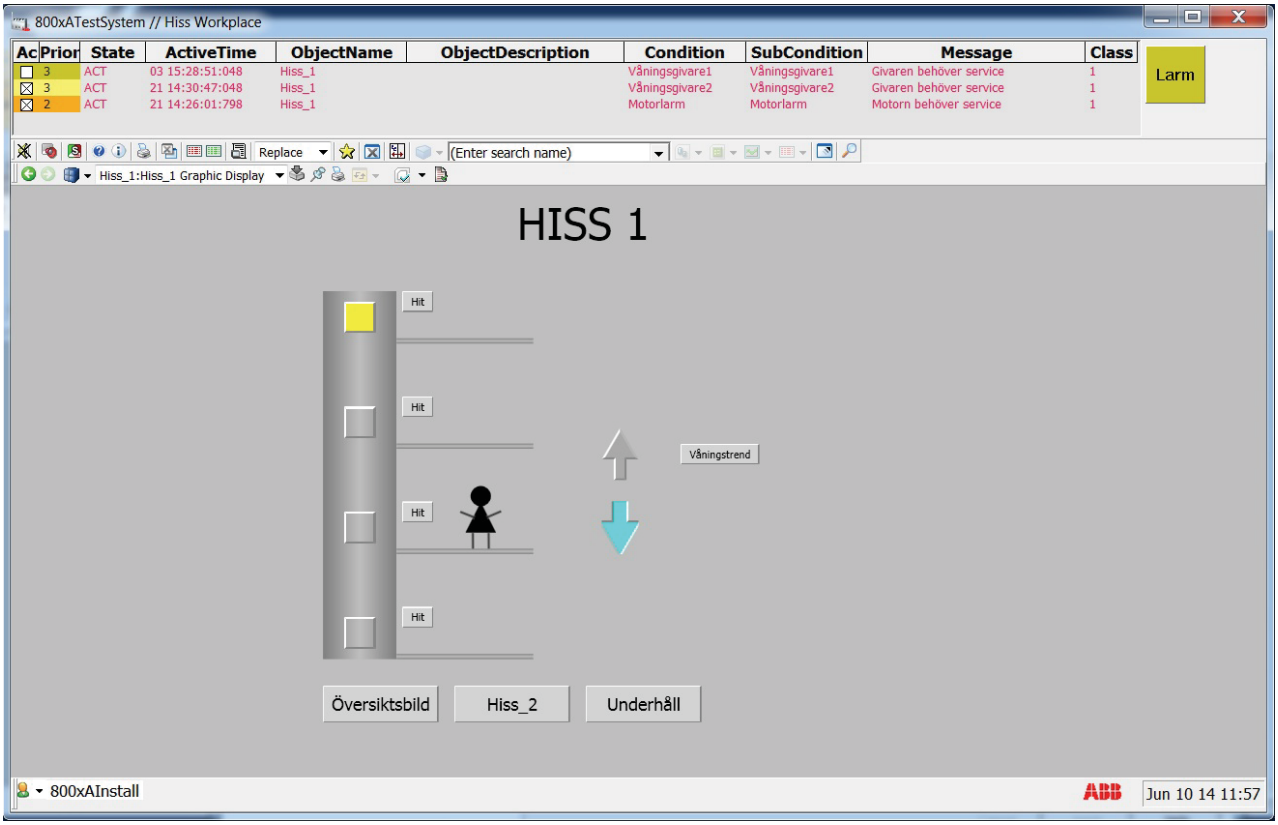
CHALMERS



Utbildningsanläggning för ABB:s SCADA-system

Examensarbete inom Högskoleingenjörsprogrammet Mekatronik

Anna Nordevik

Institutionen för signaler och system
Chalmers Tekniska Högskola
Göteborg, Sverige 2014

Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Göteborg, Sverige, 2014

FÖRORD

Detta examensarbete är ett avslutande projekt av den 3 åriga Högskoleingenjörsutbildningen med inriktning Mekatronik på Chalmers tekniska högskola.
Examensarbetet har utförts på uppdrag av ÅF i Göteborg där tanken med examensarbetet är att sätta upp ett system som sedan kan fungera som utbildning för övriga medarbetare.

Tack till:
Handledare på ÅF, Sofia Holmblad.
Patrik Kerttu, för möjligheten att göra examensarbete på ÅF.
Morgan Osbeck, handledare på Chalmers för stor hjälp med rapportskrivning.

SAMMANFATTNING

Övervakning av processer blir allt viktigare i den tekniska världen där människor skall styra och övervaka större och större anläggningar. Projektet syftar på att utforma ett styr och övervakningssystem som skall kunna användas för upplärning av medarbetare på ÅF. Ett SCADA system har satts upp där ABB:s SCADA-programvara 800xA, PLC AC 800M samt ABB:s OPC server har använts. Ett Styr och Övervakningssystem så kallat SCADA system har satts upp och konfigurerats, för att få ett fungerande SCADA system har PLC programmerats, OPC kommunikation konfigurerats och i 800xA mjukvaran har viktiga delar som grafikbilder larm och historiska trender konfigurerats.

ABSTRACT

Monitoring processes is becoming more and more important in the technical world, when people must monitor much larger production sites. The project aims for designing a control and monitoring system that could be used as a training facility at the company. A control and monitoring system known as a SCADA system has been set up and configured. To get a functional SCADA-system a PLC has been programmed, OPC communication has been configured and the 800xA software has been configured with graphic images, alarms and historical trends.

Keywords: SCADA, ABB, OPC, PLC

INNEHÅLLSFÖRTECKNING

1 Inledning 1
1.1 Bakgrund 1
1.2 Syfte 1
1.3 Avgränsningar 1
1.4 Mål 1
2 Problemställning 2
3 Teknisk bakgrund ..... **Fel! Bokmärket är inte definierat.**
3.1 PLC 3
3.2 SCADA-system 4
3.3 OPC 6
4 Systembeskrivning ..... **Fel! Bokmärket är inte definierat.**
5 Metod 10
6 Genomförande 11
6.1 Kommunikation PLC Controlbuilder 11
6.2 PLC Programmering 11
6.2.1 Kravspecifikation 11
6.2.2 Utbildning i Control Builder 12
6.2.3 Hisstyrning 15
6.2.4 Larmhantering 17
6.3 Programvaran 800xA 18
6.3.1 Inläsning och upplärning 18
6.3.2 OPC DA kommunikation 18
6.3.3 Alarm and Event 18
6.3.4 Graphic builder 20
6.3.5 Historian and trend 22
6.3.6 Workplace 23
7 Resultat 25
7.1 Rekommendationer till fortsatt arbete 25
8 Diskussion 26
9 Referenser 27
Appendix A 28

BETECKNINGAR

ABB	Automationsföretag
DCOM	Språk för att kommunicera mellan mjukvara.
FB	*Funktion block*, logiska funktioner hopbyggda till olika funktioner tex timers och räknare.
FBD	*Function block diagram*, grafiskt programmeringsspråk där logiska funktioner används tillsammans med funktionsblock.
HMI	*Human Machine Interface*, användargränssnitt mellan människa och maskin. Exempelvis en operatörsbild.
I/O	Ingång/Utgång
IP adress	Adress i ett nätverk
LD	*Ladder diagram*, grafiskt programmeringsspråk väldigt likt el schema.
OLE	*Objekt Linking and Embedding,* Dataöverföringsprotokoll mellan program i datorn.
OPC	*OLE for Process Control*, Koppling för dataöverföring mellan PC och PLC
PC	Persondator
PLC	*Programmable Logic Controller,* datorbaserat styrsystem består av processor, minne, ingångar och utgångar.
SCADA	*Supervisory Control And Data Acquisition*, System för att styra och övervaka processer.
SFC	Sequential function charts, sekventiellt programmeringsspråk.
ST	*Structured text*, textbaserat programmeringsspråk
TCP	Dataöverföringsprotokoll där mottagaren verifierar att datapaketet kommit fram.

1 INLEDNING

Här beskrivs bakgrund, syfte, avgränsningar och mål för projektet.

1.1 Bakgrund

Övervakning av industriella processer blir allt viktigare när anläggningarna blir större och större. ÅF är en konsultfirma som har ett brett kunnande och stor erfarenhet inom automationsområdet. Under åren har ÅF byggt upp ett antal mindre processer för olika styrsystem som används som labbutrustning och utbildningsmaterial för medarbetare på företaget. ÅF vill nu utveckla sin labbmiljö genom att övervaka processerna i ett gemensamt SCADA (supervisory control and data aquisition) system. SCADA systemet ska senare fungera som lärosystem för övriga medarbetare för att behålla och förbättra kompetensen i företaget.

1.2 Syfte

Syftet med projektet är att skapa ett system som övervakar och styr två processer, två hissar med styrsystem från ABB. Syftet är även att skapa sig förståelse om varför övervakande system används, vilka delar som ingår i ett SCADA-system och hur kommunikation mellan det övervakande systemet och processerna fungerar.

1.3 Avgränsningar

Projektet avgränsas till ett övervakande system och två processer som skall övervakas. Avgränsning görs även till användning av endast ABB produkter, PLC, SCADA-programvara och OPC-server.

1.4 Mål

- Sätta upp ett fungerande styr- och övervakningssystem.
- Programmera en fungerande hisstyrning i PLC (programmable logic controller).
- Konfigurera en fungerande arbetsstation varifrån processen kan köras och övervakas
- Konfigurera operatörsbilder som med signalinformation från PLC grafiskt kan visa processens tillstånd och där signaler kan skickas till PLC:t för att styra processen. Operatörsbilder skall göras i form av översiktsbild, hissbild och underhållsbild.
- Historik över hur hissen har rört sig skall finnas och en trend skall kunna visas och finnas tillgänglig från operatörsbilden.
- Fungerande process- och underhållslarm i både PLC och övervakande system.

2 PROBLEMSTÄLLNING

Projektet går ut på att konfigurera ett styr och övervakningssystem så kallat SCADA-system. Programvaran som skall användas är från ABB och i denna programvara skall de olika funktionerna som behövs konfigureras.

Processerna består av två PLC-er som styr var sin hissmodell. Hissen består av en hisskorg som kan åka mellan fyra våningsplan. På varje våning finns det en knapp med inbyggd lampa, om man trycker på knappen ska lampan tändas och hissen åka till den beställda våningen. Det finns även knappar med lampa inuti hisskorgen där samma funktion tillämpas, knappen för den våning som önskas trycks in och lampan i knappen tänder sig. Sedan går hissen till den önskade våningen. För att hissen skall veta var den befinner sig finns det på varje våning en givare som känner av ifall hisskorgen är där eller inte.

För att logiskt styra de två hissarna programmeras två PLC-er av typen AC 800M. Till detta används ABB:s programmeringsverktyg Control Builder.

Konfiguration av SCADA-systemet skall därefter ske. SCADA-systemet skall kunna styra processerna och även kunna visa var de olika hissarna befinner sig. I SCADA-systemet skall även larm kunna visas. Både underhållslarm och processlarm skall finnas och dessa skall om det behövs kunna återställas. För att kunna se vad som hänt tidigare skall sidor för historik konfigureras och för att om möjligt förutse vad som är på gång i processen skall trender visas. Dessa trender skall lätt finnas tillgängliga från operatörspanelen.

3 TEKNISK BAKGRUND

Här ges en kort beskrivning av några tekniska delar i projektet PLC, SCADA och OPC.

3.1 PLC

PLC står för Programmable logic controller. En PLC är ett programmerbart styrsystem som används för att styra olika processer. PLC:n består av olika moduler som kan kopplas ihop och utökas beroende på vad för process som skall styras. I en PLC finns alltid en processor, denna hanterar information från ingångs- och utgångsmoduler (1). Till ingångsmodulen är givare kopplade, givaren känner av statusen på processen och skickar elektriska signaler till ingångskortet. Ingångar kan hanteras som antingen digitala signaler där givaren känner av om den är påverkad eller inte, eller som analoga signaler där variation inom ett område kan kännas av. Från utgångsmodulen skickas elektriska signaler, dessa kan skickas som antingen analoga signaler eller som digitala signaler. De digitala signalerna är antingen på eller av, ex. en lampa ska tändas. De analoga signalerna kan variera inom ett bestämt område ex. lampan ska dimmas. PLC:n kan även ha moduler för kommunikation till externa enheter via olika seriella protokoll.

Vid programmering av PLC kan flera olika programmeringsspråk användas och för att inte alla tillverkare skall ha egna programmeringsspråk har en standard utvecklats. Denna standard heter IEC 61131 och publicerades första gången 1993. IEC 61131 definierar fem olika programmeringsspråk där tre är grafiska och två är textbaserade (2). De fem programmeringsspråk som är standardiserade är

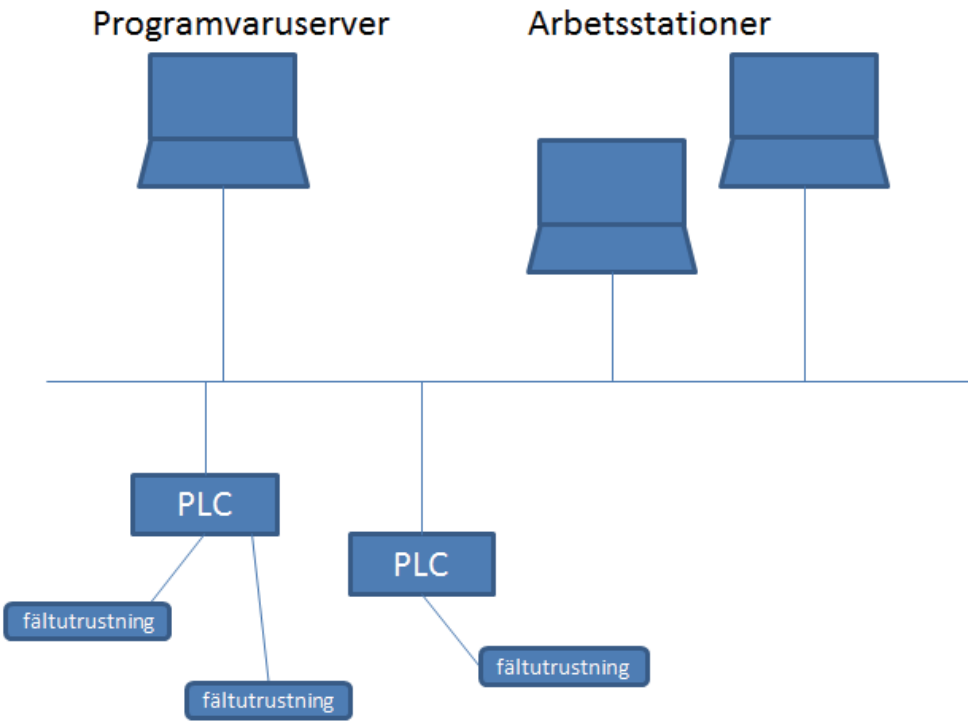
- Ladder diagram LD
- Function block diagram FBD
- Sequential function chart SFC
- Instruction list IL
- Structured Text ST

Ladder Diagram, LD är ett grafiskt språk som påminner mycket om el-schemaritningar. Genom att använda sig av detta språk vid introduktion av PLC behövde inte en hel ingenjörskår utbildas i ett nytt sätt att tänka och detta gjorde introduktionen av PLC enklare.
Function Block Diagram, FBD är ett grafiskt språk där logiska grindar (and, or, not) bygger upp programmet tillsammans med Funktionsblock, FB som är logiska grindar hopbyggda till mer avancerade funktioner som räknare eller timers. SFC Sequential Function Chart är det tredje grafiska språket, det är uppbyggt som en sekvens där olika steg skall gås igenom. För att programmet skall få gå från ett steg till ett annat måste villkor vara uppfyllda.

Instruction list, IL och Structured text, ST är de två textbaserade programmeringsspråken i IEC 61131 där IL är uppbyggt med logik och beskriver mycket enkelt vad som sker likt assembler programmering medan ST är mer likt ett något högre programmeringsspråk som C programmering där bl.a. if, while och for satser kan användas.

3.2 SCADA-system

SCADA står för Supervisory Control And Data Acquisition och är ett system som används för övervakning och styrning av processer. Flera olika system från olika fabrikat och leverantörer kan knytas ihop till ett SCADA-system. Detta görs för att få en bra överblick av hur anläggningen ser ut och hur processen körs.



Figur 3.1 Uppbyggnad av ett SCADA-system.

”Det typiska SCADA-systemet består av arbetsstationer för operatörer, programvaruserver för SCADA programvara, nätverk för kommunikation, programmerbara styrsystem och fältutrustning” (3) se figur 3.1. Ett SCADA-system kan vara mycket större än vad bilden visar med fler styrsystem och fler operatörspaneler men kan även vara mindre än vad bilden visar. Det elementära är att alla byggstenar behöver finnas med för att det ska kunna kallas ett SCADA-system. De fem delarna i ett SCADA-system punktas upp och beskrivs mer nedan.

- Fältutrustning
- Programmerbara styrsystem
- Arbetsstationer för operatörer
- Programvaruserver för SCADA programvara
- Nätverk för kommunikation

SCADA-systemet kan även delas upp i två delar där programvaruservern med SCADA programvaran tillsammans med operatörsstationerna bildar ett överordnat system för övervakning och styrning och där de programmerbara styrsystemen tillsammans med fältutrustningen bildar ett underordnat system som tar order från det överordnade systemet men ändå kan styra processen ifall inget kommando kommer från det överordnade systemet.

Fältutrustningen är den utrustning som gör att processen kan styras och övervakas av styrsystemet. Dessa är kopplade till ingångs och utgångsmoduler i styrsystemet men kan även anslutas via seriella protokoll. Fältutrustning som vanligen styrs från styrsystemet är motorer, pumpar och ventiler och

för att veta hur styrningen skall gå till behövs givare som talar om hur processen körs så som flödesmätare, temperaturgivare och tryckmätare.

I de programmerbara styrsystemen även kallade PLC programmeras hur processen skall styras och övervakas. Här programmeras olika sekvenser och program för att få processen automatiskt styrd med så lite inblandning av människan som möjligt. Ofta programmeras delar av processen så att val mellan automatisk eller manuell styrning kan ske så att operatören kan ta över och köra systemet vid speciella tillfällen.

För att operatörer skall kunna övervaka systemet behöver alltid minst en operatörsstation finnas i ett SCADA system. På dessa datorer konfigureras arbetsstationer som kan utformas specifikt för varje dator. I en arbetsstation behöver bilder över hur processen ser ut finnas, i dessa bilder skall operatören kunna se hur processen styrs och även kunna styra själv ifall någonting oväntat händer. Larm över fel i processen är viktigt att visa för operatören och för att kunna se tillbaka på hur processen styrts är historiska trender en viktig del.

I programvaruservern för SCADA-systemet finns SCADA-programvaran, denna programvara hämtar signaler från styrsystemet för presentation i arbetsstationen. Om styrning från arbetsstationen sker skickas dessa signaler via programvaran för vidarebefordring till styrsystemet. All insamlad historik lagras även i databaser på programvaruservern.

För att kunna kommunicera och skicka data mellan de olika systemen, programvaruserver, arbetsstation och PLC måste ett nätverk finnas. Detta är oftast lokala nätverk där man kommunicerar via Ethernet. Varför lokala nätverk ofta används är för att undvika att obehöriga kommer in i systemet så som virus eller hackare. Finns ingen anslutning till omvärlden är risken inte lika stor att bli utsatt för en attack.

För styrning och övervakning i ett SCADA system är det viktigaste inte att operatören får så mycket information som möjligt från systemet, utan att den relevanta informationen presenteras på ett bra sätt så att det blir överskådligt och enkelt för operatören. Det är därför viktigt att operatörsbilderna är utformade på ett sådant sätt att det inte blir jobbigt att titta på dem för länge. Lugna färger som olika gråa nyanser bör användas till sådant som inte måste påkalla operatörens uppmärksamhet medan om någonting händer skall detta påkallas med hjälp av klara färger, om en ventil stänger eller om en motor stannar skall detta påkalla uppmärksamhet.

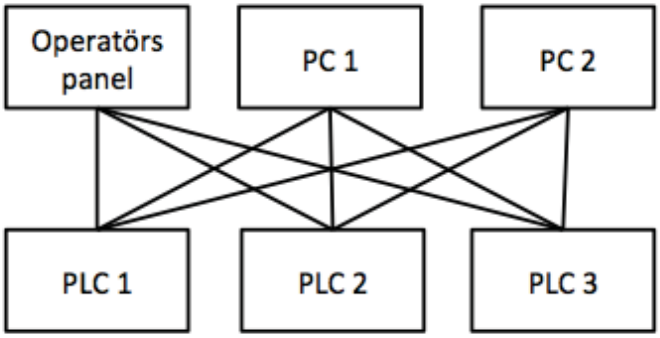
Larm används mycket i SCADA system för att göra operatören uppmärksam på att någonting skett. Ofta görs operatören uppmärksam med både ljud och ljus och beroende på hur allvarligt larmet är konfigureras ljudsignalerna på olika sätt och ljussignalerna kan exempelvis delas upp i rött gult och vitt för olika prioriteringar.

Historik och trender är bra verktyg för att kunna se tillbaka på hur processen har uppfört sig. I felsökningssyfte är trend över en givarsignal eller ventilsignal ibland ovärderlig eftersom det finns väldigt mycket information i dessa. De signaler som oftast loggas i historik och trender är analoga signaler. De analoga signalerna som exempelvis tryck eller temperatur kan sedan användas för att se tillbaka på hur processen reagerat under särskilda förhållanden.

3.3 OPC

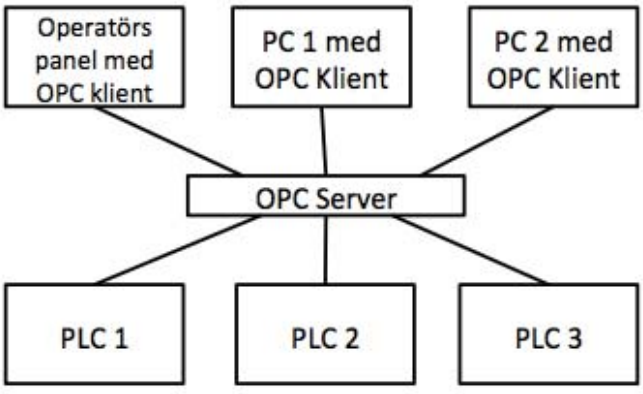
OPC står för OLE for Process Control där OLE är Microsofts sätt att kommunicera mellan olika program internt i datorn. OPC är en standard för dataöverföring som utvecklats för att göra olika tekniska system kompatibla med varandra. Detta för att få en säker och tillförlitlig dataöverföring inom automation och industrier. OPC är en öppen standard som används för att koppla ihop och kommunicera mellan överordnade system som PC och externa enheter ex. PLC.

Vid kommunikation mellan flera enheter från olika fabrikat behövdes tidigare en drivrutin för varje anslutning utvecklas, i figur 3.2 visas dessa drivrutiner som länken mellan de olika systemen. Detta gjorde att konfigurationen blev väldigt tids- och arbetskrävande. Vid kommunikation mellan endast ett fåtal enheter behövdes ett flertal specialanpassade drivrutiner.



Figur 3.2 Kommunikation utan OPC

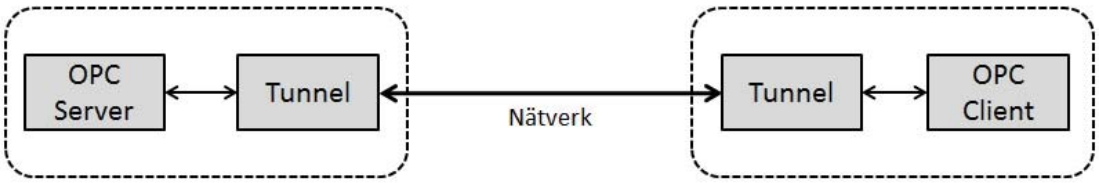
För att undvika detta och spara tid utvecklades OPC. OPC standarden är uppbyggt med server klient förhållande där OPC serverns uppgift är att tillhandahålla information till klienterna från de underordnade systemen (ex. PLC) se figur 3.3.



Figur 3 Kommunikation med OPC

OPC servern använder sig av den specifika drivrutinen som behövs för att kommunicera med PLC 1, 2 eller 3. Servern hämtar information från de olika PLC och lagrar denna. Klientens uppgift är sedan att hämta information från servern eller skicka information till servern. Detta gör att klienten med endast en drivrutin mot servern kan kommunicera med alla PLC i systemet. OPC standarden gjorde det mycket enklare att få olika system att kommunicera med varandra och tidsåtgången för implementering mellan olika system minskades drastiskt.

Ifall en OPC server ligger på en annan dator än den som klienten befinner sig på så kan en OPC tunnel behövas för att klienten skall kunna kommunicera med servern. Vid intern kommunikation i datorer eller vid kommunikation från dator till styrsystem fungerar direkt OPC kommunikation bra men när kommunikationen skall ske via en annan dator så kan problem uppstå och därför används en OPC tunnel för att öppna en fri väg för kommunikationen (4). Tunnelprogramvaran installeras på de två datorerna som skall kommunicera med varandra och sedan upprättasen kommunikationslänk mellan datorerna. OPC-tunnelprogrammet kommunicerar sedan med OPC systemet på datorn och med den andra datorns tunnelprogram. Genom att översätta OPC meddelandet från server eller klient till TCP och sedan skicka meddelandet över nätverket fås en tillförlitlig dataöverföring. När sedan ett TCP meddelande tas emot översätts detta till OPC och OPC systemet kan kommunicera detta till sin externa enhet se figur 3.4.



Figur 3.4 OPC kommunikation

På grund av olika krav från industrin har det på senare år utvecklats tre olika OPC standarder. En för dataåtkomst *Data Access* (DA) en för larm och händelser *Alarm & Event* (A&E) och en för historik *Historical Data Access* (HDA) (5).

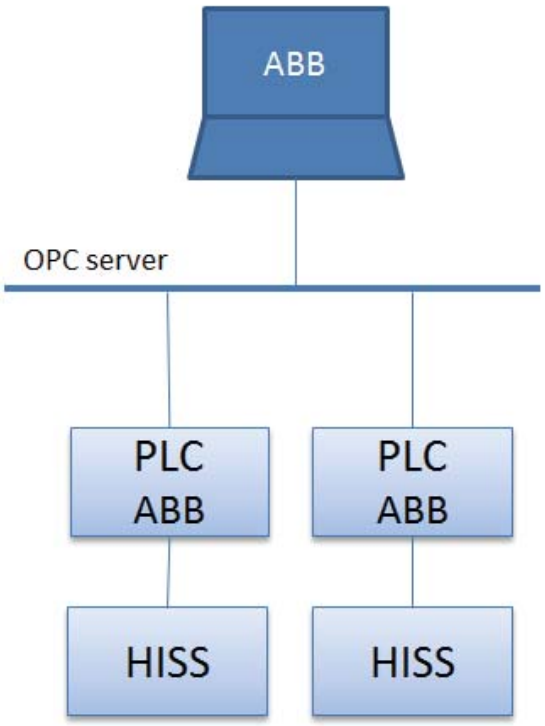
OPC Data Access används för läsning, skrivning och övervakning av signaler till och från PLC eller andra programmerbara styrsystem. OPC DA standarden är den viktigaste av standarderna när ett SCADA system sätts upp då det är detta protokoll som skickar live data mellan PLC och arbetsstation. Från SCADA programvaran får OPC DA klienten de variabler den skall bevaka och skickar sedan en förfrågan till OPC DA servern. Servern kommer då hämta variablerna från styrsystemet och gruppera dem. När variablerna som klienten önskat är grupperade i servern kan klienten läsa och skriva till de valda variablerna och servern vidarebefordrar till styrsystemet.

OPC Alarm & Events standarden används vi hämtning av händelser i styrsystemet. Händelser beskrivs som meddelanden från systemet medan ett larm är en förändring av en processvariabel. När OPC A&E klienten ber om meddelanden från OPC A&E servern hämtar servern alla händelser och klienten får sedan sortera händelserna genom filter. Skillnaden mellan OPC DA och OPC A&E är att här kan inte klienten be om specifika händelser utan alla händelser distribueras från styrsystemet. När ett larm hämtas till klienten behöver detta ofta kvitteras, kvitteringen är möjlig via OPC A&E protokollet och utförs ofta från en arbetsstation.

OPC Historical Data Access ger användaren tillgång till sparat material, där allt från enkla dataloggnings program till avancerade SCADA system kan ta del av informationen. Genom att OPC HDA klienten begär att servern loggar data med specifik sampeltid och lagringstid kan sedan klienten ta del av den information den begärt på tre olika sätt. Genom att läsa rådata från historiken under ett specifikt tidsintervall för en eller flera variabler, klienten kan även begära att få information om en eller flera variabler vid ett specifikt tillfälle eller så kan klienten begära ett beräknat totalvärde för en eller flera variabler under en viss tid.

4 SYSTEMBESKRIVNING

För att få en bra överblick av systemet ges här en presentation utifrån figur 4.1.



Figur 4.1 Topologi över systemet

ABB systemet består av två PLC-er och en dator. Varje PLC är ansluten till var sin process som i detta fall består av två hissar dvs. en hiss till varje PLC. Datorn innehåller Programvaran ABB 800xA och kan därför ses som programvaruserver samtidigt som arbetsstation.

En hiss består av fyra våningsplan där det på varje våningsplan sitter en givare som känner av ifall hisskorgen befinner sig där. Hissmodellen består även av ett antal tryckknappar med inbyggda lampor. Det sitter en knapp på varje våningsplan och ett sett med knappar i hisskorgen. Hissen består även av en motor som kör hisskorgen uppåt eller nedåt. Ingångar och utgångar är kopplade direkt till ingångs och utgångsmodulerna på PLC:n.

För att kunna programmera de två PLC av typen AC 800M som styr hissarna är ABB:s programmeringsverktyg Control Builder installerat på datorn. Vid programmering av hissarna används IEC 61131 standardiserade programmeringsspråk. Kommunikationen vid programmering av PLC görs via Ethernet.

På datorn är även programvaran ABB 800xA installerad. ABB:s 800xA programvara är en SCADA-programvara riktad mot processindustri som med hjälp av olika programmoduler kan bygga upp ett system för att styra och övervaka processer. I konfigurationsfönstret Engineering Workplace fås tillgång till de olika biblioteken som programmet är uppbyggt av och där konfigurationen utförs. Biblioteken är uppbyggda av mallar som beskriver hur systemet skall fungera. Varje konfiguration har en specifik mall som beskriver hur just den delen av programmet skall se ut och fungera. De olika programmoduler som använts är Graphic builder där

operatörsbilder utformas, i Alarm and Event modulen konfigureras hur larm skall presenteras i systemet och Historian and Trend som är historikmodulen där konfiguration sker för hur historik skall hämtas och trender ska visas. För att kunna visa alla dessa moduler på ett enkelt sätt har en arbetsstation (workplace) utformats för hisstyrningen där de olika moduler som byggts upp presenteras.

Mellan datorn där 800xA är installerat och PLC ABB AC 800M finns en OPC kommunikation så att data kan skickas och hämtas mellan systemen. I detta projekt används ABB:s OPC server som är installerad på datorn och hämtar informationen från de två PLC till klienten som är automationssystemet.

5 METOD

En kravspecifikation och flödesschema över hur processen skulle fungera togs fram. Därefter gjordes en kontroll av ingångar och utgångar så att dessa var inkopplade i PLC:t och fungerande.

Kontakt mellan dator och PLC upprättades över Ethernet.

För att komma igång med PLC programmering och inlärning av ett nytt system genomfördes en utbildning i ABB:s utvecklingsprogram Control Builder. Utifrån kravspecifikation och flödesschema utformades PLC programmet för att kunna köra processen. Ett SFC program fick bli grunden till styrningen och till detta kopplades flera program som behövdes för att få processen att fungera som tänkt.

För att sätta sig in i 800xA systemet genomfördes ytterligare en utbildning där de delar som skulle konfigureras i systemet gicks igenom.

Kommunikation mellan automationssystemet 800xA och PLC 800M upprättades via OPC för att få kontakt mellan PLC och 800xA. För denna kommunikation används ABB:s OPC server.

Operatörsbilder konfigurerades i Graphic Builder så att operatören kan köra processen. Olika operatörsbilder för de olika hissarna och även en underhållsbild konfigurerades för att kunna se när underhåll behöver utföras.

Larmmodulen konfigureras så att larm kan hanteras. Larm konfigurerades även i PLC programmet.

Konfiguration av historikmodulen gjordes för att kunna se tillbaka på tidigare händelser.

En arbetsstation utformades så att processen blir överskådlig och lätt att styra.

6 GENOMFÖRANDE

Kapitlet beskriver hur projektet har genomförts.

6.1 Kommunikation PLC Controlbuilder

För att få kontakt mellan datorn och de två PLC som använts sattes ett litet nätverk upp. Via en switch kopplades PLC ihop med PC och för att ta reda på IP adresserna till de två PLC öppnades konfigureringsverktyget Control Builder. Under *tools* > *maintenance* > *remote system* visades alla system som var ihopkopplade med varandra och här visades IP adresserna till de båda PLC.

Genom att sätta en fast IP-adress i datorn kunde kontakt upprättas mellan PLC och PC. Att sätta en fast IP-adress i datorn gjordes genom att i kontrollpanelen öppna *nätverk och delningscenter* och välja *anslutning till lokalt nätverk*. Här valdes *egenskaper* och sedan *Internet Protocol Version 4 (TCP/IPv4)*. IP adressen som fanns i PLC var en av de adresser som får användas på lokala nätverk (6) och därför valdes en IP-adress till PC:n som gör att PLC och PC ligger på samma nätverk.

6.2 PLC Programmering

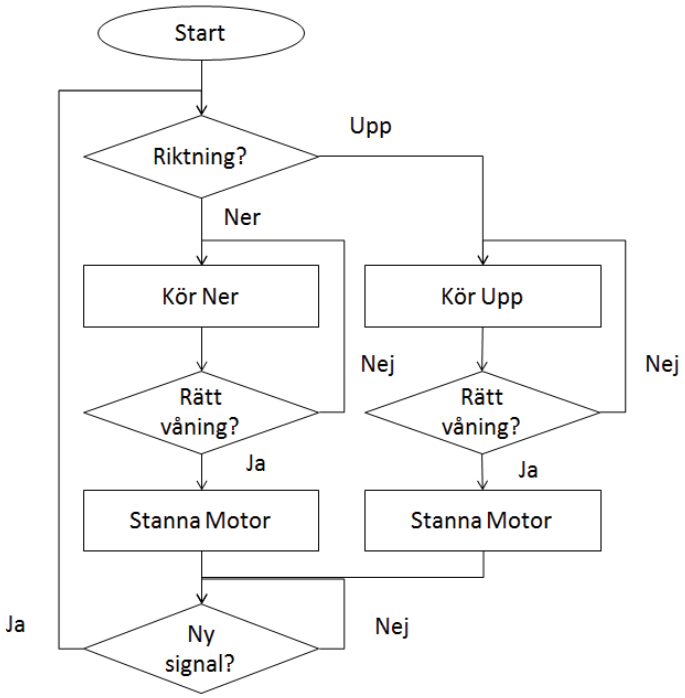
Här beskrivs arbetsgången vid programmeringen av hissprocessen.

6.2.1 Kravspecifikation

Till att börja med sattes en kravspecifikation upp för att få en övergripande bild över hur hissen skulle fungera. Utefter den ritades sedan ett flödesschema se figur 6.1 och efter det programmerades sedan PLC:n.

Krav för hissfunktion:

1. När en knapp på någon våning trycks in skall hissen gå till den beställda våningen och stanna där.
2. När en våning beställts skall lampan på beställd våning tändas till dess att hissen kommit till våningen.
3. Om flera våningsplan i rad är beställda skall hissen gå till det våningsplan som är närmast utgångspunkten först.
4. Om våningsplan är beställda både ovanför och under hissens nuvarande position skall hissen ta alla ordrar åt samma håll först innan den byter riktning.
5. Om ingen våningsgivare är påverkad under 3 sekunder skall larm ges om att hissen inte hittar en våning.
6. När en givare blivit påverkad 5 gånger skall larm ges så att service kan utföras på givaren.
7. När motorn gått i 1 minut skall larm ges så att service kan utföras på motorn.
8. Trend skall kunna visas över vilka våningsplan hissen har befunnit sig på.

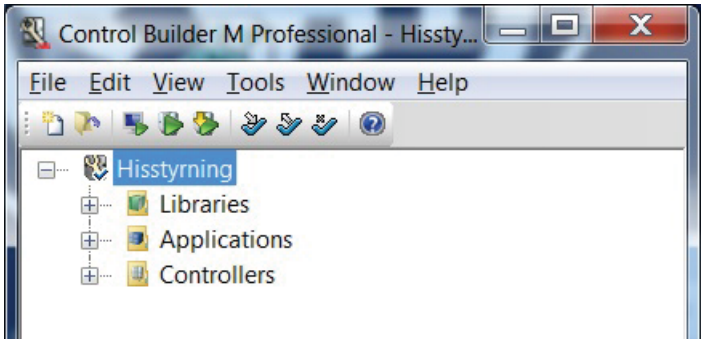


Figur 5.1 Flödesschema för hiss sekvens

6.2.2 Utbildning i Control Builder

Då ABB var ett helt nytt system att arbeta med genomfördes till att börja med en utbildning i ABB:s utvecklingsmiljö Control Builder. Utbildningsmaterialet hade tillhandahållits av ABB under tidigare utbildning på företaget. Detta var ett bra sätt att sätta sig in i systemet och få en bra grund i denna utvecklingsmiljö.

I Control Builder skapades projektet som arbetet skulle utföras i *Hisstyrning.* När ett nytt projekt skapas, skapas alltid tre huvudmappar *Libraries, Applications* och *Controllers* och under dessa byggs sedan projekt upp se figur 6.2.

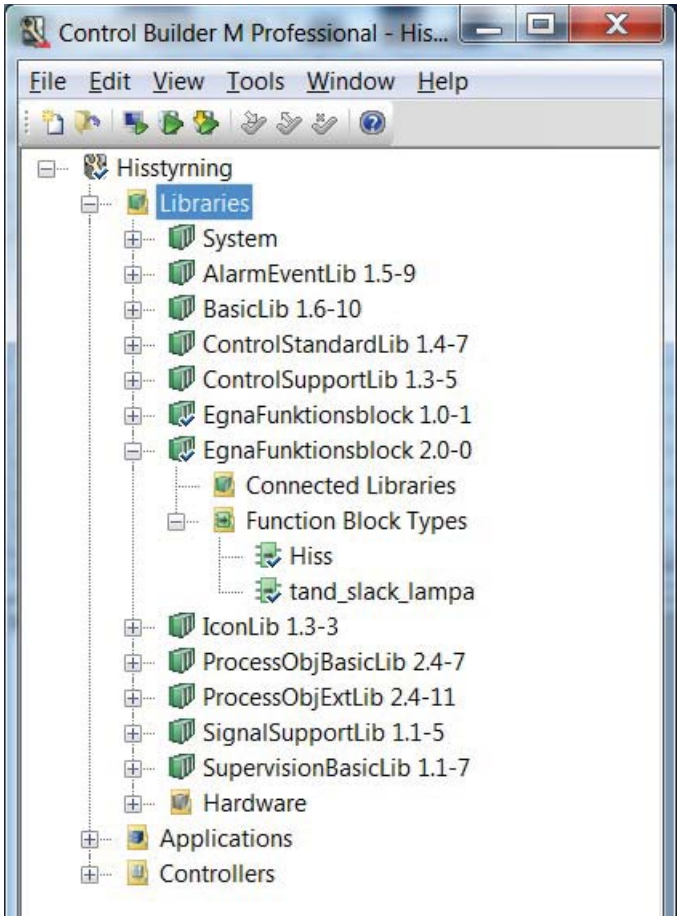


Figur 6.2 Grundstrukturen i Control Builder

6.2.2.1 Libraries

Under biblioteksmappen lades de bibliotek som behövdes för att skriva projektet in. Detta gjordes genom att högerklicka på biblioteksmappen och välja *Insert Library*. Ett bibliotek är en samling med olika funktionsblock. Det vanligaste biblioteket är BasicLib och i detta bibliotek finns de vanligaste funktionsblocken som räknare och timers. Under projektets gång lades de bibliotek som behövdes till för att enkelt kunna programmera hissen.

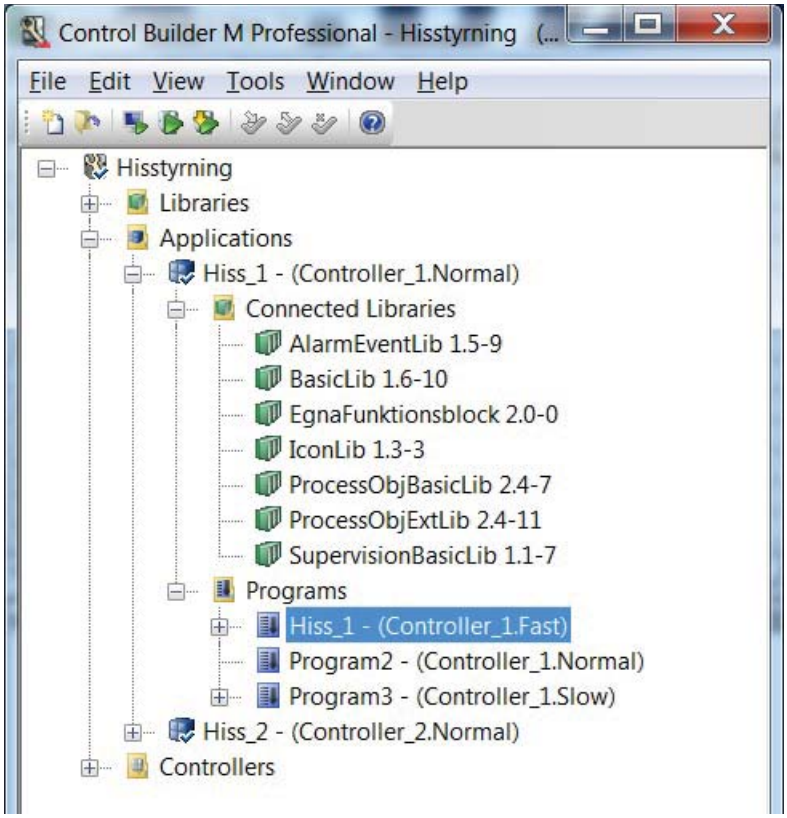
Under biblioteksmappen kan även egna bibliotek läggas till. Detta gjordes genom att högerklicka på biblioteksmappen och välja nytt bibliotek och under det nya biblioteket skapa en ny mapp för funktionsblock. Här tillverkades ett eget funktionsblock för tändning och släckning av lampa i knapp se figur 6.3, utförligare beskrivning av funktionsblocket ges i kap. 6.2.3 Hisstyrning.



Figur 6.3 Biblioteksmappens struktur

6.2.2.2 Applications

De två hissar som skulle styras lades till i mappen för applikationer. Under varje hiss tillverkades sedan automatiskt två nya mappar, en för anslutna bibliotek och en för program. I mappen för anslutna bibliotek kunde de bibliotek som skulle vara anslutna till just denna hiss läggas till. Endast de bibliotek som redan hade lagts till under biblioteksmappen gick att ansluta till hissapplikationen. Detta gör att alla bibliotek som lagts till under bibliotheksstrukturen inte behöver användas i alla applikationer. Under programmappen tillverkas sedan ett program och i programmet skrivs sedan den kod som behövs för att styra processen se figur 6.4.

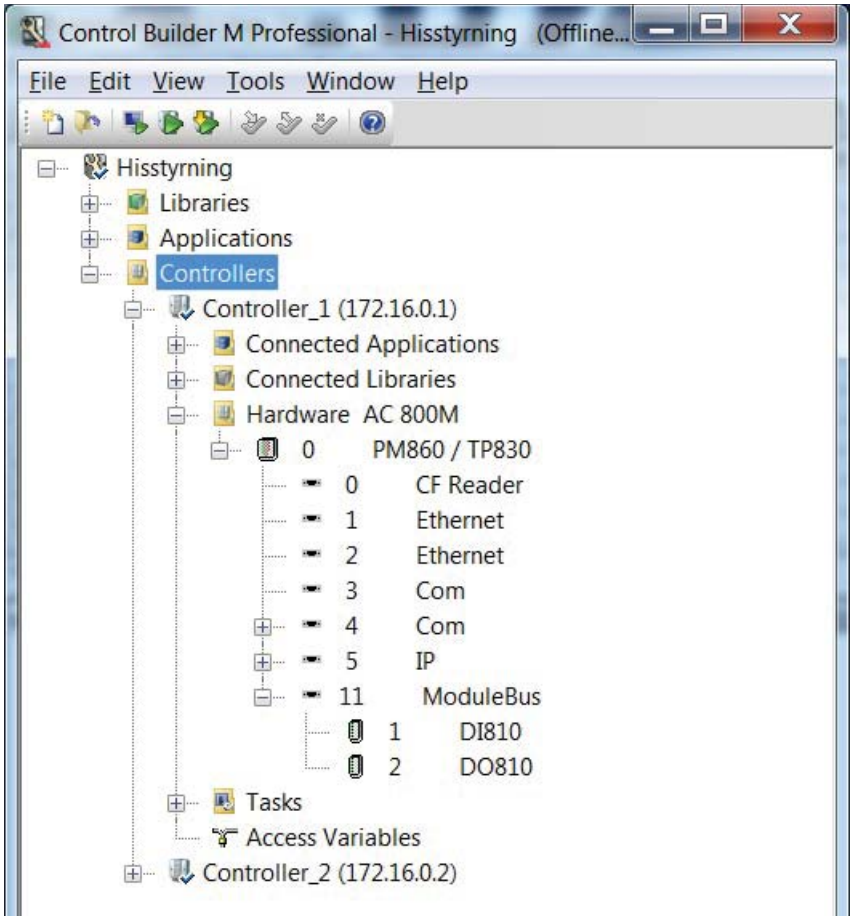


Figur 6.4 Applications strukturen med anslutna bibliotek och programmappen.

I programmet tillverkades olika kodblock och för varje block kunde programmeringsspråk väljas. Detta gjorde att det programmeringsspråk som passade bäst till den specifika funktionen valdes.

6.2.2.3 Controllers

Under controllersmappen anslöts den hårdvara som skulle användas. PLC för hiss ett och två anslöts genom att lägga till en ny controller, välja vilken sorts hårdvara som anslutits och sedan lades de moduler som var anslutna till PLC:n in se figur 6.5. I detta projekt har varje PLC en digital ingångsmodul och en digital utgångsmodul.



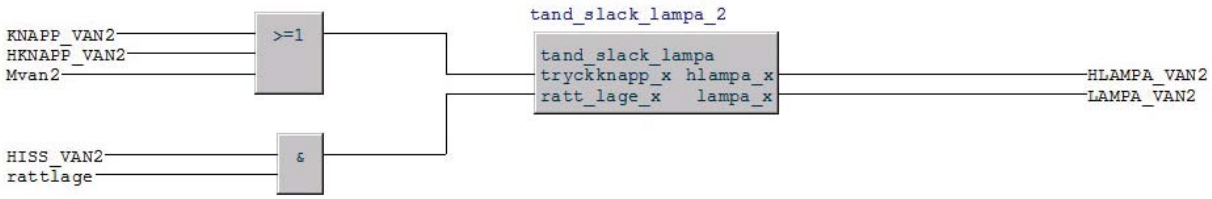
Figur 6.5 Controllerstrukturen med hårdvaran ansluten.

När hårdvaran var konfigurerad kopplades IP adressen, som tidigare tagits reda på via *remote systems*, till respektive PLC. För att kunna ladda ner programmet som skrivits till PLC:n behövde IP adressen skrivas in under *Controller, PM860/TP830, och Ethernet* se figur 6.5. I figuren syns även andra sätt som PLC:n kan kommunicera på.

6.2.3 Hisstyrning

Här beskrivs hur programmeringen av de olika funktionerna i hissen gick till.

Programmet för tändning och släckning av lamporna i knapparna programmerades som ett eget funktionsblock. Funktionsblocket är egentligen en vanlig RS-vippa men för att kunna styra två utgångar samtidigt och få en överskådlig programmering gjordes ett funktionsblock se figur 6.6. Om en knapp trycks in sätts utgången för lampan och lampan tänds. När hissen är på rätt läge nollställs utgången för lampan och lampan släcks, se flödesschema i figur 6.7. Till ingångarna i funktionsblocket kopplades de knappar som ska göra att lampan tänds och till ingången för rätt läge kopplades de villkor som om de är uppfyllda skall släcka lamporna.

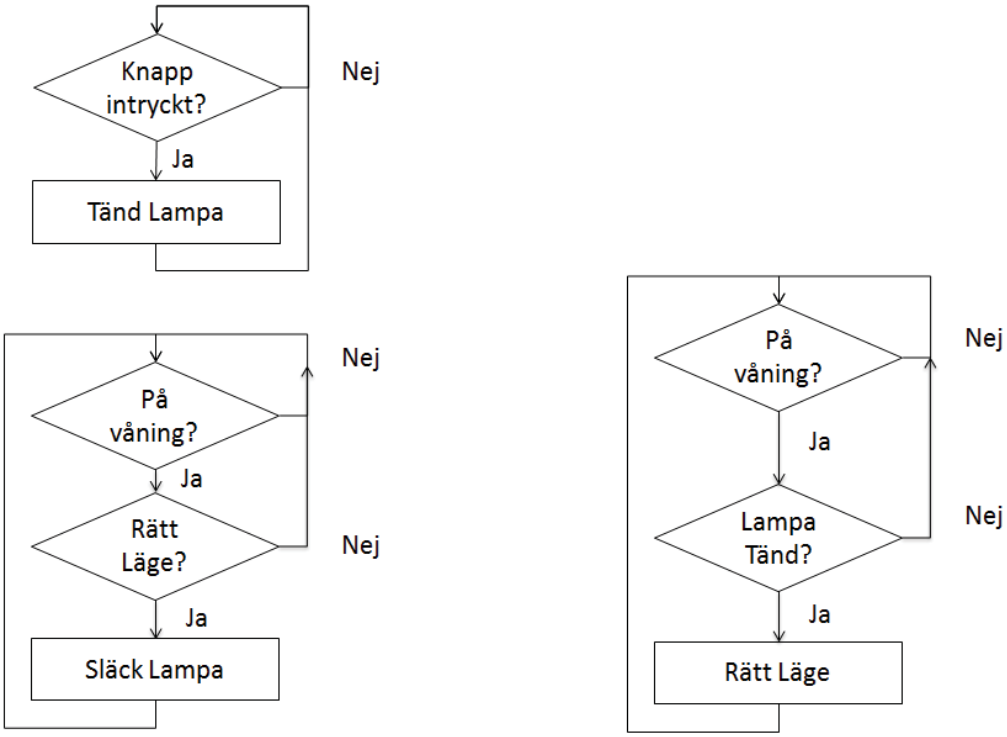


Figur 6.6 Funktionsblock för tändning och släckning av lampa

Variabeln rätt läge programmerades i FBD och blir aktiv ifall hissen befinner sig på en våning där lampan är tänd se flödesschema i figur 6.7. Rätt läge variabeln används både för att släcka lamporna och för att stanna hissen då den har nått en våning som är beställd.

För att bestämma vilken riktning som hissen skall köra åt gjordes ett program i ladder där riktningsvariabeln sätts eller nollställs beroende på vilken våning hissen befinner sig på, vilken våning som är beställd och vilken riktning hissen körde åt förra gången. Om riktningsvariabeln är satt kommer hissen att vilja köra ner och om riktningsvariabeln är nollställd kommer hissen att vilja köra upp. Vid programmering av riktningsvariabeln gjordes en funktionsanalys som sedan implementerades i ett ladder program. Programmet för riktningsvariabeln hittas i Appendix A. Riktningsvariabeln programmerades för att hissen skulle köra klart alla beställningar i samma riktning innan den byter håll.

För att få en fungerande hisstyrning programmerades sedan en sekvens i SFC. Hissen väntar på en beställning, en lampa tänds, och kör sedan upp eller ner beroende på om riktningsvariabeln är satt eller nollställd. När hissen får signal om rätt läge stannar den och nästa sekvens kan starta se figur 6.1.



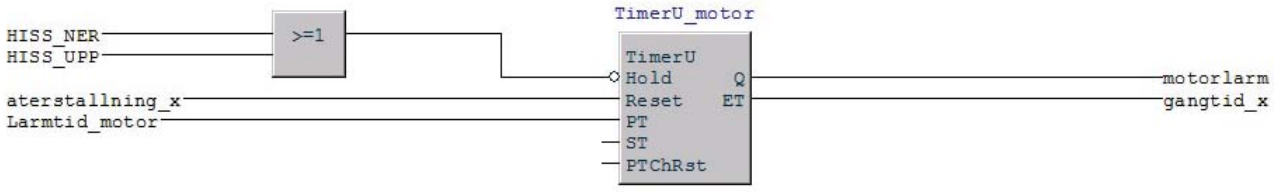
Figur 6.7 Flödesschema för lampor och rätt läge.

6.2.4 Larmhantering

Larmhanteringen i PLC programmet skrevs i FBD och ST, detta för att FBD ger en bra översikt av programmeringen och ST tar mindre plats vid användning av FB eftersom man då endast ser de använda variablerna i funktionsblocket. Funktionsblocket AlarmCond som finns under biblioteket AlarmEventLib gör att larmen skickas till 800xA automatiskt. Larm programmerades för både övervakning av processen och underhåll av processen.

6.2.4.1 Beskrivning av larmen

Underhållslarm för motorn programmerades i PLC:n. Under tiden motorn antingen går uppåt eller nedåt räknar en timer och när motorn har gått en viss tid sätts ett larm för att uppmärksamma operatörerna om att motorn behöver service se figur 6.8.



Figur 6.8 Larmprogrammering FBD i

För att larmet, dvs. ”motorlarm”, skall synas i larmlistan i 800xA användes funktionsblocket AlarmCond då detta funktionsblock skickar variablerna som programmerats direkt till automationssystemet från PLC via OPC A&E. I detta funktionsblock valdes att signalen från timern skall generera larmet till det överordnade systemet. I funktionsblocket bestämdes även vilka variabler som ska bestämma vad som skall stå då larmet presenteras i 800xA se figur 6.9. Variablerna konfigurerades som en textrad och denna text visas i larmlistan då variabeln motorlarm blir aktiv.

```
Motorlarm_1( Signal  := motorlarm,
             SrcName := Motorlarm_1SrcName,
             CondName := Motorlarm_1CondName,
             Message := Motorlarm_1Message );
```

Figur 6.9 Larmprogrammering i ST

För att kunna utföra service på givarna innan de går sönder eller kärvar ihop programmerades ett underhållslarm för varje givare. När givaren varit påverkad ett visst antal gånger sätts ett larm om att service behöver utföras.

Ifall givaren är mekanisk är det bra att utföra underhåll på dem med jämna mellanrum. I detta fall är givaren induktiv och känner av när en metallbit är framför. Om givaren går sönder kommer hissen inte att kunna åka till den beställda våningen.

Ifall hissen inte hittar en våning exempelvis om en givare gått sönder kommer den att stanna mellan våningsplanen. Om hissen inte hittar någon våningsgivare på tre sekunder ges ett larm om att hissen befinner sig mellan våningsplanen, när larmet aktiveras ges efter ytterligare tre sekunder ges signal om att hissen skall köra nedåt tills den hittar en våning, detta för att hissen inte skall fastna mellan våningsplanen utan att ta sig därifrån.

6.3 Programvaran 800xA

Till ABB:s automationssystem 800xA har de två PLC:erna kopplats för att bygga ett SCADA system. I detta projekt har de olika modulerna Alarm and event, Graphic Builder, Historian and Trend och Workplace använts.

6.3.1 Inläsning och upplärning

För att sätta sig in i systemet och lära sig hur programmet fungerar och är uppbyggt har gammalt kursmaterial använts. ABB har även ett stort antal manualer (7) på sin hemsida som tillsammans med hjälpverktyget i 800xA har använts vid upplärning och konfigurering.

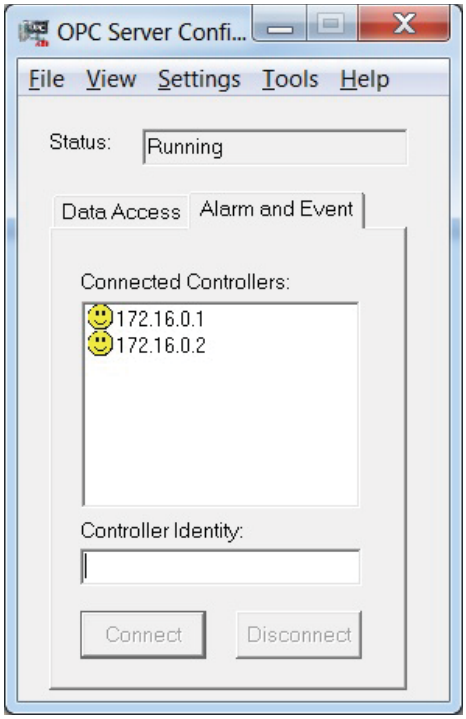
6.3.2 OPC DA kommunikation

För att upprätta kommunikation mellan 800xA och PLC AC800M startades ABB:s egen OPC-server *OPC server for 800M*. Detta är en OPC server som hämtar information från ABB:s PLC 800M. Till denna anslöts de två ABB PLC som finns med i nätverket och skall ha ett datautbyte med SCADA systemet. Genom att ange de aktuella IP-adresserna och trycka på anslut upprättades OPC kommunikation mellan systemen.

För att kunna påverka PLC:n från 800xA behövde nya signaler för OPC kommunikation konfigureras. Dessa signaler läses från grafiska bilder i 800xA och förmedlas till PLC. Under kontrollstrukturen och *Control Network* tillverkades en ny mall av typen *OPC Data Source Definition.* En ny service grupp tillverkades och kopplades till OPC servern. Genom koppling till OPC servern kan nu programmet 800xA via sin OPC-klient hämta och skicka information från och till OPC-servern och därmed till de anslutna PLC.

6.3.3 Alarm and Event

För att larm och händelser skall kunna hämtas till 800xA från PLC upprättades en OPC A&E kommunikation mellan 800xA och PLC. Denna konfigurerades också i ABB:s egen OPC-server genom att skriva in IP adressen till den PLC som skulle generera larm och händelser och trycka på anslut. När den Glada gubben syntes bredvid IP adressen var kontakt upprättad se figur 6.10.

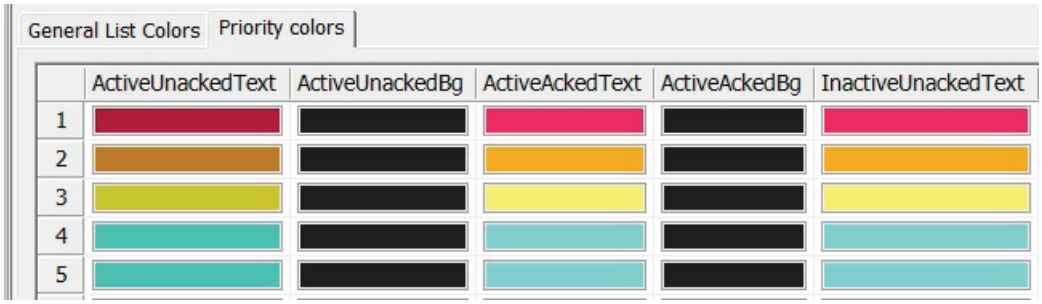


Figur 6.10 OPC kommunikation för larm och händelser

Konfiguration för hämtning av händelser gjordes i 800xA genom att i service strukturen välja *Event Collector, Service*. Här tillverkades ett nytt objekt av typen *Service Group,* under den nytillverkade servicegruppen tillverkades ett objekt av typen *Service Provider*, servicegruppen kopplades till OPC servern för larm och händelser. Genom anslutning mellan 800xA och OPC servern för larm och händelser kan OPC servern leverera larm till 800xA systemet.

När ett larm blir aktivt i PLC:n skickas detta med hjälp av funktionsblocket AlarmCond till 800xA. För att göra de som arbetar med systemet uppmärksamma på att ett larm aktiverats skapas en specifik larmlista. För att konfigurera larmlistan specifikt för projektet gjordes en egen mall under biblioteksstrukturen. Här kopierades en befintlig mall som sedan specificerades för projektet. Kopiering av en befintlig mall gjordes för att lättare förstå hur mallen fungerade. Om en befintlig mall ändras på kan den konfiguration som gjorts förstöras vid uppdatering av systemet. Larmlistan konfigurerades med specifikationerna att endast processlarm skall visas, aktiva larm skall visas i larmlistan även om de är kvitterade, om larmet ej är aktivt skall larmet ligga kvar till dess att det kvitterats, detta för att få en överblickbar larmlista.

I larmlistan skall larmen presenteras med en färg beroende på hur alvarligt larmet är. Genom att skapa en ny mall för färgdefinition i biblioteksstrukturen valdes de färger som skulle användas vid presentation av larm från PLC. Här valdes färgprioritering så att de allvarligaste larmen visas med röd markering, lägre prioriteringsordning med orange och ännu lägre med gul markering. De larm som inte har en prioritering kommer att visas med blå markering se figur 6.11.



Figur 6.11 Färger för olika prioriteringsnivå.

För att presentera de larm som konfigurerats läggs en ny mall till i funktionsstrukturen, här valdes att lägga larmlistan under huvudprojektet eftersom larm för båda hissarna skall visas i larmlistan. I mallens konfigurerings vy väljs det bibliotek som skapades för presentation av larmen.

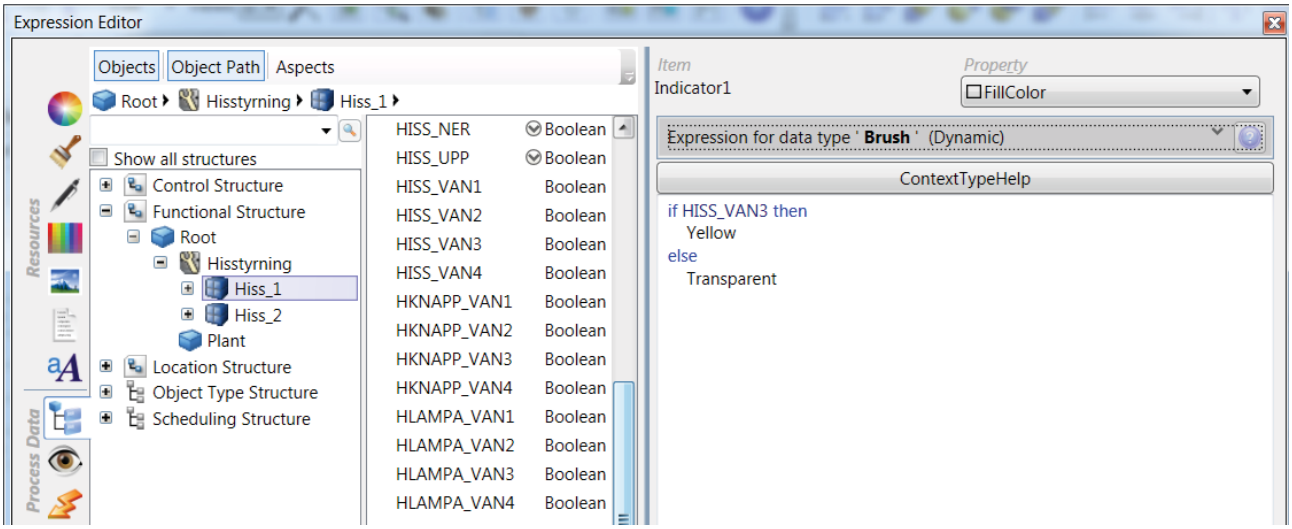
6.3.4 Graphic builder

Då konstruktion av operatörsbilder gjordes användes Graphic Builder som är ett program i 800xA. För att kunna använda programmet görs en mall av typen *Graphic Display PG2* under den del av funktionsstrukturen som bilden ska presentera. Genom att välja editera mallen öppnas programmet Graphic Builder. I programmet finns förprogrammerade figurer att använda, exempelvis rör, tankar och olika sorters knappar.

6.3.4.1 Beskrivning av funktionerna.

När en ny bild tillverkas bestäms först storleken på bilden, antalet pixlar på höjd och bredd bestämdes till 900*1200 då detta gav ett bra utseende på skärmen.

Indikeringar konfigurerades genom att lägga till en indikeringssymbol, efter det skrevs en if sats för att få indikeringen att byta färg när givaren aktiverades figur 6.12. Då 800xA är ihopkopplat med PLC syns alla variabler i en trädstruktur vid konfigurering och det var lätt att klicka ner i strukturen för att välja de variabler som skulle användas.



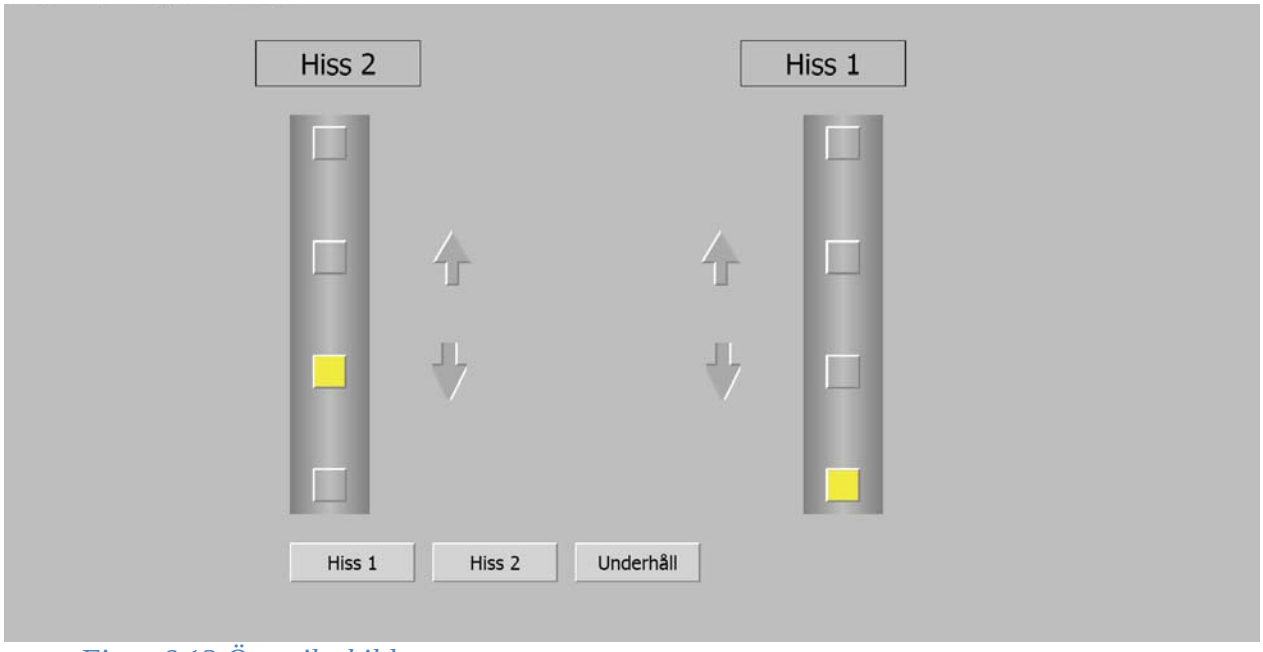
Figur 6.12 trädstruktur och if sats.

Knappar av typen *aspect view button* valdes ur listan av grafiska symboler och konfigurerades för att kunna byta mellan bilderna. Knappen kopplades ihop med den mall där bilden som skulle bytas till var konfigurerad. Vid konfigurering av knappen valdes att vid tryck på knappen skall den bild som var kopplad till knappen ersätta den bild som precis visats.

Tryckknappar för att påverka PLC:n valdes ur listan för grafiska symboler och kopplades till variabeln som styrde den önskade funktionen. När knappen trycks ner sätts en etta på den bestämda variabeln i PLC:n och när knappen släpps upp nollställs signalen i PLC:n.

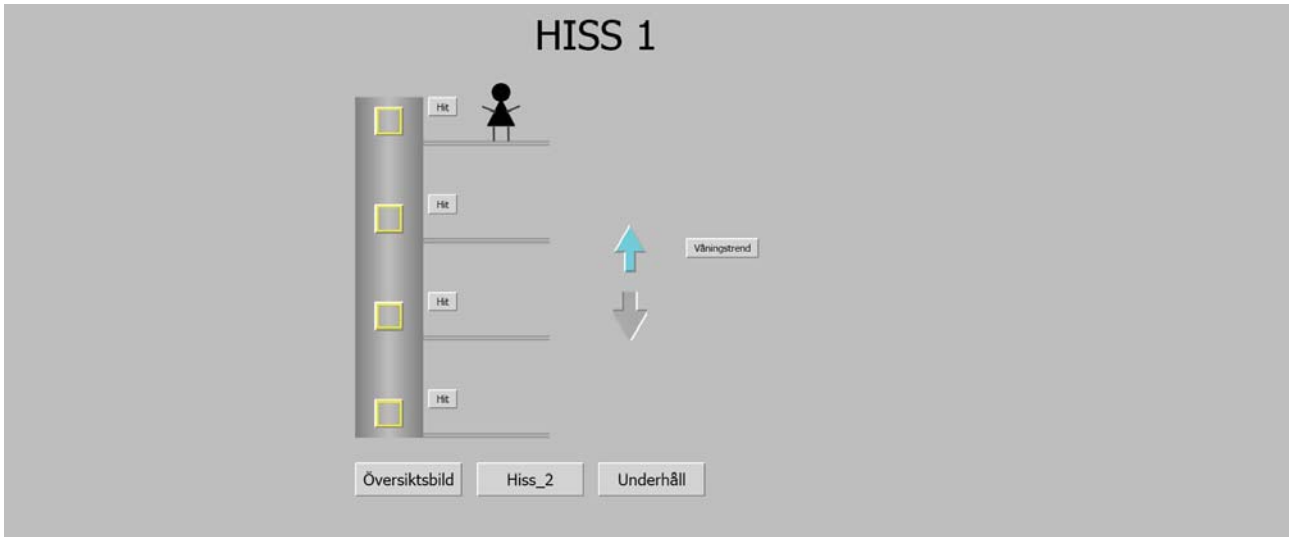
6.3.4.2 Beskrivning av bilderna.

Till att börja med konfigurerades en översiktsbild över de två hissarna se figur 6.13. Här visas de två hissarna, indikering på vart hissen befinner sig och pilen symboliserar vilken riktning hissen kör åt. Knappar för att byta mellan bilderna är också konfigurerade.



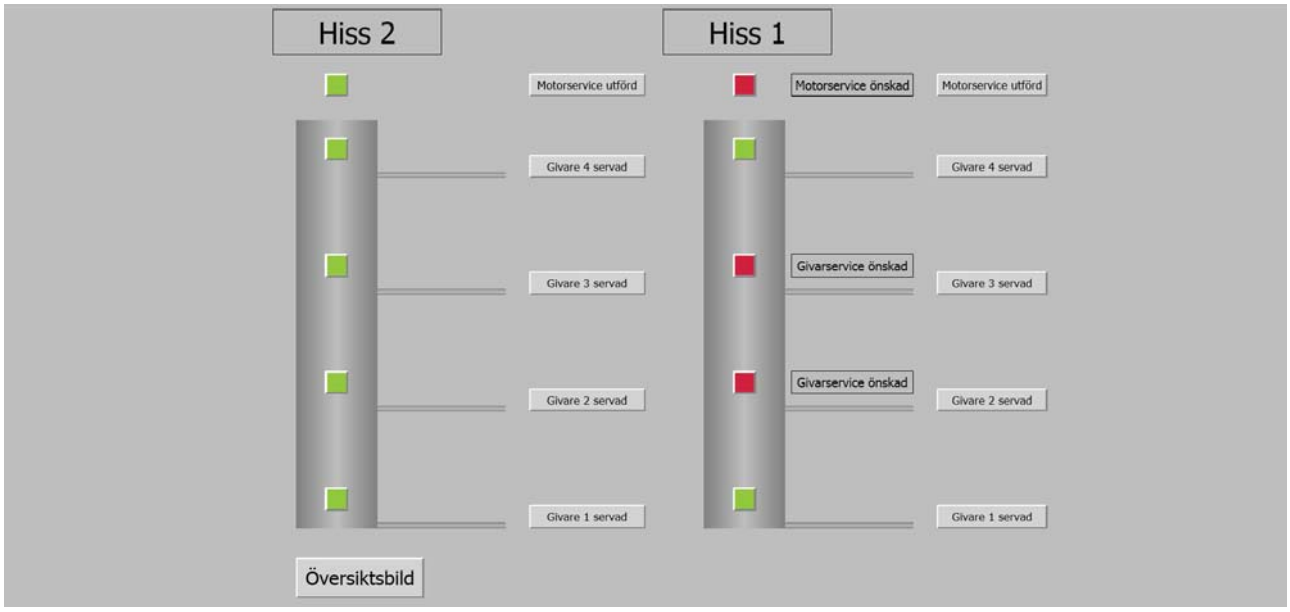
Figur 6.13 Översiktsbild

Efter det gjordes en operatörsbild för varje hiss. I operatörsbilden visas var hissen befinner sig, vilka våningar som är beställda symboliseras av gubben. I dessa bilder kan operatören även beställa vilken våning hissen skall köra till nästa gång se figur 6.14.



Figur 6.14 Operatörsbild Hiss 1

För att kunna återställa underhållslarmen i PLC:n tillverkades en underhållsbild. Underhållsbilden är uppbyggd så att alla underhållslarm har en indikering på om larm har aktiverats eller inte grön indikering visar på oaktiverat larm och röd indikering visar aktiverat larm. Till varje underhållslarm har även en återställningsknapp konfigurerats se figur 6.15.



Figur 6.15 Underhållsbild

6.3.5 Historian and trend

För att kunna se tillbaka i tiden på vad som skett med processen konfigurerades historikmodulen. Historikmodulen är uppbyggd på loggar av två olika sorter så kallade direkta och hierarkiska loggar. Den direkta loggen samlar in mätvärden med ett specifikt intervall medan den hierarkiska loggen plockar värden från den direkta loggen och kan sedan presentera sitt resultat som exempelvis ett medelvärde eller maxvärde över tid.

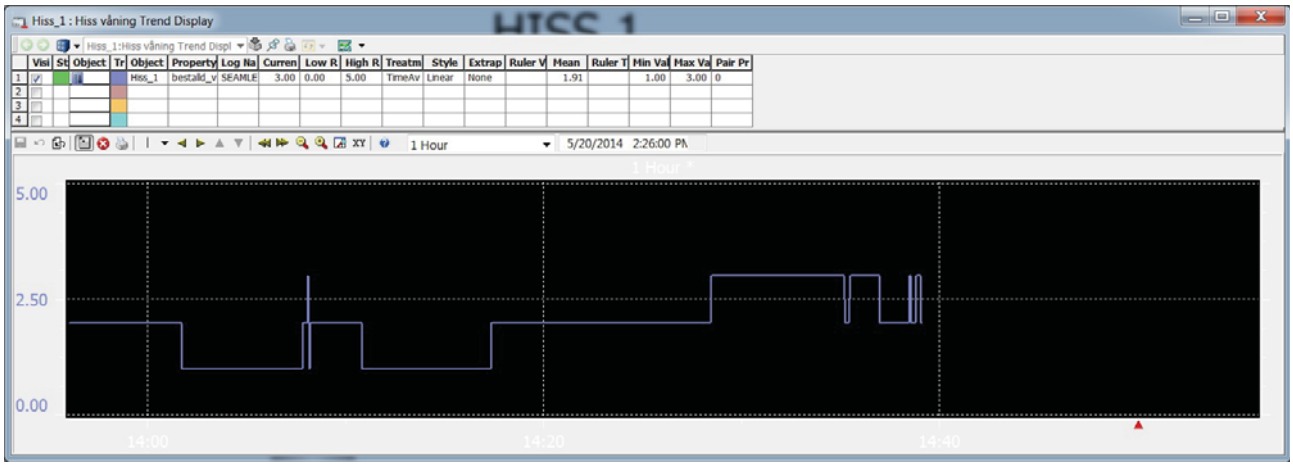
I aspekten *log template* konfigurerades först en direkt logg som konfigurerades att ta ett sampel två gånger varje sekund och sedan spara datan i två veckor. För att kunna spara data en längre tid utan att kräva allt för mycket minne av datorn ökades samplingstiden till en sekund och data sparas då i 15 veckor. När minnesbufferten blir full skrivs det första sparade värdet över så att datan i loggen alltid är de senast sparade samplen.

Under de två loggarna som sparar direkt data tillverkades sedan hierarkiska loggar. En hierarkisk logg för den direkta loggen som sparar värden två gånger i sekunden tillverkades, denna sparar medelvärdet var femte sekund och sparar det i fyra veckor. För den direkta loggen som sparar värden varje sekund tillverkades en hierarkisk logg som tar ett värde var tionde sekund och summerar värdena. De hierarkiska loggarna är väldigt bra om en analog process ska loggas då medelvärden, max eller min värden kan välja att presenteras, i detta projekt finns inga analoga signaler och därför ger de hierarkiska loggarna inte detta projekt någon direkt vinst.

När konfiguration skett med avseende på hur ofta värden ska loggas bestäms även vilka värden som ska hämtas. I kontrollstrukturen valdes den applikation där värden skulle hämtas, sedan skapades en ny mall av typen *log configuration.* I mallen lades sedan de värden som ska loggas till. Om variablerna som skall loggas ligger på olika ställen i hierarkin tillverkas en ny mall på varje ställe där variabler ska loggas. De värden som loggas i projektet är om motorn går uppåt eller neråt, vilken våning hissen befinner sig på, och våningslarmen. Våningslarmen loggas för att kunna se hur länge larmet legat aktivt innan åtgärd skett.

För att presentera de variabler som loggats görs en mall av typen *trend display*. Under konfigurationsmenyn för trenddisplayen valdes en grå visningsvy där de variabler som valts för visning presenteras ovanför visningsbilden. Mellan vilket intervall värdena skall visas bestäms i

raden för varje variabel. En trenddisplay för hur hissen hade rört sig tillverkades och variabeln *bestalld_vaning* lades till för visning i fönstret. Lägsta visningsvärde konfigurerades till noll och högsta till fem för att få en bra översikt se figur 6.16



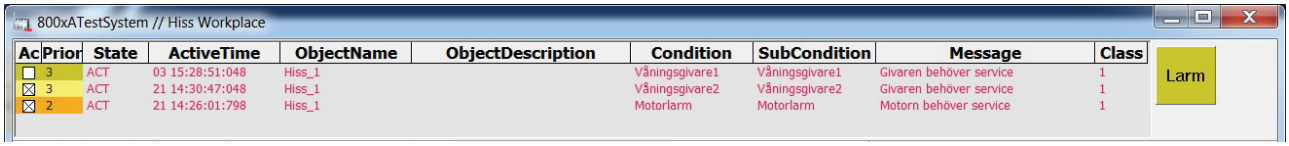
Figur 6.16 Trend över hissrörelsen för hiss 1

Då detta inte är en analog process så är ett medelvärde ingenting som önskas utan meningen med att kunna se historik är för att kunna se hur ofta hissen går till en viss våning. Detta gör att om sampel tas för sällan kan vissa våningar missas och då fås ingen tillförlitlig statistik.
Om det istället hade varit en temperatur som hade loggats hade ett medelvärde under en längre tid eller en längre samplingstid varit mer givande då temperaturer inte ändras så snabbt.

6.3.6 Workplace

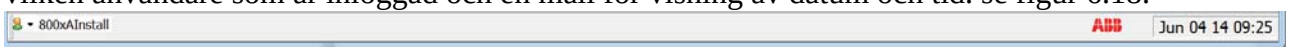
Ett workplace är den bild som skall visas för operatören vid körning av processen. Det finns ett antal förinställda arbetsstationer vid installation av 800xA men för att få den konfigurering som önskades gjordes en ny arbetsstation till projektet. I workplacestrukturen skapades ett nytt objekt av typen *plant workplace*. I mallen workplace layout konfigurerades sedan hur arbetsstationen skulle vara uppbyggd, här valdes att visa arbetsstationen på en skärm för att det är vad som erbjuds i projektet, 800xA har dock stöd för att visa 4 skärmar samtidigt. I layout mallen bestämdes även att en applikationsrad skulle visas överst i fönstret och en statusrad visas längst ner i fönstret.

I applikationsfönstret högst upp på skärmen implementerades en larmlista och en knapp för att kunna öppna larmlistan och titta på den i större skala se figur 6.17. Genom att välja mallen för applikationsfönstret kunde höjden på fönstret bestämmas, sedan lades mallen för larmlistan och knappen till. Hur stor del av applikationsfönstret varje mall skulle visas på bestämdes i procent och här gavs larmlistan 90% och knappen 10% detta för att kunna se alla fält i larmlistan utan att behöva scrolla åt sidan.



Figur 6.17 Applikationsfönster med larmlista

I nederkant av bilden konfigurerades en statusrad här lades mall till för visning av ABB logotypen, vilken användare som är inloggad och en mall för visning av datum och tid. se figur 6.18.



Figur 6.18 statusrad i nederkant med tid logga och användarinformation.

I workplace strukturen konfigurerades även vilken av operatörsbilderna som skulle vara startup bild. Detta gjordes genom att lägga in mallen för underhållsbilden under arbetsstationen för hissen och döpa mallen till *startup display*. För att operatören skall slippa öppna många olika program vid start av datorn valdes att hissarbetsstationen skall starta automatiskt vid start av datorn. Om flera datorer är installerade i SCADA-systemet med 800xA kan olika arbetsstationer konfigureras, exempelvis en för varje dator. Dessa kan sedan ha olika startbilder. Varje dator konfigureras sedan så att den specifika arbetsstationen startas automatiskt när datorn startas. Detta gör att olika stationer kan konfigureras med olika startbilder.

7 RESULTAT

För att kunna styra de två hissarna har två PLC-er programmerats. Programmen för hisstyrningen fungerar bra och hissarna fungerar enligt de krav som sattes upp i kravspecifikationen.

Processlarm är programmerat ifall hissen stannar mellan två våningar och underhållslarm är programmerade för att service skall kunna utföras på givare och motor innan dessa går sönder. En larmlista har konfigurerats för att operatören skall få en bra överblick av vilka larm som är aktiva.

För att kunna se hur hissen har rört sig över tid har historik konfigurerats och en trendbild tillverkats.

En övervakande operatörsbild har utformats för att kunna ha koll på båda processerna samtidigt. Till detta har en processbild för varje hiss konfigurerats där mer information kan utläsas, exempelvis vilka våningar som är beställda för den specifika hissen. Från processbilden kan hissen styras och trenbild visas. För att underhållslarm skall kunna kvitteras vid utfört underhåll har en bild med underhållslarmen och kvittensknappar skapats.

I arbetsstationen har larmlistan lagts in tillsammans med operatörsbilderna för att ge operatören en bra arbetsplats.

Genom att utföra dessa delar har ett fungerande styr och övervakningssystem utformats, förståelse över de ingående delarna i ett SCADA system har skapats och genom att upprätta OPC kommunikation mellan systemen har förståelse för detta också skapats.

7.1 Rekommendationer till fortsatt arbete

För att utbildning av nya medarbetare i ABB:s programvara 800xA skall gå så fort och smidigt som möjligt borde en manual utformas där grundläggande beskrivning ges om hur ett SCADA system sätts upp och utformas i ABB:s utvecklingsmiljö.

Utveckling av PLC programmet borde möjligtvis göras för att kunna köra hissarna tillsammans, ett smart hissystem där närmaste hiss går till den beställda våningen hade kunnat utvecklas.

För att få SCADA systemet mer intressant hade det varit roligt med analoga signaler. Exempelvis en nivåtank som skulle kunna regleras med ett antal ventiler. Eftersom 800xA systemet är utvecklat för processindustri hade detta gjort att fler delar av systemet hade kunnat utnyttjas.

8 DISKUSSION

Genom att sätta upp och konfigurera ett SCADA system har jag fått en god förståelse för hur och varför ett SCADA system sätts upp och vilka konfigurationer som behöver göras. Genom OPC kommunikation har nätverket satts upp. Detta har gått väldigt smidigt då både OPC servern, PLC:n och 800xA programvaran har varit från ABB. Ifall ett system från en annan leverantör också skulle ha konfigurerats hade omständigheterna runt nätverkskonfigurationen blivit lite mer arbetskrävande.

Programmeringen av hisstyrningen tycktes vid start av projektet vara ganska enkelt då detta hade utförts i tidigare projekt. Detta blev inte fallet då programvaran Control Builder var väldigt annorlunda mot tidigare programvara som jag använt för PLC programmering och tog således ganska lång tid att sätta sig in i och lära sig.

Vid programmering av bilderna i Graphics Builder märktes tydligt att 800xA är ett system för processtyrning. Det finns många bra färdiga bilder för tankar, rör och ventiler.

800xA systemet har även det varit lite krångligt att sätta sig in i. Jag hittar fortfarande inte riktigt bland menyerna och ser ingen riktig logik i hur det är uppbyggt. Jag tror att det behövs mycket längre tid för att verkligen förstå och bli bra på programmering av 800xA system. Utbildningsmaterialet som fanns på ÅF har hjälpt till mycket vid konfiguration av de olika delarna i systemet.

9 Referenser

1. I. Wiklund, et al., *Underlag för Styrprojekt,* 2012

2. *Overview of the IEC 61131 Standard*, ABB, 2008-11-19 http://www05.abb.com/global/scot/scot267.nsf/veritydisplay/95c0dd2588fe299b852575060077296b/$file/2101127.pdf (Acc 2014-05-26)
3. Stewart McCrady "Understanding the Elements of SCADA Software," i *Designing SCADA Application software: A Practical Approach,* London: Elsevier, 2013
4. McIlvride B., Thomas A *OPC Tunneling – Know your options,* Cogent Real-Time Systems, 2008 http://www.opcdatahub.com/Download/PDF_Release/OPC_Tunnelling_Options.pdf (Acc 2014-06-02)
5. M. Damm, et al., "Introduction," i *OPC Unified Architecture*, Berlin: Springer, 2009
6. Rekter et al.,*Address Allocation for Private Internets*, February 1996 http://tools.ietf.org/html/rfc1918 (Acc 2014-06-02)
7. *Användarmanualer för Abb 800xA,* ABB http://www.abb.se/product/ap/seitp334/19a268505c70cb45c1257a4f00423cc3.aspx (Acc 2014-06-03)

Appendix A
PLC kod

Hiss_1 – Global Variables

	Name	Data Type	Attributes	Initial value	I/O address	Access Variables
1	HISS_NER	bool			Controller_1.0.11.2.1	
2	HISS_UPP	bool			Controller_1.0.11.2.2	
3	HISS_VAN1	bool	retain		Controller_1.0.11.1.5	
4	HISS_VAN2	bool	retain		Controller_1.0.11.1.6	
5	HISS_VAN3	bool	retain		Controller_1.0.11.1.7	
6	HISS_VAN4	bool	retain		Controller_1.0.11.1.8	
7	HKNAPP_VAN1	bool	retain		Controller_1.0.11.1.9	
8	HKNAPP_VAN2	bool	retain		Controller_1.0.11.1.10	
9	HKNAPP_VAN3	bool	retain		Controller_1.0.11.1.11	
10	HKNAPP_VAN4	bool	retain		Controller_1.0.11.1.12	
11	HLAMPA_VAN1	bool			Controller_1.0.11.2.7	
12	HLAMPA_VAN2	bool			Controller_1.0.11.2.8	
13	HLAMPA_VAN3	bool			Controller_1.0.11.2.9	
14	HLAMPA_VAN4	bool			Controller_1.0.11.2.10	
15	KNAPP_VAN1	bool	retain		Controller_1.0.11.1.1	
16	KNAPP_VAN2	bool	retain		Controller_1.0.11.1.2	
17	KNAPP_VAN3	bool	retain		Controller_1.0.11.1.3	
18	KNAPP_VAN4	bool	retain		Controller_1.0.11.1.4	
19	LAMPA_VAN1	bool			Controller_1.0.11.2.3	
20	LAMPA_VAN2	bool			Controller_1.0.11.2.4	
21	LAMPA_VAN3	bool			Controller_1.0.11.2.5	
22	LAMPA_VAN4	bool			Controller_1.0.11.2.6	
23	Mvan1	bool				
24	Mvan2	bool				
25	Mvan3	bool				
26	Mvan4	bool				
27	aterstallning_van4	bool	retain			Controller_1.acc_aterstallning_van4(MMS)
28	aterstallning_van3	bool	retain			
29	aterstallning_van2	bool	retain			
30	aterstallning_van1	bool	retain			
31	aterstallning_x	bool	retain			
32	Citect_Test_Tag	bool	retain			

Table 1. Application - Hiss_1 (Global variables)

Property	Read permission	Write permission	Authentication level
HISS_NER			None
HISS_UPP			None
HISS_VAN1			None
HISS_VAN2			None
HISS_VAN3			None
HISS_VAN4			None
HKNAPP_VAN1			None
HKNAPP_VAN2			None
HKNAPP_VAN3			None
HKNAPP_VAN4			None
HLAMPA_VAN1			None
HLAMPA_VAN2			None
HLAMPA_VAN3			None
HLAMPA_VAN4			None
KNAPP_VAN1			None
KNAPP_VAN2			None
KNAPP_VAN3			None
KNAPP_VAN4			None
LAMPA_VAN1			None
LAMPA_VAN2			None
LAMPA_VAN3			None
LAMPA_VAN4			None
Mvan1			None
Mvan2			None
Mvan3			None
Mvan4			None
aterstallning_van4			None
aterstallning_van3			None
aterstallning_van2			None

Property	Read permission	Write permission	Authentication level
aterstallning_van1			None
aterstallning_x			None
Citect_Test_Tag			None

Table 2. Application - Hiss_1 (User Permission)

Hiss_1

	Name	Data Type	Attributes	Initial value	I/O address	Access Variables	Description
1	Riktning	bool	retain				
2	forrariktning	bool	retain				
3	rattlage	bool	retain				
4	klar	bool	retain				
5	aktuell_vaning	dint	retain				
6	bestalld_vaning	dint	retain				
7	nuvarande_bestallning	dint	retain				
8	gangtid_x	time					
9	Larmtid_givare	time	retain	t#60s			
10	Larmtid_motor	time	retain	t#10s			
11	motorlarm	bool					
12	vaningslarm_1	bool					
13	vaningslarm_2	bool					
14	vaningslarm_3	bool					
15	vaningslarm_4	bool					
16	tid	time	retain	T#3s			
17	timerklar	bool	retain				
18	starttimer	bool	retain				
19	value	time	retain				
20	Vaningslarm1SrcName	string		'Hiss_1'			
21	Vaningslarm1CondName	string		'Våningsgivare1'			
22	gangtid_2	time	retain				
23	gangtid_3	time	retain				
24	gangtid_van4	time	retain				
25	Vaningslarm2SrcName	string	retain	'Givarfel'			
26	Vaningslarm2CondName	string	retain	'Givarfel'			
27	Vaningslarm3SrcName	string	retain	'Våningsgivare3'			
28	Vaningslarm3CondName	string	retain	'Våningsgivare3'			
29	Vaningslarm4SrcName	string	retain	'Våningsgivare4'			
30	Vaningslarm4CondName	string	retain	'Våningsgivare4'			
31	INGEN_VANING	bool	retain				
32	VaningslarmSrcName	string	retain	'Hiss_1'			
33	VaningslarmMessage	string	retain	'hissen hittar ingen våning'			
34	VaningslarmCondName	string	retain	'Våningslarm'			
35	Motorlarm_1SrcName	string	retain	'Hiss_1'			
36	Motorlarm_1CondName	string	retain	'Motorlarm'			
37	Motorlarm_1Message	string	retain	'Motorn behöver service'			
38	count	dint	retain	5			
39	Kommunikation	bool	retain				
40	Vaningslarm1Message	string		'Givaren behöver service'			
41	Vaningslarm2Message	string	retain	'Givaren behöver service'			
42	Vaningslarm3Message	string	retain	'Givaren behöver service'			
43	Vaningslarm4Message	string	retain	'Givaren behöver service'			

Table 3. Program - Hiss_1.Hiss_1 (Variables)

	Name	Function Block Type	Task Connection	Description
1	tand_slack_lampa_1	tand_slack_lampa		
2	tand_slack_lampa_2	tand_slack_lampa		
3	tand_slack_lampa_3	tand_slack_lampa		
4	tand_slack_lampa_4	tand_slack_lampa		
5	TimerU_motor	TimerU		
6	TOn_1	TOn		
7	Givarlarm_van1	AlarmCond		
8	Givarlarm_van2	AlarmCond		
9	Givarlarm_van3	AlarmCond		
10	Givarlarm_van4	AlarmCond		
11	vaningslarm	AlarmCond		
12	Motorlarm_1	AlarmCond		
13	TOn_2	TOn		
14	CTU_1	CTU		
15	CTU_2	CTU		
16	CTU_3	CTU		
17	CTU_4	CTU		

	Name	Function Block Type	Task Connection	Description
18	TOn_3	TOn		

Table 4. Program - Hiss_1.Hiss_1 (Function blocks)

Property	Read permission	Write permission	Authentication level
Riktning			None
forrariktning			None
rattlage			None
klar			None
aktuell_vaning			None
bestalld_vaning			None
nuvarande_bestallning			None
gangtid_x			None
Larmtid_givare			None
Larmtid_motor			None
motorlarm			None
vaningslarm_1			None
vaningslarm_2			None
vaningslarm_3			None
vaningslarm_4			None
tid			None
timerklar			None
starttimer			None
value			None
Vaningslarm1SrcName			None
Vaningslarm1CondName			None
gangtid_2			None
gangtid_3			None
gangtid_van4			None
Vaningslarm2SrcName			None
Vaningslarm2CondName			None
Vaningslarm3SrcName			None
Vaningslarm3CondName			None
Vaningslarm4SrcName			None
Vaningslarm4CondName			None
INGEN_VANING			None
VaningslarmSrcName			None
VaningslarmMessage			None
VaningslarmCondName			None
Motorlarm_1SrcName			None
Motorlarm_1CondName			None
Motorlarm_1Message			None
count			None
Kommunikation			None
Vaningslarm1Message			None
Vaningslarm2Message			None
Vaningslarm3Message			None
Vaningslarm4Message			None

Table 5. Program - Hiss_1.Hiss_1 (User Permission)

Lampor

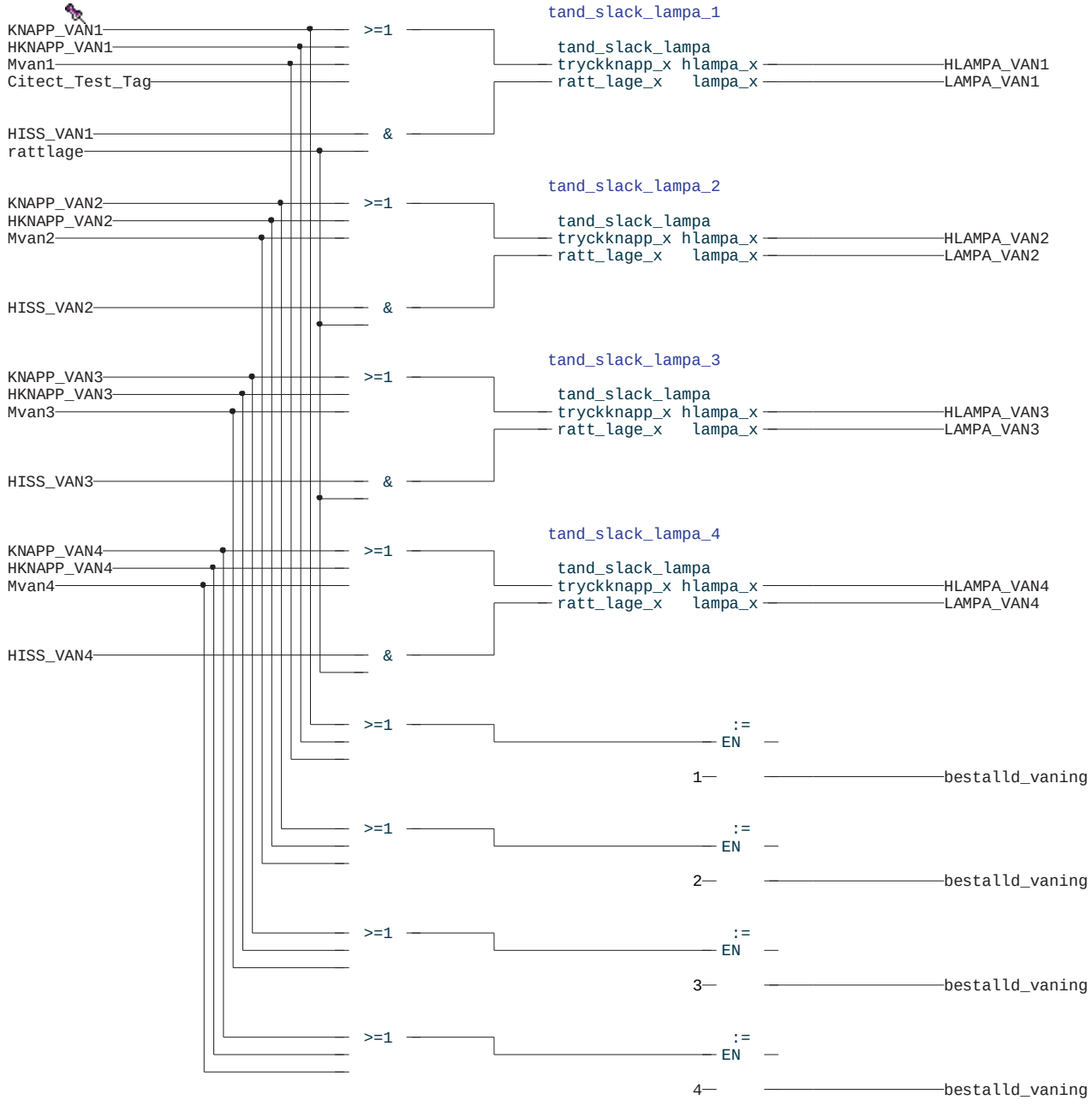


Figure 1. Lampor - Program - Hiss_1.Hiss_1

Rätt läge

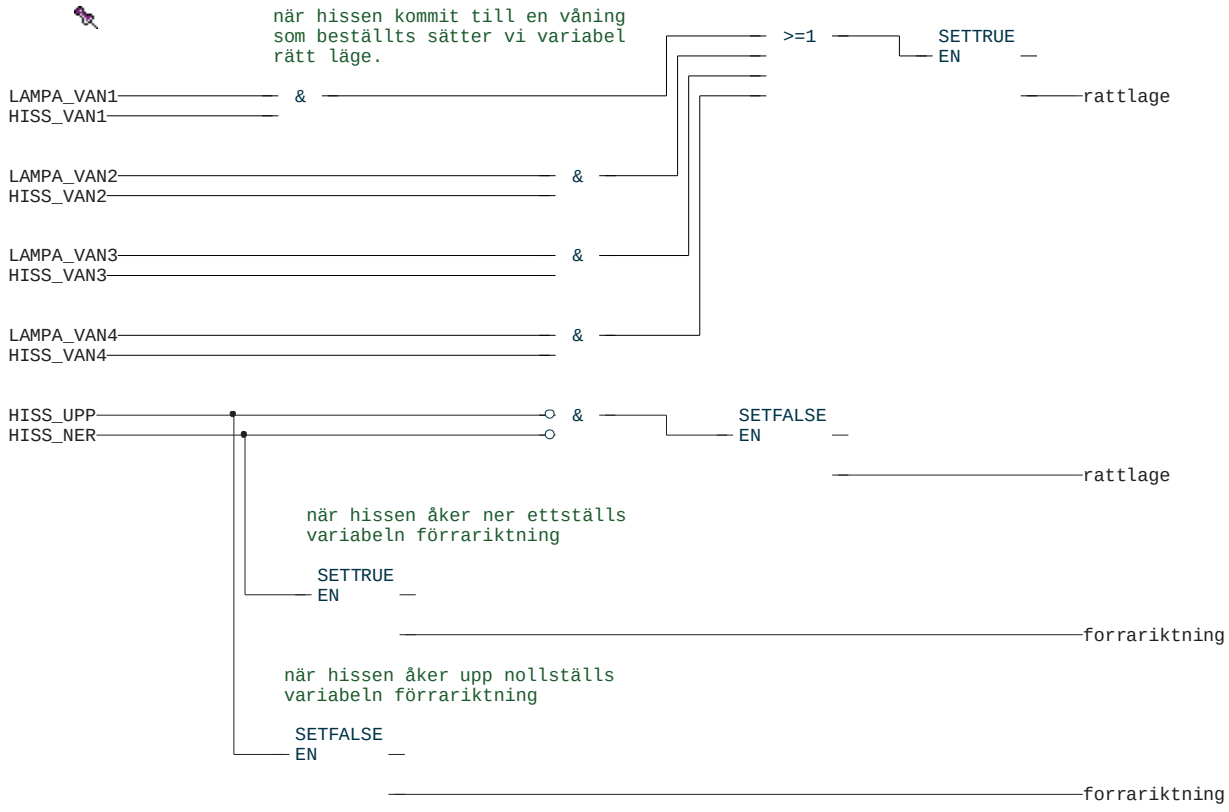


Figure 2. rättläge - Program - Hiss_1.Hiss_1

Riktning

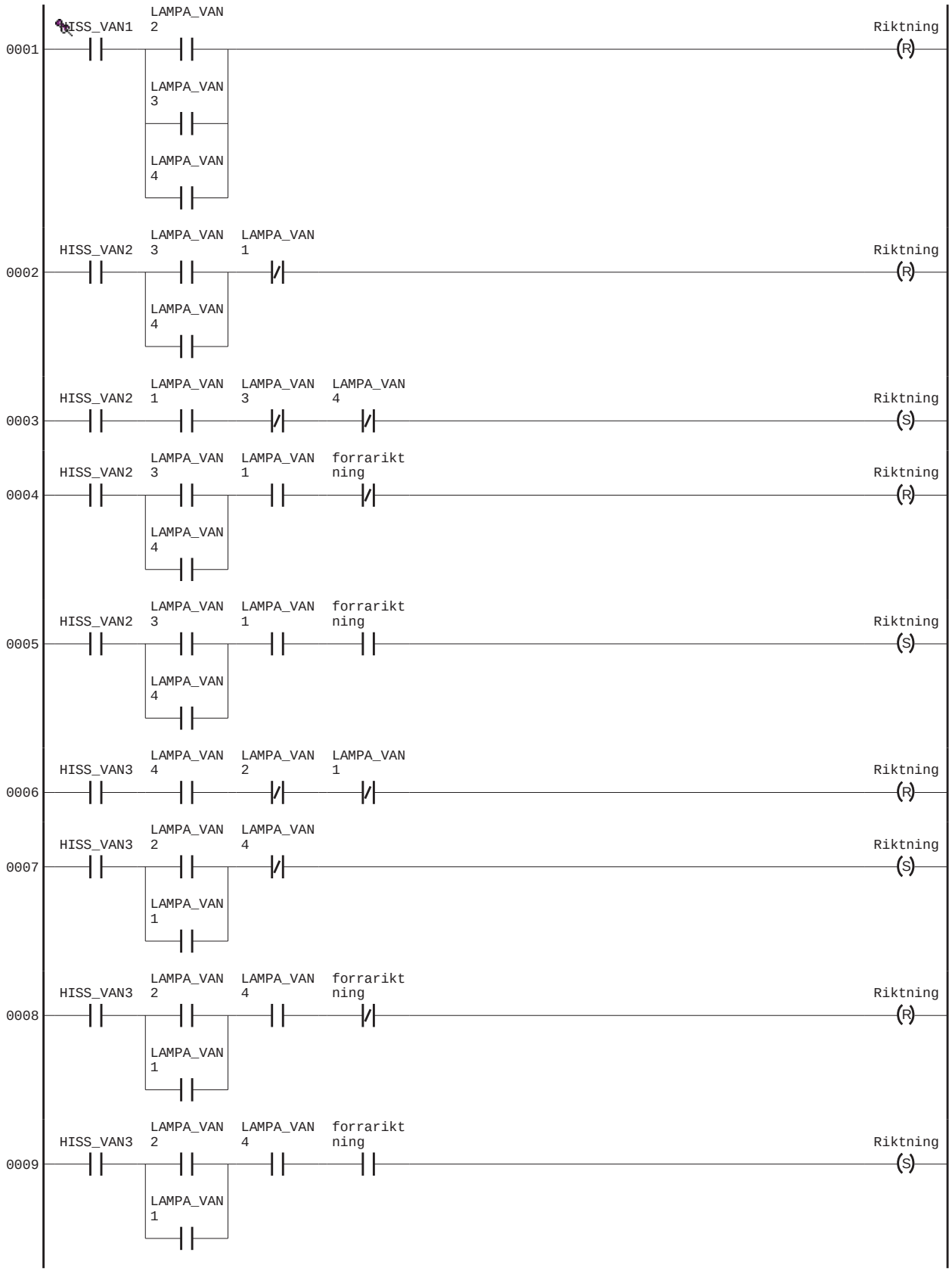


Figure 3. riktning - Program - Hiss_1.Hiss_1

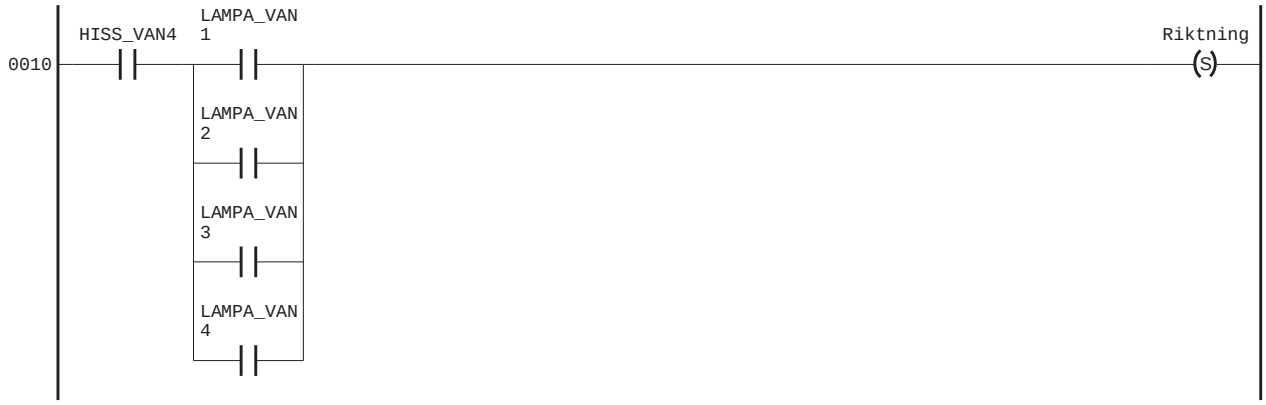


Figure 4. riktning - Program - Hiss_1.Hiss_1

Hiss

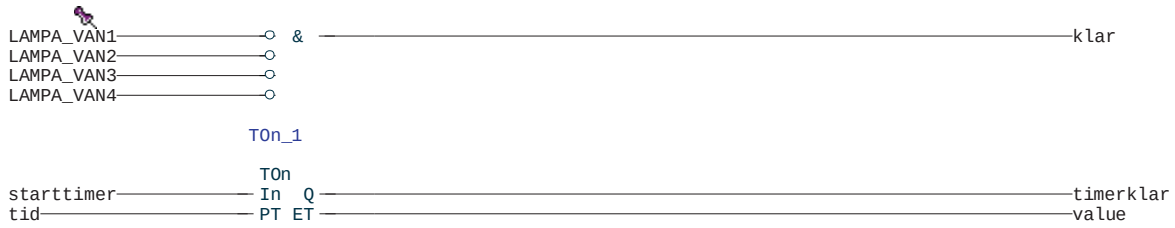


Figure 5. hiss - Program - Hiss_1.Hiss_1

Sätta_larm

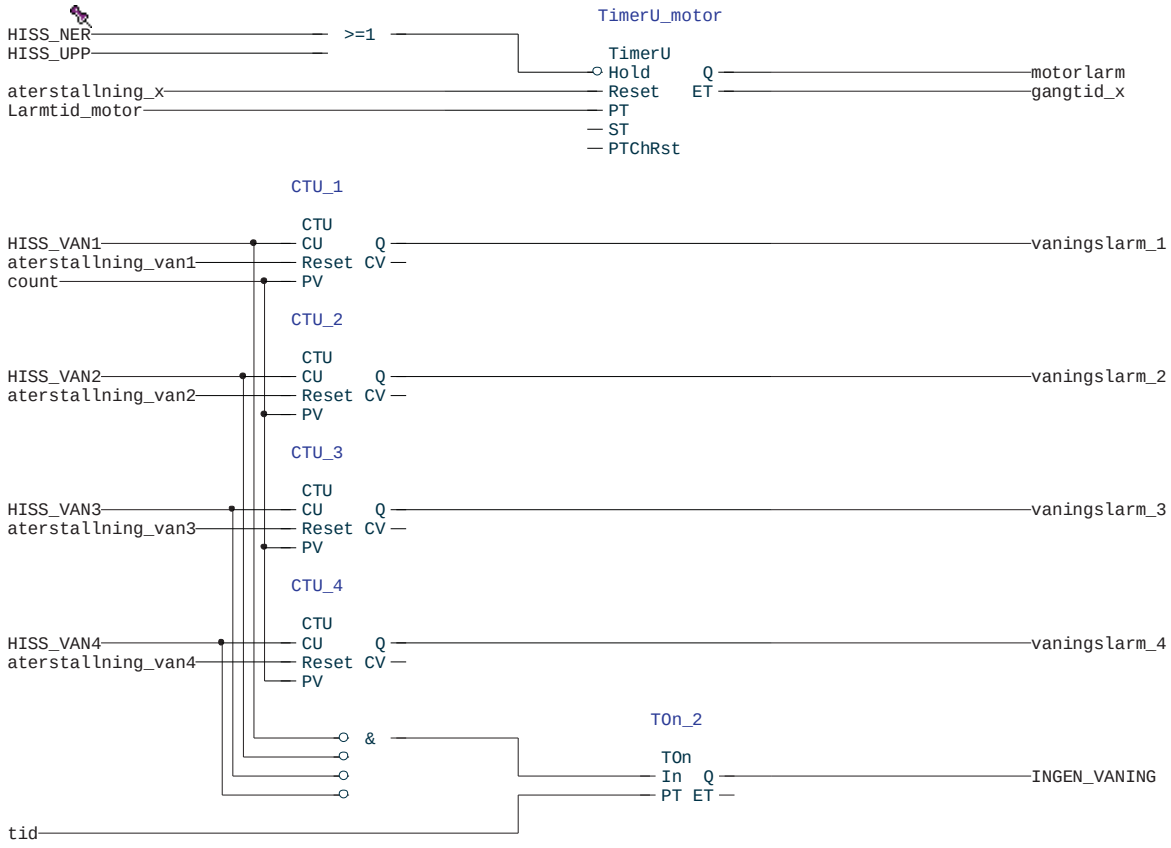


Figure 6. Sätta_larm - Program - Hiss_1.Hiss_1

Larm_till_800xA

```
givarlarm_van1( Signal := vaningslarm_1,
        SrcName := Vaningslarm1SrcName,
        CondName := Vaningslarm1CondName,
        Message := Vaningslarm1Message );

givarlarm_van2( Signal := vaningslarm_2,
         SrcName := Vaningslarm2SrcName,
         CondName := Vaningslarm2CondName,
         Message := Vaningslarm2Message  );

givarlarm_van3( Signal := vaningslarm_3,
         SrcName := vaningslarm3SrcName,
         CondName := Vaningslarm3CondName );

givarlarm_van4( Signal := vaningslarm_4,
         SrcName := Vaningslarm4SrcName,
         CondName := Vaningslarm4CondName,
         Message := Vaningslarm4Message );
```

```
vaningslarm( Signal := INGEN_VANING,
        SrcName := VaningslarmSrcName,
        CondName := VaningslarmCondName,
        Message := VaningslarmMessage,
        Severity := 900 );

Motorlarm_1( Signal := motorlarm,
        SrcName := Motorlarm_1SrcName,
        CondName := Motorlarm_1CondName,
        Message := Motorlarm_1Message,
        Severity := 600 );
```

Hiss_sekvens

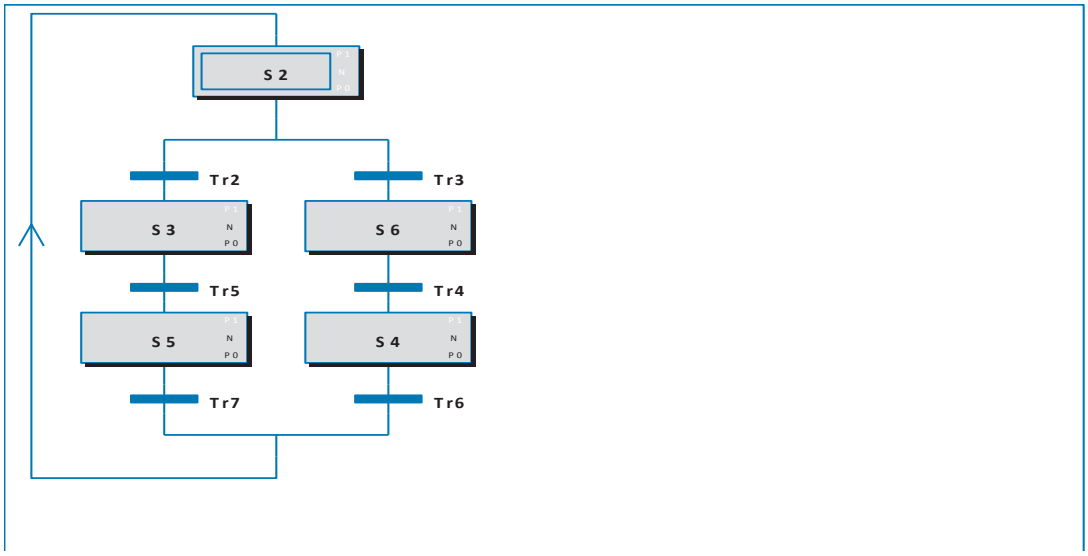


Figure 7. Hiss_sekvens_Program - Hiss_1.Hiss_1

S2

S2_P1

S2_N

S2_P0

S3

S3_P1

S3_N

HISS_UPP := true;

S3_P0

HISS_UPP := false;

S4

S4_P1

S4_N

starttimer := true ;

S4_P0

starttimer := false;

S5

S5_N

starttimer := true;

S5_P0

starttimer := false;

S6

S6_P1

S6_N

HISS_NER := true ;

S6_P0

HISS_NER := false;

Tr2

not riktning and not klar

Tr3

riktning and not klar

Tr4

rattlage

Tr5

rattlage

Tr6

timerklar

Tr7

timerklar

Mellan_våning

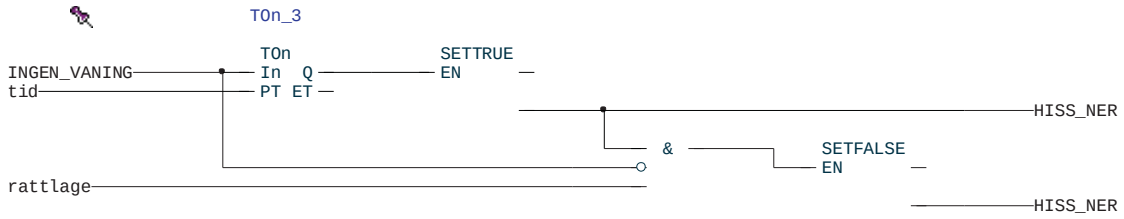


Figure 8. Mellan_våning - Program - Hiss_1.Hiss_1

Hiss_2 – Global variables

	Name	Data Type	Attributes	Initial value	I/O address	Access Variables	Description
1	HISS_NER	bool			Controller_2.0.11.2.1		
2	HISS_UPP	bool			Controller_2.0.11.2.2		
3	HISS_VAN1	bool	retain		Controller_2.0.11.1.5		
4	HISS_VAN2	bool	retain		Controller_2.0.11.1.6		
5	HISS_VAN3	bool	retain		Controller_2.0.11.1.7		
6	HISS_VAN4	bool	retain		Controller_2.0.11.1.8		
7	HKNAPP_VAN1	bool	retain		Controller_2.0.11.1.9		
8	HKNAPP_VAN2	bool	retain		Controller_2.0.11.1.10		
9	HKNAPP_VAN3	bool	retain		Controller_2.0.11.1.11		
10	HKNAPP_VAN4	bool	retain		Controller_2.0.11.1.12		
11	HLAMPA_VAN1	bool			Controller_2.0.11.2.10		
12	HLAMPA_VAN2	bool			Controller_2.0.11.2.9		
13	HLAMPA_VAN3	bool			Controller_2.0.11.2.8		
14	HLAMPA_VAN4	bool			Controller_2.0.11.2.7		
15	KNAPP_VAN1	bool	retain		Controller_2.0.11.1.1		
16	KNAPP_VAN2	bool	retain		Controller_2.0.11.1.2		
17	KNAPP_VAN3	bool	retain		Controller_2.0.11.1.3		
18	KNAPP_VAN4	bool	retain		Controller_2.0.11.1.4		
19	LAMPA_VAN1	bool			Controller_2.0.11.2.6		
20	LAMPA_VAN2	bool			Controller_2.0.11.2.5		
21	LAMPA_VAN3	bool			Controller_2.0.11.2.4		
22	LAMPA_VAN4	bool			Controller_2.0.11.2.3		
23	Mvan1	bool					
24	Mvan2	bool					
25	Mvan3	bool					
26	Mvan4	bool					
27	aterstallning_van1	bool	retain				
28	aterstallning_van2	bool	retain				
29	aterstallning_van3	bool	retain				
30	aterstallning_van4	bool	retain				
31	aterstallning_motor	bool	retain				

Table 6. Application - Hiss_2 (Global variables)

Property	Read permission	Write permission	Authentication level
HISS_NER			None
HISS_UPP			None
HISS_VAN1			None
HISS_VAN2			None
HISS_VAN3			None
HISS_VAN4			None
HKNAPP_VAN1			None
HKNAPP_VAN2			None
HKNAPP_VAN3			None
HKNAPP_VAN4			None
HLAMPA_VAN1			None
HLAMPA_VAN2			None
HLAMPA_VAN3			None
HLAMPA_VAN4			None
KNAPP_VAN1			None
KNAPP_VAN2			None
KNAPP_VAN3			None
KNAPP_VAN4			None
LAMPA_VAN1			None
LAMPA_VAN2			None
LAMPA_VAN3			None
LAMPA_VAN4			None
Mvan1			None
Mvan2			None
Mvan3			None
Mvan4			None
aterstallning_van1			None
aterstallning_van2			None
aterstallning_van3			None
aterstallning_van4			None
aterstallning_motor			None

Table 7. Application - Hiss_2 (User Permission)

Programs
Hiss_2 - (Controller_2.Fast)

	Name	Data Type	Attributes	Initial value	I/O address	Access Variables	Description
1	rattlage	bool	retain				
2	forrariktning	bool	retain				
3	Riktning	bool	retain				
4	klar	bool	retain				
5	timerklar	bool	retain				
6	value	time	retain				
7	starttimer	bool	retain				
8	klartimer	bool	retain				
9	motorlarm1	bool	retain				
10	stopptid	time	retain	T#3s			
11	gangtid_x	time	retain				
12	Larmtid	time	retain	T#10s			
13	vaningslarm_van2	bool	retain				
14	vaningslarm_van1	bool	retain				
15	vaningslarm_van3	bool	retain				
16	vaningslarm_van4	bool	retain				
17	gangtid_motor	time	retain				
18	Larmtid_vaning	time		t#50s			
19	Ingen_vaning	bool	retain				
20	VaningslarmSrcName	string	retain	'Hiss_2'			
21	VaningslarmCondName	string	retain	'Våningslarm'			
22	VaningslarmMessage	string	retain	'Hissen hittar ingen våning'			
23	Givarlarm_van1SrcName	string	retain	'Hiss_2'			
24	Givarlarm_van1CondName	string	retain	'Våningsgivare1'			
25	Givarlarm_van2SrcName	string	retain	'Hiss_2'			
26	Givarlarm_van2CondName	string	retain	'Våningsgivare2'			
27	Givarlarm_van3SrcName	string	retain	'Hiss_2'			
28	Givarlarm_van3CondName	string		'Våningsgivare3'			
29	Givarlarm_van4SrcName	string	retain	'Hiss_2'			
30	Givarlarm_van4CondName	string	retain	'Våningsgivare4'			
31	Motorlarm_2SrcName	string	retain	'Hiss_2'			
32	Motorlarm_2CondName	string	retain	'Motor'			
33	Motorlarm_2Message	string	retain	'Motorn behöver service'			
34	count	dint	retain	5			
35	Givarlarm_van1Message	string	retain	'Givaren behöver service'			
36	Givarlarm_van2Message	string	retain	'Givaren behöver service'			
37	Givarlarm_van3Message	string	retain	'Givaren behöver service'			
38	Givarlarm_van4Message	string	retain	'Givaren behöver service'			

Table 8. Program - Hiss_2.Hiss_2 (Variables)

	Name	Function Block Type	Task Connection	Description
1	tand_slack_lampa_1	tand_slack_lampa		
2	tand_slack_lampa_2	tand_slack_lampa		
3	tand_slack_lampa_3	tand_slack_lampa		
4	tand_slack_lampa_4	tand_slack_lampa		
5	TOn_1	TOn		
6	TimerU_motor	TimerU		
7	Vaningslarm	AlarmCond		
8	Givarlarm_van1	AlarmCond		
9	Givarlarm_van2	AlarmCond		
10	Givarlarm_van3	AlarmCond		
11	Givarlarm_van4	AlarmCond		
12	Motorlarm_2	AlarmCond		
13	TOn_2	TOn		
14	CTU_1	CTU		
15	CTU_2	CTU		
16	CTU_3	CTU		
17	CTU_4	CTU		
18	TOn_3	TOn		

Table 9. Program - Hiss_2.Hiss_2 (Function blocks)

Property	Read permission	Write permission	Authentication level

Property	Read permission	Write permission	Authentication level
rattlage			None
forrariktning			None
Riktning			None
klar			None
timerklar			None
value			None
starttimer			None
klartimer			None
motorlarm1			None
stopptid			None
gangtid_x			None
Larmtid			None
vaningslarm_van2			None
vaningslarm_van1			None
vaningslarm_van3			None
vaningslarm_van4			None
gangtid_motor			None
Larmtid_vaning			None
Ingen_vaning			None
VaningslarmSrcName			None
VaningslarmCondName			None
VaningslarmMessage			None
Givarlarm_van1SrcName			None
Givarlarm_van1CondName			None
Givarlarm_van2SrcName			None
Givarlarm_van2CondName			None
Givarlarm_van3SrcName			None
Givarlarm_van3CondName			None
Givarlarm_van4SrcName			None
Givarlarm_van4CondName			None
Motorlarm_2SrcName			None
Motorlarm_2CondName			None
Motorlarm_2Message			None
count			None
Givarlarm_van1Message			None
Givarlarm_van2Message			None
Givarlarm_van3Message			None
Givarlarm_van4Message			None

Table 10. Program - Hiss_2.Hiss_2 (User Permission)

Rätt läge

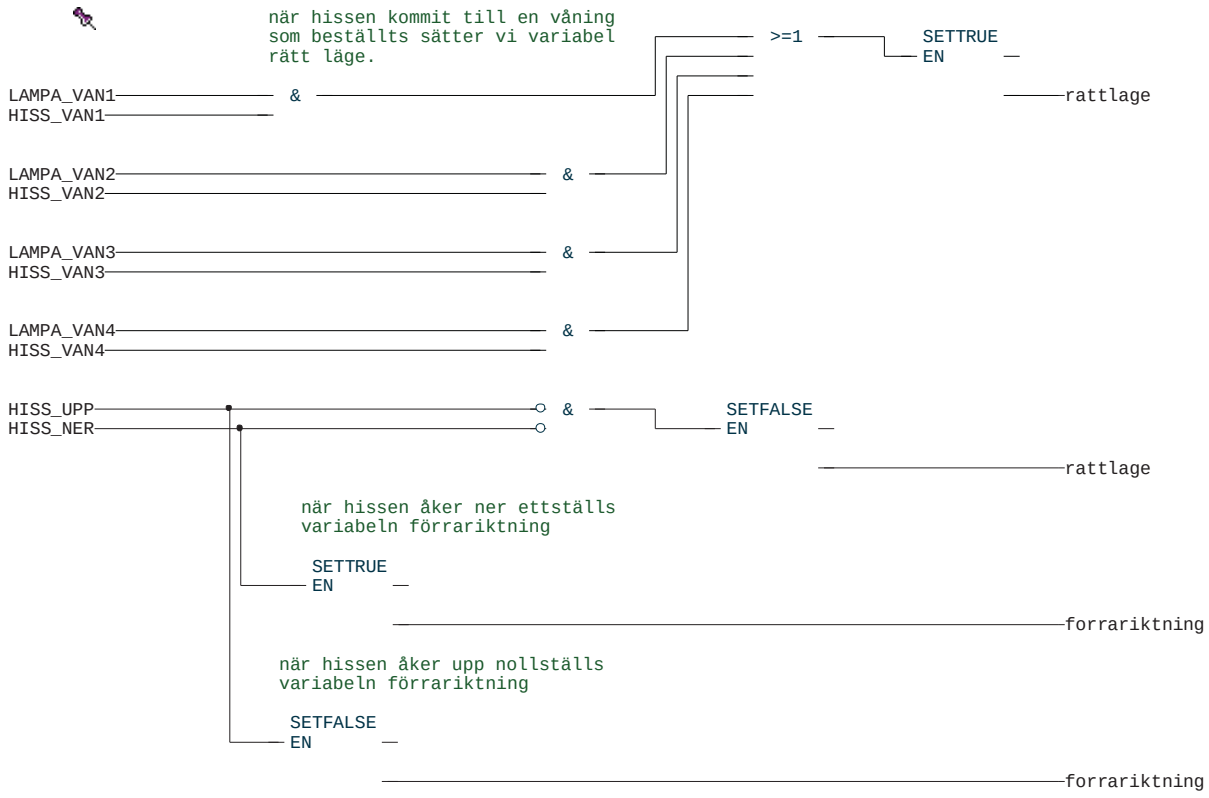


Figure 9. rättläge - Program - Hiss_2.Hiss_2

Lampor

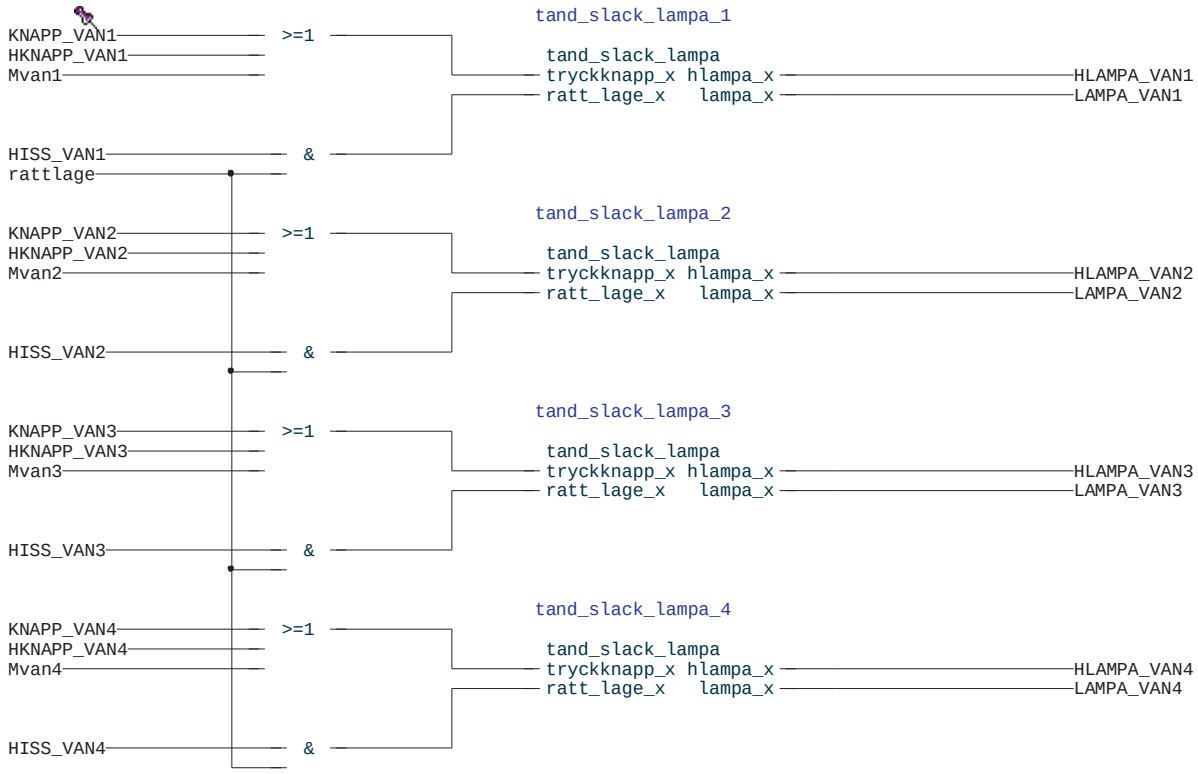


Figure 10. lampor - Program - Hiss_2.Hiss_2

Riktning

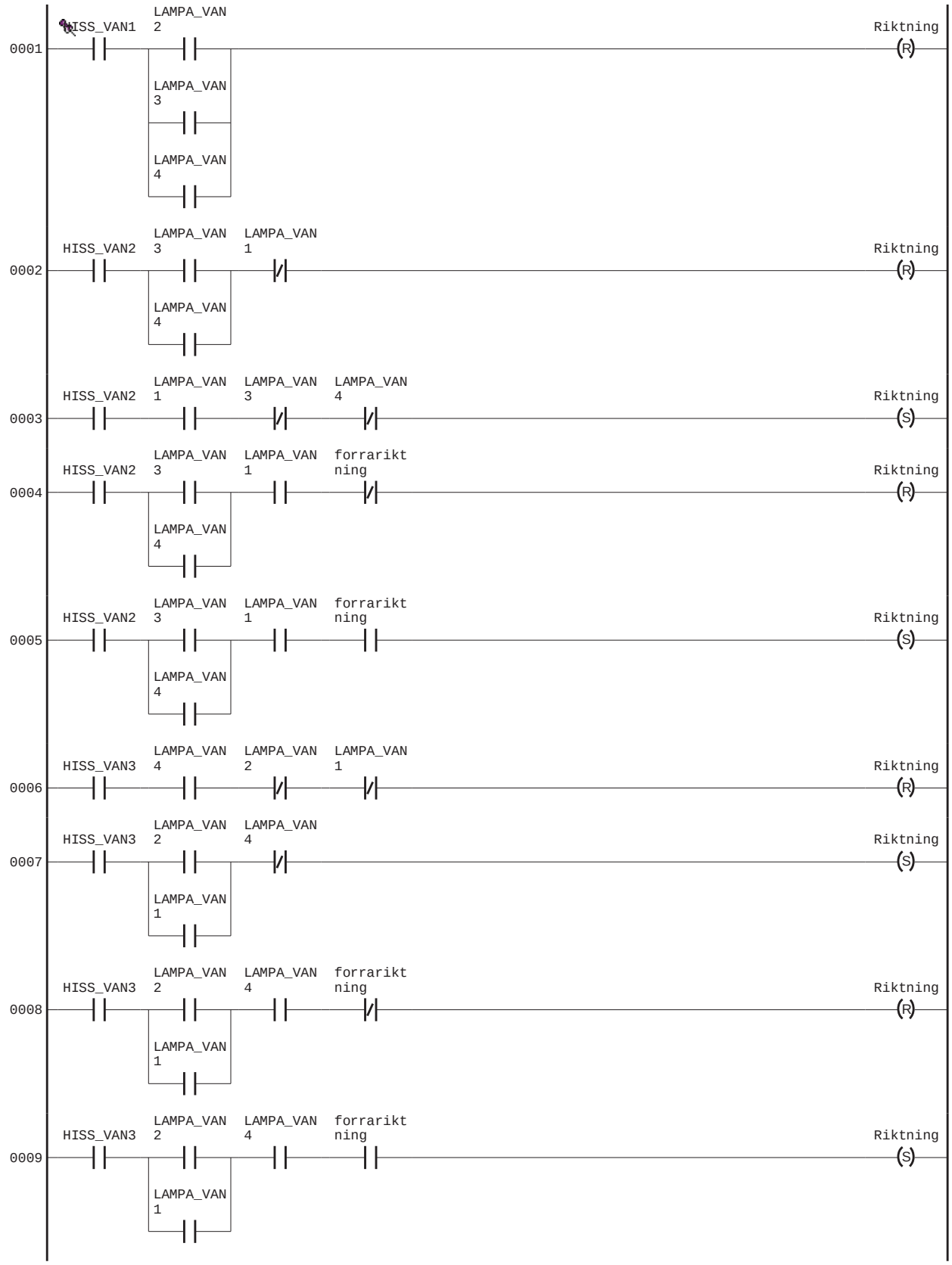


Figure 11. riktning - Program - Hiss_2.Hiss_2

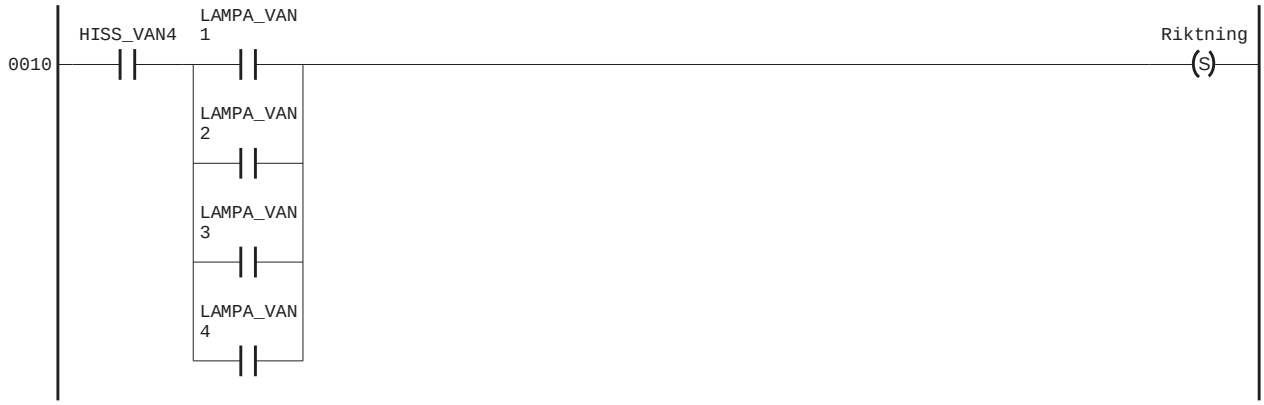


Figure 12. riktning - Program - Hiss_2.Hiss_2

Hiss

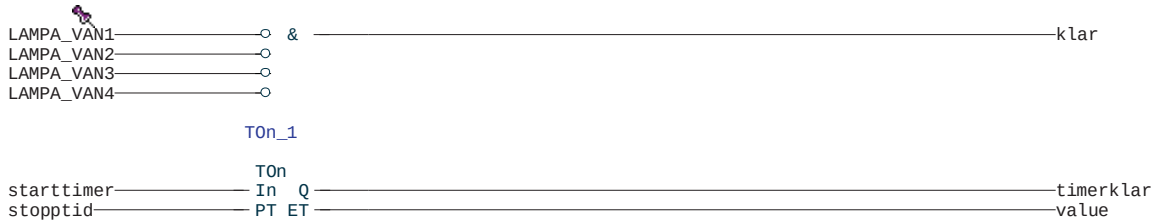


Figure 13. hiss - Program - Hiss_2.Hiss_2

Hiss_sekvens

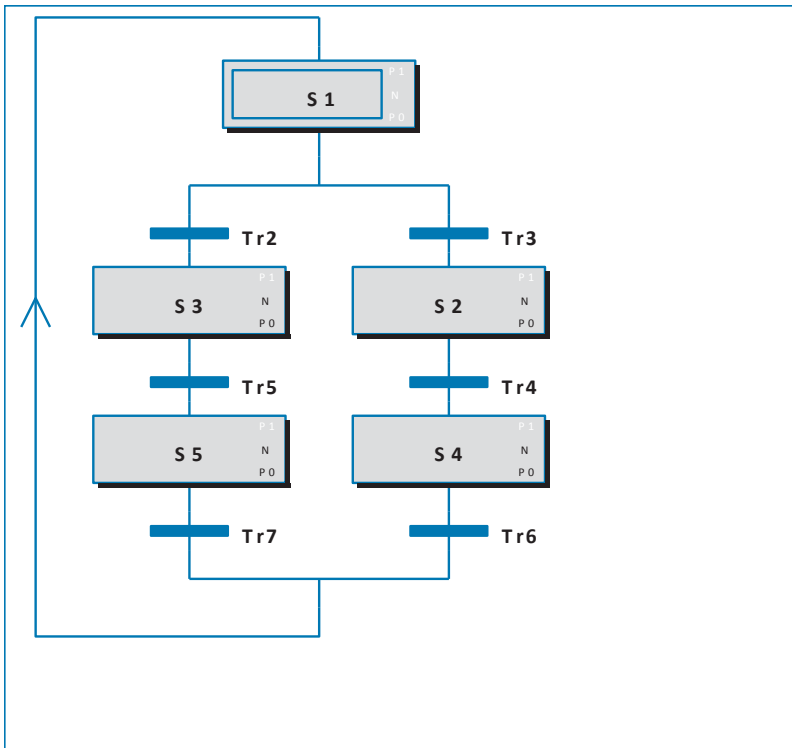


Figure 14. Hiss_sekvens_Program - Hiss_2.Hiss_2

S1

S2

S2_N

HISS_NER := true ;

S2_P0

HISS_NER := false;

S3

S3_N

HISS_UPP := true;

S3_P0

HISS_UPP := false;

S4

S4_N

starttimer := true ;

S4_P0

starttimer := false;

S5

S5_N

starttimer := true;

S5_P0

starttimer := false;

Tr2

not riktning and not klar

Tr3

riktning and not klar

Tr4

rattlage

Tr5

rattlage

Tr6

timerklar

Tr7

timerklar

Sätta_alarm

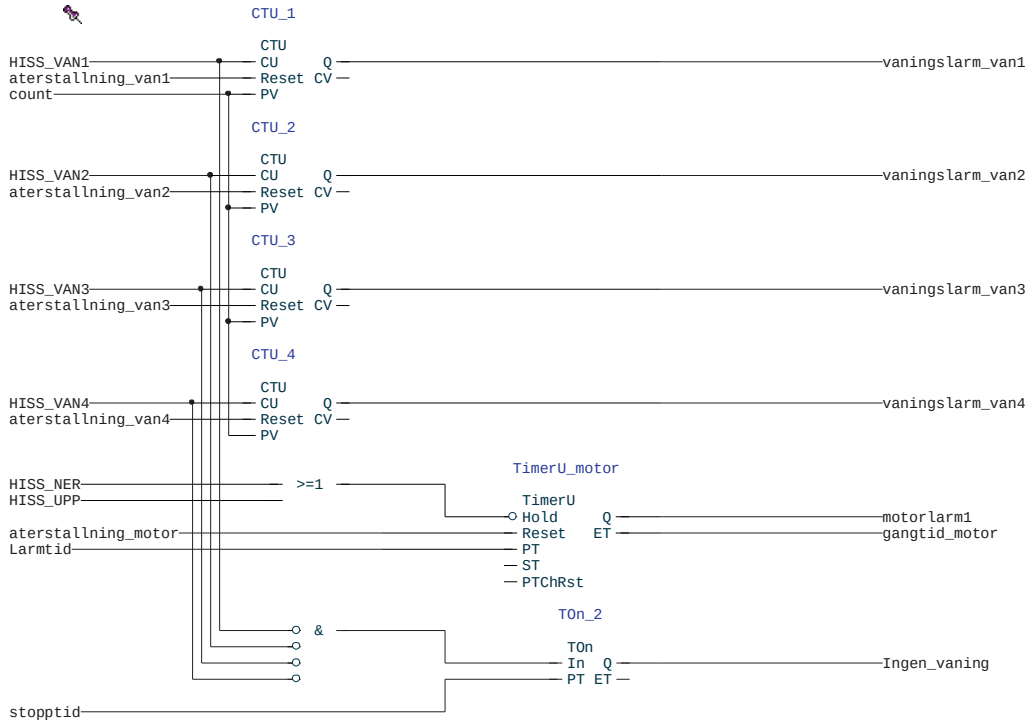


Figure 15. Sätta_alarm - Program - Hiss_2.Hiss_2

Larm_till_800xA

```
Vaningslarm( Signal := Ingen_vaning,
        SrcName := VaningslarmSrcName,
        CondName := VaningslarmCondName,
        Message := VaningslarmMessage,
        Severity := 900 );

Givarlarm_van1( Signal := vaningslarm_van1,
          SrcName := Givarlarm_van1SrcName,
          CondName := Givarlarm_van1CondName,
                Message := Givarlarm_van1Message );

Givarlarm_van2( Signal := vaningslarm_van2,
          SrcName := Givarlarm_van2SrcName,
          CondName := Givarlarm_van2CondName,
                Message := Givarlarm_van2Message );

Givarlarm_van3( Signal := vaningslarm_van3,
          SrcName := Givarlarm_van3SrcName,
          CondName := Givarlarm_van3CondName,
                Message := Givarlarm_van3Message );
```

```
Givarlarm_van4( Signal := vaningslarm_van4,
        SrcName := Givarlarm_van4SrcName,
        CondName := Givarlarm_van4CondName,
            Message := Givarlarm_van4Message
 );

Motorlarm_2( Signal := motorlarm1,
        SrcName := Motorlarm_2SrcName,
        CondName := Motorlarm_2CondName,
        Message := Motorlarm_2Message,
        Severity := 600 );
```

mellan_våning

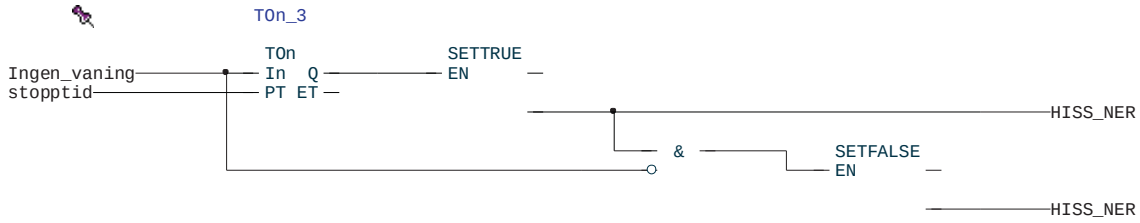


Figure 16. mellan_våning - Program - Hiss_2.Hiss_2