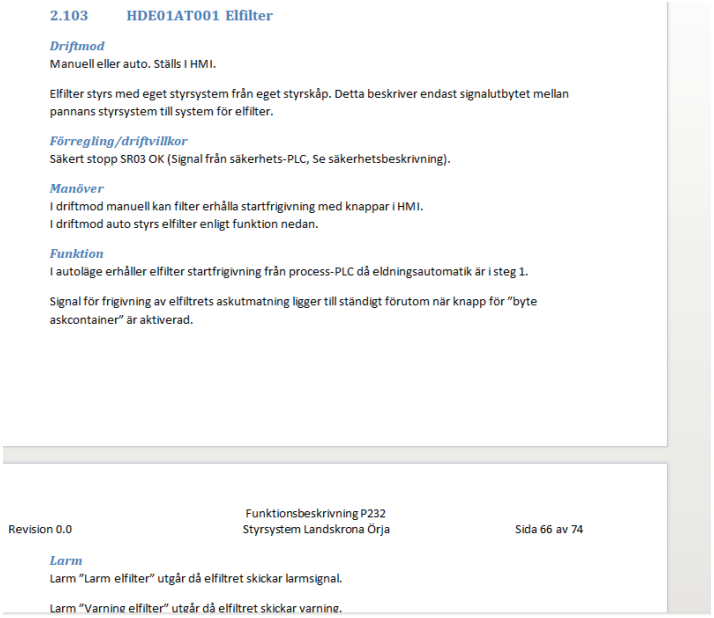
Av allt som kan levereras av en leverantör till ett programmeringsföretag, så är nog en I/O-lista och elritningar av enheter som ska programmeras, de mest lågprioriterade. Att kunna tilldela varje instrument, regulator, pumpar etc en fysisk IO-adress samt att kunna rita ut adresserna på en elritning för lättare förståelse,är väldigt tidsödande. Det är inte så konstigt att tänka sig då en leverantör har också mycket annat som ska levereras också (t.ex ett elskåp till kunden).

Men faktumet att det inte är så ovanligt med den långa väntetiden innan I/O-listan och elritningar blir levererade, vilket kan fördröja en programmerares arbete. Förutom att en I/O lista innehåller vilka specifika I/O-kort som ska användas för projektet och dess adresseringar, så innehåller generellt en I/O-lista tydligt den information som kan saknas i en funktionsbeskrivning. Ta Elfiltret som ett exempel. (Se figur nedan).

1	Index	Typ	System	Skåp	CPU	Module-buss	Slot	Kanal	Adress	KKS	Detalj	KKS-Beskrivning	Område / funktion	Kommentar	Retä
107	106	DI	UAD141	CJF01	0	11	9	10	0.11.9.ch10	HDE01AT001	Nödstopp utl	Elfilter			
108	107	DI	UAD141	CJF01	0	11	9	11	0.11.9.ch11	HDE01AT001	A-Larm	Elfilter			
109	108	DI	UAD141	CJF01	0	11	9	12	0.11.9.ch12	HDE01AT001	B-Varn	Elfilter			
110	109	DI	UAD141	CJF01	0	11	9	13	0.11.9.ch13	HDE01AT001	Filter Standby	Elfilter			
111	110	DI	UAD141	CJF01	0	11	9	14	0.11.9.ch14	HDE01AT001	Filterdrift	Elfilter			
112	111	DI	UAD141	CJF01	0	11	9	15	0.11.9.ch15	HDE01AT001	Askborttagn drift	Elfilter			
113	112	DI	UAD141	CJF01	0	11	9	16	0.11.9.ch16	HDE01AT001	Start external Transport	Elfilter			
149	148	DO	UAD141	CJF01	0	11	103	12	0.11.103.ch12	HDE01AT001	Ext Frislagsn	Elfilter			x
150	149	DO	UAD141	CJF01	0	11	103	13	0.11.103.ch13	HDE01AT001	Ext asktransp ok	Elfilter			x

Figur 39: Exempelbild för hur en I/O lista kan se ut

Namn: Zaker Taqawi och Mohammed Abou Naasa



2.103 HDE01AT001 Elfilter

Driftmod
Manuell eller auto. Ställs I HMI.

Elfilter styrs med eget styrsystem från eget styrskåp. Detta beskriver endast signalutbytet mellan pannans styrsystem till system för elfilter.

Förregling/driftvillkor
Säkert stopp SR03 OK (Signal från säkerhets-PLC, Se säkerhetsbeskrivning).

Manöver
I driftmod manuell kan filter erhålla startfrigivning med knappar i HMI.
I driftmod auto styrs elfilter enligt funktion nedan.

Funktion
I autoläge erhåller elfilter startfrigivning från process-PLC då eldningsautomatik är i steg 1.

Signal för frigivning av elfiltrets askutmatning ligger till ständigt förutom när knapp för "byte askcontainer" är aktiverad.

Funktionsbeskrivning P232
Revision 0.0 Styrsystem Landskrona Örja Sida 66 av 74

Larm
Larm "Larm elfilter" utgår då elfiltret skickar larmsignal.

Larm "Varning elfilter" utgår då elfiltret skickar varning.

Figur 40: Bild på Elfiltrets funktionsbeskrivning

I/O-listan för elfiltret innehåller mer information för vad funktionen av filtret ska innehålla, än vad som beskrivs i funktionsbeskrivningen. Detta kan orsaka oro eller avbrott i arbetet när programmeraren reflekterar över koden och inser att systemet inte är klart på grund av saknad information.

4.5 En Control Builder i två virtuella maskiner

En av de bästa egenskaperna hos Control Buildern i ABB 800xA är förmågan att kunna ha flera olika maskiner, fysiska eller virtuella, arbeta i samma Application. Den möjligheten underlättar för organisering av arbetet och minskar tiden som behövs ägnas åt applikationen. Egenskapen att kunna arbeta med varandra har att göra med hur ABB har byggt upp anslutningsstrukturen, att det finns en maskin som kör Connectivity Server och resterande maskiner kör på Aspect Server.[12]

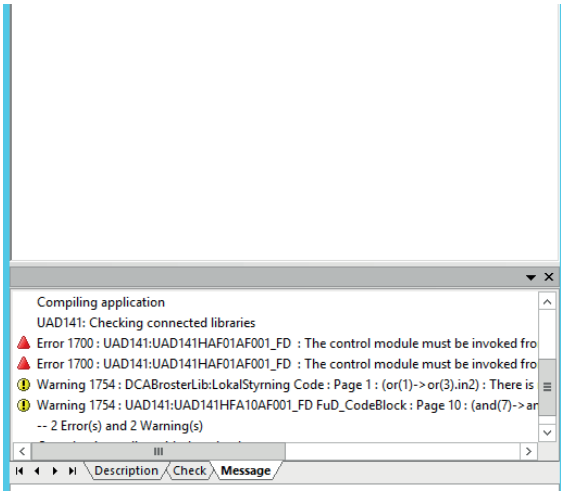
[12] **Connectivity Servern:** En enda uppkoppling till en miljö, där en användare ansvarar för uppsättning av SoftControllern och ser till att den går att använda för nedladdning.
Aspect Server: Där flera uppkopplingar kan ske av flera användare till samma miljö som Connectivity Serverns uppkoppling, för att koda eller debugga.

Men det finns olika scenarios som kan ställa till det för arbetare inom applikationen. Denna rubrik ska belysa dessa scenarios.

4.5.1 Nedladdning till SoftController

Vid arbete med flera personer, så kommer ett läge för programmeraren att debugga och testa en del av applikationskoden. SoftController gör en kontroll av hela applikationen och skannar för errors. Om SoftController hittar error från något diagram i applikationen, så stannar nedladdningen tills de är åtgärdade. Problemet är att det kan vara oavsiktliga fel som någon har orsakat och som måste hittas, begripas varför felet har uppstått och åtgärda det innan programmet laddas ned igen.

Om det inte är fallet, så kan det finnas fel i koden pga. att någon arbetare sitter och kodar i den stunden. Då har de ingen anledning att kolla igenom om koden är fel då en annan medarbetare sitter i sittande stund och arbetar igenom. Antingen måste diagrammet avallokeras, vilket gör att diagrammet tappar kompileringsfunktionen[13], eller så får felen åtgärdas snabbt utan att störa varandras arbetsprocess.



Figur 41: Exempelbild på hur felmeddelanden ser ut vid nedladdning..

[13] **Kompilering** - en funktion som kontrollerar om objekt koppling eller kod är rätt skrivet respektive rätt kopplad.

4.5.2 I/O Allokering

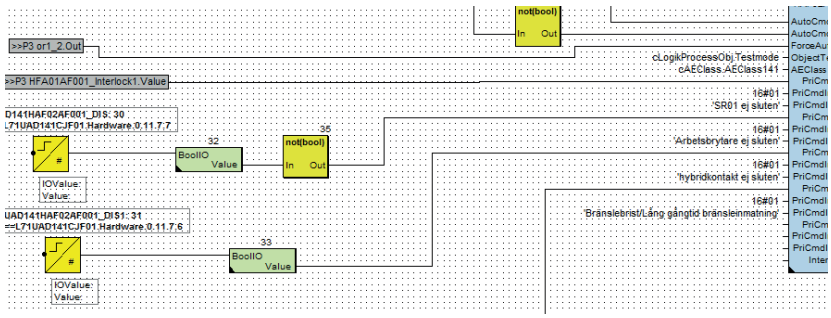
Inom ABBs 800xA så brukar en I/O allokering ske via Control Buildern. Den är byggd på sådant sätt att det ska vara enkelt för användaren att kunna strukturera upp I/O adresser för följande anledningar:

- Att kunna se vilka I/O-kretskort som används och vid ett enkelt knapptryck kunna få upp en vy av alla I/O-allokerade adresser för just det specifikt valda I/O kortet.
- Förhindrar att I/O-allokeringar stoppar användarens arbetsprocess, vare sig de kodar eller debuggar diagram.
- Utgöra möjlighet att ladda in filer med mappade I/O-adresser och information som exempelvis beskrivning av I/O-enheten, signalparametrar, signalgränser mm.

Channel	Name	Type	Variable	I/O Description
IX0.11.7.1	Input 1	BoolIO	UAD141.UAD141HAF01AF001_FD.UAD141HAF01AF001_DIS1	UAD141HAF01AF001_DIS1:
IX0.11.7.2	Input 2	BoolIO	UAD141.UAD141HAF01AF001_FD.UAD141HAF01AF001_DIS	UAD141HAF01AF001_DIS:
IX0.11.7.3	Input 3	BoolIO	UAD141.UAD141HAF01AF001_FD.HAF01AF001_LokalOrder	HAF01AF001_LokalOrder:
IX0.11.7.4	Input 4	BoolIO	UAD141.UAD141HAF01AF001_FD.HAF01AF001_FjärrLäge	HAF01AF001_FjärrLäge:
IX0.11.7.5	Input 5	BoolIO	UAD141.UAD141HAF01AF001_FD.HAF01AF001_BackKörning	HAF01AF001_BackKörning:
IX0.11.7.6	Input 6	BoolIO	UAD141.UAD141HAF01AF001_FD.UAD141HAF02AF001_DIS1	UAD141HAF02AF001_DIS1:
IX0.11.7.7	Input 7	BoolIO	UAD141.UAD141HAF01AF001_FD.UAD141HAF02AF001_DIS	UAD141HAF02AF001_DIS:
IX0.11.7.8	Input 8	BoolIO	UAD141.UAD141HAF01AF001_FD.HAF02AF001_LokalOrder	HAF02AF001_LokalOrder:
IX0.11.7.9	Input 9	BoolIO	UAD141.UAD141HAF01AF001_FD.HAF02AF001_FjärrLäge	HAF02AF001_FjärrLäge:
IX0.11.7.10	Input 10	BoolIO	UAD141.UAD141HAF01AF001_FD.HAF02AF001_BackKörning	HAF02AF001_BackKörning:
IX0.11.7.11	Input 11	BoolIO		
IX0.11.7.12	Input 12	BoolIO		
IX0.11.7.13	Input 13	BoolIO		
IX0.11.7.14	Input 14	BoolIO		

Figur 42: Bild på hur en I/O allokering ser ut i Control buildern.

Function Designer har inte dessa fördelar. När en I/O tilldelas i Function Designers, så görs det via typobjekt som kallas för CBM-signaler. En CBM-signal tilldelas först ett tänkt I/O kretskort och sedan den kanal på kretskortet som signalen ska befinna sig i, på så vis så får CBM-signalen en I/O-allokering. En allokerad CBM signal kopplas sedan till de objekt som de är tänkt för.



Figur 43: Bild på Function Designers CBM I/O allokering.

Med erfarenhet vet en programmerare i förväg vilka sorters objekt som har en motsvarande rätt CBM-signalkoppling. Då går det i förväg att veta och lägga till de CBM-objekten i diagrammen som de ska tillhöra, även om det inte finns en färdig I/O-lista. Flera saker kan ske vid nedladdning till SoftControllern. Vid bästa fall kastas CBM-signaler om på fel sida i hela diagrammet. I värsta fall så förhindras nedladdning till softControllern pga. att CBM-objekten har kastats om utanför fönsterrammen, oåtkomligt. I ett sådant fall behöver allt kod på en av sidorna i diagrammet raderas bort eller i värsta fall behöva radera hela diagrammet och börja om på nytt.

5 Resultat

I det här kapitlet presenteras resultat av tester som har utförts om pelletspannan har programmerats rätt samt om processbilder är rätt framställda och om pelletspanna kan styras via processbilder.

5.1 IFAT- och FAT-tester

Det första IFAT (Internal Factory Acceptance Test) gjordes med Martin Olsson från AKJ den 17/4/2023 som var ansvarig för material som P&I-schema och så vidare. Mycket av anmärkningarna från Martin handlade om grafik- och textförändringar som behövde göras för att inte förvirra operatörerna. Men det fanns flera allvarligare problem som stöttes på under testet och det var följande som behövdes åtgärdas och testas om igen:

- Vid lokalstyrningen så ska lokalorderknappen hela tiden vara nedtryckt för drifts körning (applikationen tog hand om knappen som om en enda puls skickades in)
- Då ett övergångsvillkor i eldningsautomatiken slutar att gälla så ska sekvensen gå tillbaka till det korrekta steget tills övergångsvillkoret börjar gälla igen (sekvensen i applikationen tog inte hand om detta fall)
- Att rostern fortfarande inte ska ha sitt automatik driftläge när driften övergår till manuellt läge
- Se till att rökgasfläkten kunde mäta dess insignal korrekt och larma om hastigheten blev för hög

Dessa punkter var allvarliga av olika anledningar. Lokalstyrningsknappen var programmerad på sådant sätt att ett minnesobjekt (SR-vippa) kunde komma ihåg att en knapp var nedtryckt och då kunde logiken köra motorobjektet framåt eller bakåt utan att knappen behövde hållas nedtryckt. Dessutom så stoppades motorobjektet bara när driftvreden vreds från eller till ett annat läge, medan tanken var att motorn endast skulle stoppas så länge knappen ej är nedtryckt. Detta kan lätt förvirra eller skrämma driftteknikern på plats.

Att övergångsvillkoren inte avbryter nuvarande eldningsautomatik steg är också allvarligt. Ta rökgasfläkten som ett exempel. Dess uppgift är att lufta ut all förbränningsluft som finns i ugnen för att det inte ska finnas högtryck eller hög temperatur. Om rökgasfläkten hade slutat

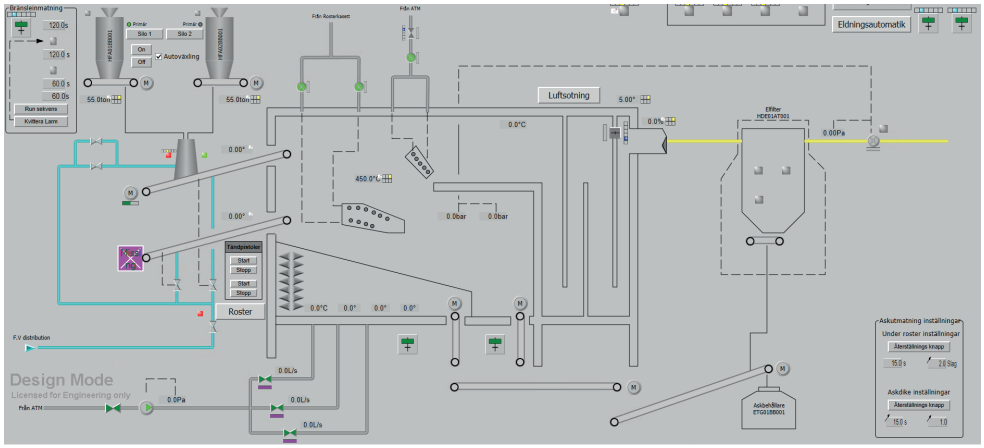
fungera och eldingautomatiken hade inte brutits utan bara fortsätta elda på, då hade ugnen garanterat kunna explodera och skada personal eller utrustning.

Funktionen att kunna sätta objekten i manuellt läge är en viktigt princip om specifika uppgifter ska utföras, som till exempel att stoppa utrustningen i verkligheten för att felsöka eller rengöra. Under FAT-testet så visade det sig att rostern inte hade den någon manuella styrning utan den kunde fortsätta som om den fortfarande var i automatikläge. Detta kan lätt vara skadligt för en drifttekniker som sitter ute vid rostern för att bearbeta en specifik roster och helt plötsligt väcks den till liv.

Den 21/4/2023 så påbörjades det andra FAT-testet. Testet var ännu mer lyckat och hade i princip inga allvarliga problem. Mycket av testet utgick på att redovisa de föregående anmärkningarna och se till att de blev rättade. Huvudsakligen handlade testet om hur det skulle gå till med informationsvisning på grafiken och vilken sorts information var relevant för varje bild.

Ta ett exempel på en säkerhetsklassad signal som kommer ifrån två givare, till exempel pannans temperatur givare HAJ10CT001 och HAJ10CT002. Bägge givarna är positionerade precis bredvid varandra, om en av dem går sönder, då ett mätvärde fortfarande behöver visas. För PLC-logik så finns det en signal som har samma KKS-beteckning men med en ändelse som kallas för ZQ01, dvs HAJ10CT001ZQ01. Denna signal indikerar att det är det gällande värdet som används för att styra PLC-processer eller för att larma operatören och säkerhets plc. Signalen visar det högsta värdet mellan HAJ10CT001 och -002 och indikerar även om differensvärdet mellan givarna överstiger larmgränsen för hur mycket det får skilja sig (två givare som sitter på samma ställe ska ju visa samma värde, annars innebär det att något är fel). För bilden säkerhetssystem så är det ju logiskt att kunna visa både givarevärden och det aktuella styrningsvärdet i en och samma bild, men på andra bilder som översiktsbilden eller fjärrvärme bilden så räcker det med att visa det gällande värdet istället.

Namn: Zaker Taqawi och Mohammed Abou Naasa



Figur 44: Översiktsbild över pelletspannan (En gammal version)

6 Slutsats

I detta kapitel presenteras slutsatsen för hela examensarbetet där problemformuleringar (se avsnitt 1.4) också besvaras. Dessutom beskrivs en reflektion över etiska aspekter. Här framförs även en diskussion och förslag på vidarestudie, nämligen några frågeställningar som kan undersökas i framtiden.

Huvudmålet som sattes upp i början av arbetet har uppnåtts i stora drag, det vill säga PLC-programmering av en pelletspanna till det nya fjärrvärmeverket samt även framställning av processbilder. Detta uppnåddes genom användning av material som levererades av företaget, AKJ. Material som funktionsbeskrivning, I/O-lista och P&I-schema. PLC-programmeringen skedde mot en softController som är en del av ABBs 800xA-system. Denna möjliggör simulering som om det bearbetade programmet direkt i en PLC. Under arbetets gång redovisades varje del till chefen i Deterministic Control AB för att stämma av funktionen. Därefter utfördes ett FAT (Factory Acceptance Test) tillsammans med Martin Olsson (se avsnitt 5). FAT-testet blev godkänt, dock var det några detaljer som behövde åtgärdas.

Under hela programmeringen användes tre programmeringsspråk, nämligen FBD, ST och SFC. Detta besvarar frågeställningen om programmeringsspråk som tillämpas för delar i pelletspannan. Ett av målen var att göra programmeringen så enkel som möjligt med mycket kommentarer för att kunna felsöka sen vid eventuellt fel. Därför användes SFC för sekvenser så att slutanvändarna har möjlighet att följa sekvensen utan att behöva gå in i koden. FBD användes mest av allt, eftersom det fanns färdiga objekt i ABBs bibliotek. Dessutom tillämpades ST för att göra vissa beräkningar, som är svårt att göra med FBD eller andra programmeringsspråk enligt IEC standarden 61131-3 (se avsnitt 4.2).

När det gäller frågeställningen, vilka utmaningar och problem uppstod under projektet, så var de följande: För det första blev material till detta projekt fördröjd av AKJ vilket i sig besvarar en annan frågeställning, nämligen erfarenheter av ett verkligt arbetsprojekt med flera parter involverade. Risken för fördröjningar är given och då ska samtliga parter ha en ständig kommunikation och samarbeta för att uppnå det gemensamma målet (se avsnitt 4.4). För det andra börja förstå funktionsbeskrivningen, I/O listan och P&I-schemat. Dessutom veta hur de hänger ihop samt programmera pelletspannan baserad på de materialen (se avsnitt 4.1.1). För

det tredje gick det att arbeta två personer i en enda Control Builder men det var tämligen utmanande, vilket besvarar en annan frågeställning om det var möjligt att två personer eller fler arbetar med en enda Control Builder. Anledningen är att det stötte till problem vid nedladdningen till softController. För att lösa detta, krävdes konstant kommunikation mellan programmerare innan nedladdningen så att det inte störde motsvarande programmerarens debuggingsprocess (se avsnitt 4.5). Sist men inte minst hur och var ska projektet påbörjas samt hur projektet ska planeras så att det blir färdigt i tid (se avsnitt 4.1.2).

En problemformulering angående programmering i Function Designer som också har analyserats i avsnitt 4.3. Function Designer var väldigt utmanande med tanke på att programmet har begränsade funktioner gentemot Control Builders funktioner. Trots de här brister som Function Designer har, är det viktigt i slutet av dagen att kunden känner sig bekväm med programmet och nöjd.

6.1 Reflektion över etiska aspekter

Ett lyckat examensarbete med en fungerande panna och styrning via processbilder innebär att företaget Landskrona Energi AB, som är slutkunden, kan försörja Landskrona stadsdelen Örja med fjärrvärme. Detta kan de nu även utnyttjas som reserv när behovet kommer att vara stort.

6.2 Vidarestudie

På grund av begränsad tid och omfattande projekt kunde gruppen inte utforska ett antal problemformuleringar som var relevanta för detta examensarbete. Däremot kan dessa problemformuleringar utredas i framtiden.

Fjärrvärmeverket i Örja byggs främst för att stödja Landskrona Energi ABs kraftvärmeverk, Energiknuten. Pelletspannan i Örja är mindre än den i Energiknuten, vilket innebär att vid hög eller maximal belastning är pannans pump effektiv och kan pumpa ut fjärrvärme med samma tryck som Energiknutens starka pumpar. Vid låg belastning är dock pumpen i Örja mindre effektiv än den i Energiknuten, så pass mindre effektiv att den nästan kan uteslutas. Det är därför viktigt att undersöka hur pumpen i Örja kan regleras på sådant sätt så att den inte förblir helt utesluten.

Namn: Zaker Taqawi och Mohammed Abou Naasa

I ett fjärrvärmenät används differenstrycksreglering för att säkerställa att rätt mängd värmeenergi levereras till byggnader och hushåll som är anslutna till nätet. Genom att mäta trycket på två punkter i nätet, till exempel i ett fördelningsskåp eller vid en värmecentral, kan trycket regleras så att det hålls inom en viss gräns. Detta säkerställer att det finns tillräckligt med tryck för att leverera värme till alla anslutna enheter, samtidigt som det undviks högt tryck som kan orsaka skador på systemet.

Genom att reglera trycket i fjärrvärmenätet kan driften optimeras av systemet, vilket säkerställer att det fungerar på ett energieffektivt sätt och att onödiga värmeläckor undviks . Därför är det viktigt att undersöka hur differenstryckreglering ska ske i ett fjärrvärmenät.

7 Terminologi

800xA	Överordnad programmeringsmiljö framtagen av ABB
HMI	Human Machine Interface
IEC	International Electrotechnical Commission -Standardiseringsorgan som tar fram och publicerar standarder för el, elektronik och relaterade teknologier.
PLC	Programmable Logic Controller
SFC	Sequential Function Chart
ST	Structured Text
FBD	Function Block Diagram
FAT	Factory Acceptance Test - Testning av system i en anläggning, syfte att verifiera om funktionerna bemöter leverantörens förväntningar och kundens
IFAT	Internal Factory Acceptance Test - ett FAT test men internt utfört utan kunden.

8 Källförteckning

[1] ABB. 3BSE041880-600 A. 2016. Tillgänglig: https://library.e.abb.com/public/5e6a75b1c25d438db9c7def5ad8cad4a/3BSE041880-600_A_en_System_800xA_Control_6.0_AC_800M_Getting_Started.pdf Hämtad: 16-01-2023

[2] ABB. *Control Builder*. Tillgänglig: https://new.abb.com/control-systems/system-800xa/800xa-dcs/hardware-controllers-io/control-builder-engineering-software Hämtad: 16-01-2023

[3] ABB. 3BSE049230-600 B. 2016. Tillgänglig: https://library.e.abb.com/public/1bfa63553a6b4640a5a521685cf3036b/3BSE049230-600_B_en_System_800xA_Engineering_6.0_Process_Graphics.pdf Hämtad: 23-01-2023

[4] ABB. 3BDS100968-600 C. 2016. Tillgänglig: https://library.e.abb.com/public/1781629a5cdf4770b8ad427c663aed29/3BDS100968-600_C_en_System_800xA_Engineering_6.0_Engineering_Studio_Function_Designer_Getting_Started.pdf Hämtad: 16-01-2023

[5] ABB. 3BSE038018-510 D. 2011. Tillgänglig: https://library.e.abb.com/public/f087119820dbee89c1257b1a005be203/3BSE038018-510_D_en_System_800xA_5.1_System_Guide_Functional_Description.pdf Hämtad: 18-03-2023

[6] Conny Franzon chef i Deterministic Control AB. Fil: *KKS Beteckningssystem*. Hämtad: 27-01-2023

Namn: Zaker Taqawi och Mohammed Abou Naasa

9 Appendix

https://docs.google.com/spreadsheets/d/1NzfxV2c5MEAoVK20xyfMHK66AKcCEgmdC4s7Zv2aYDg/edit?usp=sharing

https://docs.google.com/spreadsheets/d/1qA3hGUge4ZZrFkdjAvbE5XWvwRYwdZctJZTrvjROqvA/edit#gid=0

Printed by Tryckeriet i E-huset, Lund 2023

LUND
UNIVERSITY

Series of Bachelor´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2023-923
http://www.eit.lth.se