LUNDS UNIVERSITET
Lunds Tekniska Högskola

Logo Schneider Electric
(Schneider Electric 2009)

Robot på Schneider Electric

Av

Flamur Breznica och Hoong Hong

Avdelningen för Industriell Elektroteknik och Automation (IEA)

LTH Ingenjörshögskolan vid Campus Helsingborg, Lunds Universitet
221 00 Lund, Sverige

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

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

Tryckt i Sverige
Avd. för Industriell Elektroteknik och Automation
Lunds universitet
Lund 2015

Sammanfattning

Rapporten omfattar ett projekt som utförts hos Schneider Electric i Helsingborg där syftet var att använda en gammal robot med tre linjära axlar. Motorerna byttes ut och ett elskåp med Schneiders senaste produkter byggdes för att tillsammans samverka till att bli en fullständig enhet. Enheten monterades i en bur för att sen utnyttjas av Schneider Electric för marknadsföring. Projektet grundas på en produkt som Schneider Electric säljer för att förenkla uppbyggnaden av skåp till sina kunder.

Robotens uppgift är att kunna utföra en automatisk rörelse samt manuell körning. Den automatiska rörelsen består av en plocka-och-släpp-funktion medan manuell körning tillåter användaren att styra axlarnas beteende manuellt. Roboten kan styras trådlöst via olika externa användargränssnitt som t.ex. mobiltelefoner, bärbara datorer och surfplattor.

Nyckelord: Robot, programmering, elskåp, montering, koppling, användargränssnitt, mobiltelefoner, surfplattor, bärbar dator, accesspunkt

Abstract

The report includes a project carried out at Schneider Electric in Helsingborg, where the objective was to use an old robot with three linear axes. The engines were replaced and control cabinets with their latest products were built together to work and to become a complete unit. The complete unit was mounted on a cage which will be used by Schneider Electric for marketing. The Project is based on a product that Schneider Electric sells to simplify the construction of control cabinets to its customers.

The robot should be able to perform tasks both in automatic mode and in manual mode. The automatic movement consists of a pick-and-drop function while manual operation allows the user to control the behavior of the axes manually. The robot can be controlled by a user interface with external devices such as mobile phones, laptop or tablet devices through a wireless access point.

Keywords: Robot, programming, control cabinet, installation, wiring, user interface, cellphone, tablet, laptop, accesspoint

Robot, programmering, elskåp, montering, koppling, användargränssnitt, mobiltelefoner, surfplattor, bärbar dator, accesspunkt

Förord

Detta examensarbete utfördes hos Schneider Electric under våren 2015 och är den avslutande delen för högskoleingenjörs-utbildningen Elektroteknik och Automationsteknik vid Lunds Tekniska Högskola på Campus Helsingborg. Projektet omfattar 22,5 högskolepoäng.

Vi vill tacka **Schneider Electric** för tiden vi har varit hos dem och fått chansen att ta en del av deras dagliga liv samt fått den underbara upplevelsen. Ett speciellt tack till:

Josef Christoffersson

Thomas Karlsson

Johan Hult

Tack för ni har stöttat oss genom hela resans gång och hjälp oss under hela examensarbetet. Detta hade aldrig gått utan er hjälp.

Vi vill också tacka **Rolf Grahl** på **Emil Lundgren** för den tiden vi fick vara i deras verkstad och bygga ihop elskåpet samt för vägledningen under processen av kopplingen samt driftsättningen.

Vi vill också tacka **Mats Lilja**, vår examinator.

Sist men inte minst vill vi även tacka vår handledare, **Johan Björnstedt** för hjälpen med rapportens strukturering samt tips och råd.

Hoong Hong och Flamur Breznica

Innehållsförteckning

1. Inledning ... 1
1.1 Bakgrund ... 1
1.2 Syfte ... 1
1.3 Problemformulering ... 1
1.4 Mål ... 2
1.5 Avgränsning ... 3
2. Roboten ... 4
3. Apparatskåpskonstruktion ... 6
3.1 TVDA ... 6
3.2 Funktionsbeskrivning ... 7
3.3 Komponenter ... 8
3.3.1 Styrsystemet - LMC078 (Logic Motion Controller 078) ... 8
3.3.2 Servoenhet- Lexium 32s ... 9
3.3.3 Säkerhetsstyrenhet - Preventa Modular Safety controller XPSMCM ... 10
3.3.4 Övriga komponenter ... 11
3.4 Upplägg av skåpet ... 11
3.5 Dokumentation ... 13
4. Styrsystem ... 15
4.1 SoMove ... 15
4.2 SoMachine ... 15
4.2.1 Init ... 16
4.2.2 Input ... 17
4.2.3 ModeSelect ... 17
4.2.4 Manual ... 18
4.2.5 Auto ... 19
4.2.6 IntoPos ... 20

4.2.7 Stop 20
4.2.8 ErrorHandler 20
4.2.9 Motorer 20
4.2.10 PowerMeter 21
4.2.11 WebVisualization 21
4.3 SoSafe 23
4.4 Vanligt förekommande mjukvaruproblem 24
5. Slutsats 25
5.1 TVDA 25
5.2 Testkörning av robot 25
5.3 Framtida arbete 26
Terminologi 27
Referenser 28

1. Inledning

I detta kapitel ges en kort introduktion om Schneider Electrics bakgrund, syfte med projektet samt avgränsningar till projektet som utfördes hos Schneider Electric.

1.1 Bakgrund

Schneider Electric är ett franskt företag som grundades år 1836. Från början jobbade Schneider Electric mycket med järn- och stålindustrin, tunga maskiner och skeppsbygge men idag fokuserar de mer på elhantering och automatisering.

Vid första kontakten med Schneider Electric meddelades att det fanns ett tänkbart examensarbete som handlade om att återanvända en robot och bygga ett elskåp för roboten endast med produkter från Schneider Electric som används i dagens läge.

Genom att endast använda Schneider Electrics produkter för uppbyggnaden av elskåpet visar Schneider Electric upp sina produkter. Möjligheten finns att kunna få tag på alla produkter som krävs för att bygga ett elskåp från ett och samma företag.

1.2 Syfte

Syftet med projektet var att programmera en gammal robot samt bygga ett elskåp till roboten med endast Schneider Electrics produkter som sedan skall utföra en rörelse som plockar och släpper. Projektet grundas på en tidsbesparande produkt som säljs av Schneider Electric och som erbjuder en bas för utformningen av elskåp i många olika tillämpningar.

1.3 Problemformulering

En av de viktigaste aspekterna i dagens automationsbransch är tiden. Kunder har idag stora krav på att få saker gjorda på väldigt kort tid, vilket skapar problematiska tankar hos företag. Detta är något som

Schneider Electrics has lagt en stor vikt på då verksamheten valt att ta fram något som kallas för en TVDA (Tested, Validated, Documented Architectures). Detta är en mall som kunderna ska spara tid på när de arbetar med uppbyggnad av robot, vilket i sin tur leder till bättre och snabbare färdiga produkter som i slutändan gynnar kunden.

Genom att använda denna mall ska det påvisas om mallen TVDA faktiskt hjälper till att korta ner arbetstiden för att ta fram en robot. Detta görs genom att bygga upp en robot för att se hur TVDA hjälper en utövare att påskynda sitt arbete.

Därför ställs följande problemformulering:

- Är TVDA användbart och effektivt vid design av ett styrskåp för en automationsprocess bestående av en 3-axlig robot?

1.4 Mål

Målet var att framställa en färdig produkt som senare skall kunna användas för att demonstrera möjligheter att kombinera Schneider Electrics produkter på ett marknadsföringsmässigt sätt. Med hjälp av slutprodukten skall Schneider Electric kunna visa sin mångsidighet, exempelvis att ta produkter från olika industriavdelningar hos Schneider Electric och sammanfläta dem till en samverkande enhet. Den slutgiltiga produkten ska innehålla följande:

- Elskåp som baseras på komponenter från Schneider
- Programmera roboten till en plocka och släppa rörelse med hjälp av en elektromagnet
- Programmera en manuell körning på roboten
- Skapa ett användargränssnitt till roboten
- Läsa av energiförbrukningen med hjälp av en energimätare
- Säkerhetsfunktioner som nödstoppsknapp och ljusridå

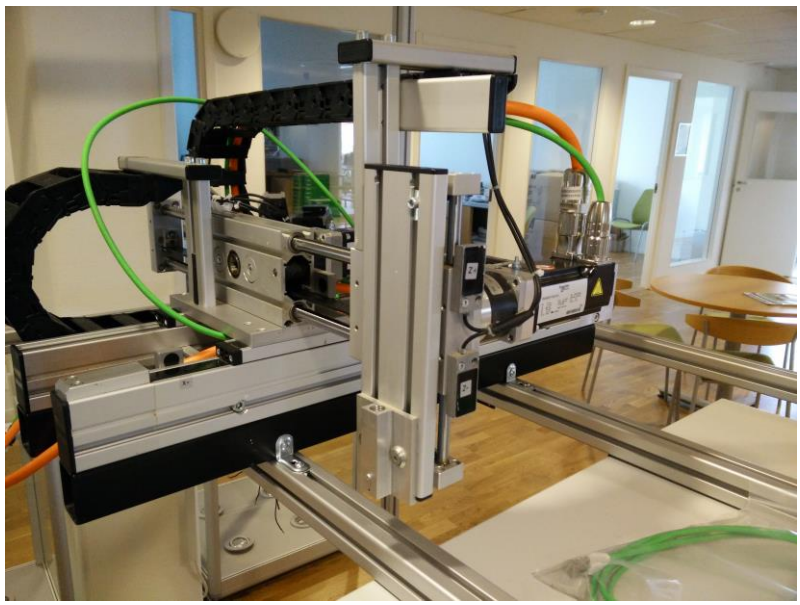
1.5 Avgränsning

Programmeringen av roboten begränsas till utförandet av en enkel dra-och-släpp-rörelse.

En annan avgränsning som gjordes var interpolering. Interpolering innebär att en robot utnyttjar sina 3 axlar genom att använda sig av en matematisk algoritm för att robotarmen skall nå en specificerade punkt via en jämn och kontinuerlig rörelse.

Det kosmetiska för hela uppställningen av roboten samt skåpet görs i efterhand av Schneider Electric och ingår inte.

2. Roboten



Figur 2.1 En bild på robotens axlar

Roboten som användes är en treaxlig linjärenhet, vilket innebär att roboten kan drivas i tre olika led (x, y och z) och har en arbetsytan på cirka 0.2m^2 Robotens växellåda har skalningen 8:1, det vill säga att det krävs åtta motorvarv för att växeln skall rotera ett varv. På roboten sitter tre stycken BSH0551T02A1A motorer.

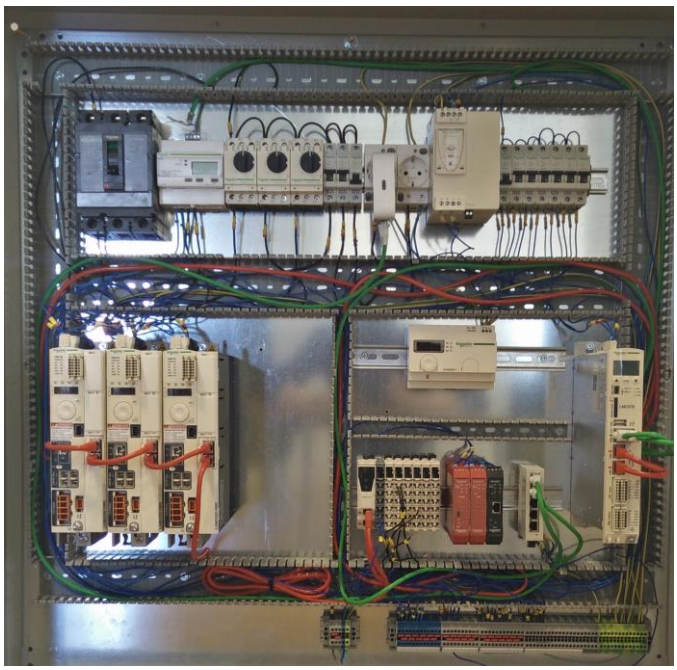
Motorerna av typen Lexium BSH styrs av servoenheter av typ Lexium 32. Motorerna är det perfekta valet för att uppfylla kraven för dynamiken och prestandan. De är konstruerade för att tåla -20 till 40°C värme, detta innebär att ifall Schneider Electric önskar kan de ha roboten utomhus och köra den även när det är kallt. De har även egenskapen “multiturn”, vilket innebär att positionen kan skrivas i motorernas kodavläsare som sedan används för att komma ihåg sina senaste positioner. Motorerna har ett maximalt varvtal på 9000 rpm. Dessa motorer klarar av att köra med 230 volt enfas men klarar även av trefas, skillnaden är att drivkraften blir lägre på 230 volt än om det hade använts 400 volt trefas.

Roboten monteras i en bur där den skall samverka med ett apparatskåp som har till syfte att marknadsföra Schneider Electric. När roboten körs skall den utföra en plocka-och-släpp-rörelse. Roboten är kapabel till mycket mer men avgränsning har gjorts på grund av begränsad tid. I kombination med elskåpet finns möjligheten att montera en elektromagnet på z-axeln för möjligheten att klara av att lyfta ett föremål exempelvis ett mynt.

Det skall implementeras nödstopp samt ljusridå som skall stoppa roboten om exempelvis någon sticker in handen, fingret eller vid knapptryckning av nödstopp. Syftet med ljusridån är att slippa använda någon dörr så att roboten skall ha fri sikt och undvika skador om någon försöker sticka in en hand eller finger. Då skall de tre motorerna stanna på ett säkert sätt innan någon hunnit sticka in handen tillräckligt långt för att skada ska uppstå.

3. Apparatskåpskonstruktion

I detta kapitel presenteras Schneider Electrics TVDA-arkitektur. I kapitlet beskrivs också valet av komponenter och hur dessa monterades i elskåpet.



Figur 3.1 Illustrerar elskåpet

3.1 TVDA

TVDA är en arkitektur, lanserad av Schneider Electric, som erbjuder en bas för utformningen av elskåp inom många olika tillämpningar där ett av huvudmålen är att spara tid. Ett TVDA dokument består av:

- Design av elskåpets insida
- Elritningar
- Manualer till enheter
- Programmallar för Schneider Electrics programmeringsmiljö SoMachine
- HMI-applikationer

Dokumentet beskriver design av elskåp, elritning samt manualer till de valda produkterna. Dokumentet innehåller även mallar och HMI (Human Machine Interface) applikationer på SoMachine.

Ett exempel är ett företag som ska bygga en maskin för tillverkning av pappkartonger. Maskiner inom denna bransch fungerar efter ungefär samma principer och de behöver därför väsentligen samma typ av styrsystem.

Till dessa kunder erbjuder Schneider Electric en TVDA. Den innehåller ritningar på hur man ska koppla och eventuellt mjukvaror som behövs. Den ger även förslag på hur en sådan maskin kan styras och en dokumentation på hur det hela fungerar.

Alla kunder som ska bygga en maskin av denna typ behöver ritningar och mjukvara. Genom att använda TVDA vinns massor av tid på att börja på något färdigt där man kanske behöver ändra lite små saker. Eftersom mjukvaran också är testad av Schneider Electric och flera andra kunder innebär det hög kvalité på mjukvaran.

Under projektet fanns en TVDA som var inköpt till ett annat projekt. Eftersom den innehöll produkter som inte ska vara med i detta projekt fick TVDA modifieras om, framförallt fick elritningen redigeras om. De stora skillnaderna var att TVDA:n var utformad med trefas medan projektet gick ut på att använda enfas, energimätaren var annorlunda samt att TVDA:n var utformad för en sexaxlig robot medan det gamla projektet innehöll en treaxlig robot.

3.2 Funktionsbeskrivning

Skåpet spänningssätts med 230 volts enfas som sedan går runt i skåpet för att spänningssätta energimätaren, servoenheterna samt 24-volts nätaggregat.

Servoenheterna tar emot mätsignaler från motorerna som exempelvis position och varvtal.

Skåpet ska på ett säkert sätt styra roboten samt hantera elektriska problem som uppstår. Vid kortslutning eller liknande problem skall säkringarna bryta för att undvika förstörelse av hårdvaror.

Mätningen av hur mycket energi som går åt i skåpet sker genom en energimätare som sedan kommunicerar med styrenheten via Ethernet (Sercos III) för att sedan läsa av informationen genom programmet.

Energimätaren mäter saker som frekvens, ström samt spänning med flera. Genom kommunikationen klarar den av att visa informationen på en HMI eller på ett användargränssnitt som finns på styrenheten.

Skåpet skall kunna hantera in och utgångar om så önskas som exempelvis givare på motorn. Ingångar finns förberedda för att kunna ta in olika givarsignaler.

Skåpet har även två eluttag för att spänningssätta en accesspunkt samt en möjlighet till att kunna koppla eller ladda enheten som styr roboten. En accesspunkt sätts in i ena uttaget för att via ett trådlöst nät kunna styra roboten.

3.3 Komponenter

Produkterna valdes utifrån Schneider Electrics katalog och tillhör det nyaste från sortimentet därifrån.

3.3.1 Styrsystemet - LMC078 (Logic Motion Controller 078)

Figur 3.2 Illustrerar styrenheten LMC078

LMC078 är en styrenhet som klarar av många olika funktioner och kan kontrollera ett stort antal tillämpningar. Styrenheten kan skapa och synkronisera en rörelsefunktion för en maskin med upp till 16 axlar. Den minsta synkroniseringstid och cykeltid på styrenheten är 2ms och 250µs. Dessutom klarar styrenheten av följande sex olika programmeringsspråk:

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

På LMC078 finns det inbyggda fältbussar med kommunikationsgränssnitt som Sercos III, CANopen och Ethernet TCP/IP (Transmission Control Protocol/Internet Protocol). Det finns även inbyggda digitala I/O (ingångar/utgångar) på styrenheten samt portar för externa ingångar och utgångar från CANopen och Sercos fältbussar som exempelvis en TM5 modul. Med hjälp av Schneider Electrics programmeringsmiljö, SoMachine blir LMC078 en optimal lösning för axelpositionering.

3.3.2 Servoenhet- Lexium 32s

Figur 3.3 Illustrerar servoenheten Lexium 32s

Lexium 32 är en produkt som består av många olika modeller och täcker olika applikationsområden. Tillsammans med Lexium BMH eller Lexium BSH motorer samt ett omfattande tillval av tillbehör är enheterna idealiska för att genomföra kompakta, högpresterande drivlösningar för ett brett spektrum av kraftkrav.

Lexium 32s är även fylld med funktioner som kommunikationsgränssnittet Sercos III, kylningen är designad så de kan ställas tätt intill varandra, samt STO (Safe Torque Off). STO är en säkerhetsfunktion som enligt IEC 61800-5-2 (International Electrotechnical Commission) implementerats.

3.3.3 Säkerhetsstyrenhet - Preventa Modular Safety controller XPSMCM

Figur 3.4 Illustrerar säkertsstyrenheten XPSMCM

Säkerhetsstyrenheten består av flera moduler som går att kombineras tillsammans. Eftersom det finns en stor variation av moduler kan säkerhetsstyrenheten konfigureras på flera sätt, exempelvis genom fältbuss kommunikation som Modbus TCP, Ethernet TCP/IP eller CANopen.

Säkerhetsstyrenheten skall hantera säkerhetsfunktioner som nödstopp-knappar eller ljusridån. Om någon sticker in handen i roboten eller trycker på nödstoppen skall roboten stoppas och det är säkerhetsstyrenhetens uppgift. Eftersom det är en programmerbar säkerhetsstyrenhet som kommunicerar med styrenheten kan upplägget se ut på många sätt.

Programmet kan exempelvis läggas upp som så att signalen fördröjs på 100ms innan säkerhetsstyrenheten bryter spänningen till STO på

servoenheten som i sin tur stoppar motorerna med hjälp av inbyggda säkerhetsfunktionen. Då har styrenheten en tidsram på 100ms där den skall stoppa roboten, hinner den inte tar Lexium 32s hand om det. Detta är fördelen med en programmerbar säkerhetsstyrenhet, att signaler kan fördröjas och låta programmet sköta roboten istället för att bryta spänningen till servoenheten direkt.

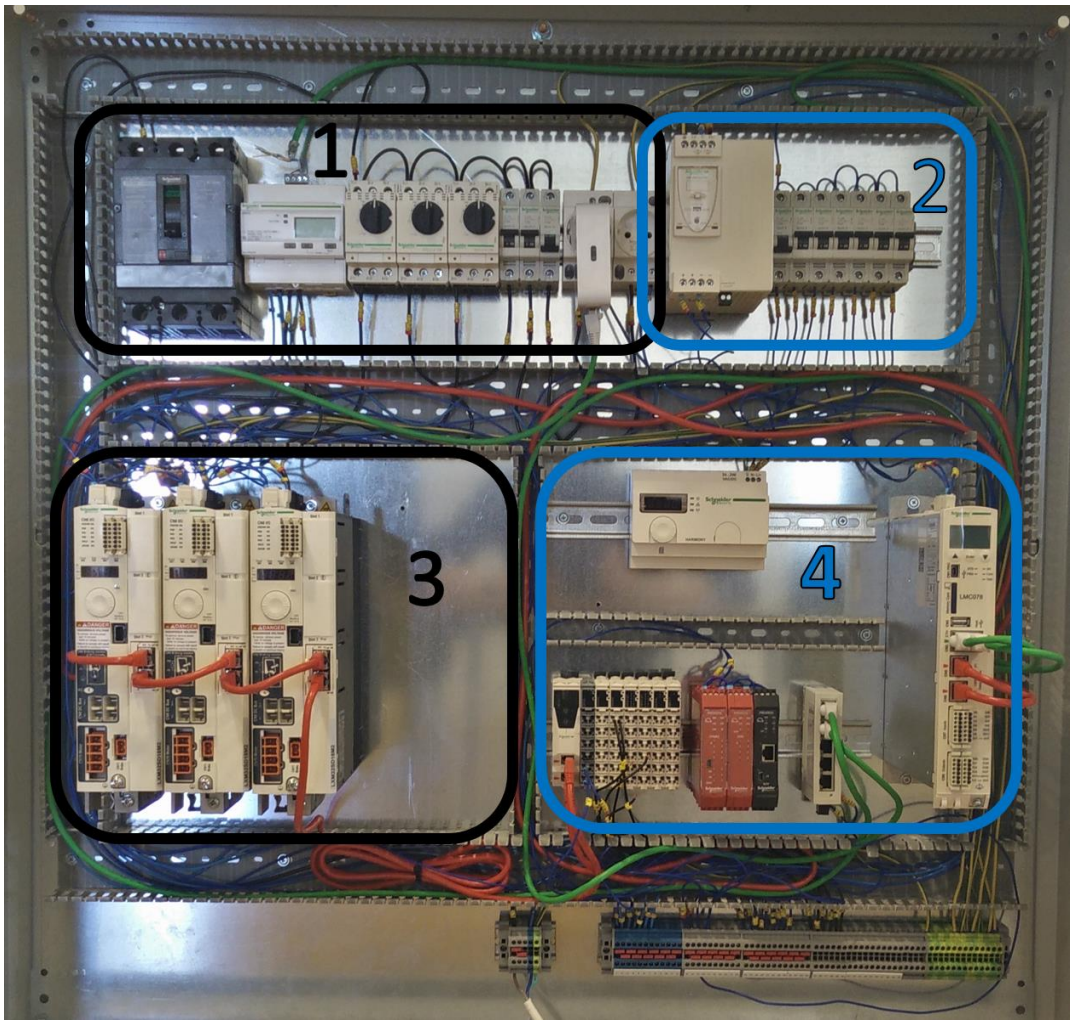
3.3.4 Övriga komponenter

Förutom styrsystemet, säkerhetssystemet och servoenheten behövs det även:

- Ethernet Switch
- TM5 I/O (ingångar/utgångar) ö
- 24 Volt spänningsaggregat
- PowerPact H-Frame (strömbrytare)
- Power Meter iEM3155
- 3 stycken Dvärgbrytare 10A
- 5 stycken Dvärgbrytare 4A
- 5 stycken Dvärgbrytare 2A
- 2 stycken Vägguttag
- DIN-skenor (för fästningen av produkter på elskåpet)
- 100 m R.K kablar 0.75 mm² i färgerna: mörk blå, svart
- 100 m R.K kablar 1.5 mm² i färgerna: ljusblå, svart, gul/grön
- Plintar (färger: grå, blå, gul/grön)

3.4 Upplägg av skåpet

Upplägget av elskåp kan variera från byggare till byggare men i detta projekt är upplägget av elskåpet uppdelat fyra olika sektioner som illustreras i *figur 3.5*.



Figur 3.5 Illustrerar elskåpets fyra uppdelade sektioner

När elskåp byggs upp skall servoenheten helst vara skild från styrenheten, ingångar/utgångar ön och säkerhetsstyrenheten på grund av att servoenheten avger värme när de är igång. Helst skall servoenheten sitta på nedre delen av elskåpet eftersom när värme bildas inne i skåpet kommer värmen att stiga uppåt.

I projektet placeras servoenheten i den vänstra nedre delen (*se figur 3.5 sektion 3*) vilket leder till att styrenheten, ingångar/utgångar ön och säkerhetsstyrenheten placeras i den högra nedre delen (*se figur 3.5*

sektion 4) för att inte påverkas så mycket av värmen som avges från servoenheten.

Eftersom säkringar och nätaggregat inte påverkas lätt av värme placeras de i övre delen av skåpet. För att urskilja vilka säkringar som tillhör 230 volt eller 24 volt är de fördelade på olika sektioner. På högra delen sitter 230 volt sektionen (*se figur 3.5 sektion 1*) medan 24 volt sektionen på vänstra delen (*se figur 3.5 sektion 2*)

Alla kablar som dras internt skall kabelmärkas med en nolla framför för att enkelt visa att kabeln är dragen internt mellan produkterna. Alla kablar som dras externt skall ner till en plint som sedan markeras med ett nummer för att enklare kunna lokalisera kablarna som ligger externt. Vilket nummer som används på varje kabel har inte större betydelse men det kan vara lämpligt att vara konsekvent när numret väljs. Det är väldigt viktigt att det inte får finnas dubbletter eller flera med likadan kabelmärkning.

Det finns olika färgstandarder som följs när det gäller kablar och plintar. Den som användes är följande:

För kablar: (måtten kan variera beroende hur stort elskåpet är men färgen ska vara detsamma)

- +230 volt: R.K svart 1,5 mm^2
- 230 volt nolla: R.K ljus blått 1.5 mm^2
- +24 volt: R.K mörk blått 0.75mm^2
- 24 volt nolla: R.K mörk blått 0.75mm^2
- Digital signal: R.K svart 0.75mm^2
- Jord: R.K Gulgrön 1.5mm^2

För plintar:

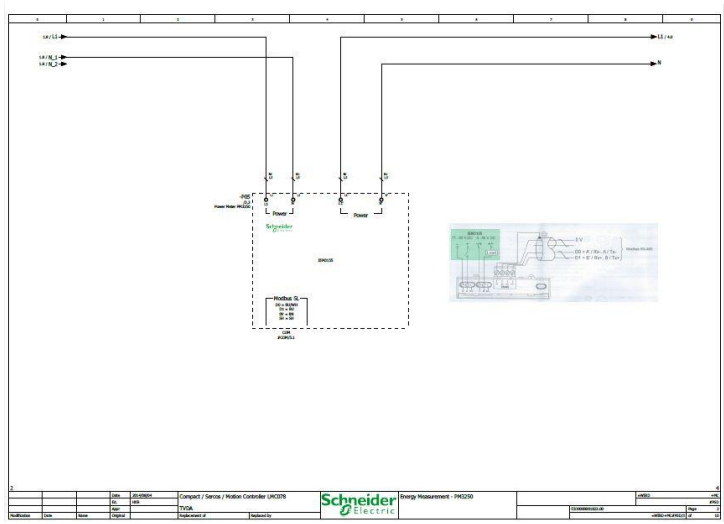
- +230 volt: Grå
- 230 volt nolla: Mörkblå
- +24 volt: Grå
- 24 volt nolla: Grå
- Digital signal: Grå
- Jord: Gulgrön

3.5 Dokumentation

TVDA har möjliggjort illustration av elritningar utan att behöva använda något cadprogram.

För att anpassa TVDA till detta projekt krävdes redigering eftersom den ursprungliga TVDA var anpassad för ett annat projekt med sex motorer som drevs av sex servoenheter. I detta projekt finns det enbart tre motorer som drivs av tre servoenheter vilket ledde till att ett antal redigering gjordes. Dessutom innehöll den ursprungliga TVDA andra produkter som uteslöts från detta projekt, som t.ex. frekvensomvandlare.

Ett exempel på redigering var energimätaren som ritades om enligt figur 3.6. Den ursprungliga ritningen var ritad för en Power Meter iEM3250 medan i projektet användes en Power Meter iEM3155 vilket ledde till att hela ritningen för energimätaren fick ritas om.



Figur 3.6 Illustrerar energimätaren på elritningen

Dessutom var den ursprungliga TVDA anpassad för trefas vilket fick ändras om eftersom det endast utnyttjades enfas på projektet, se bilaga 3 för mer exempel.

I slutändan blev den modifierade elritningen basen för konstruktionen av elskåpet. Trots all redigering sparades massor av tid jämfört med ifall elritningen skulle ritas från grunden. För en översikt av kabeldiagram se på Bilaga 4.

4. Styrsystem

I skåpet sitter en styrenhet och en säkerhetsstyrenhet som programmeras med varsitt program. Roboten skall konfigureras till en plock-och-släpp-rörelse med nödstopp samt ljusridå. Konfigurationen krävde totalt tre program som användes till olika ändamål. SoMove för att ställa in positionen på motorerna, SoMachine för att programmera styrenheten och SoSafe för att programmera säkerhetsstyrenheten.

4.1 SoMove

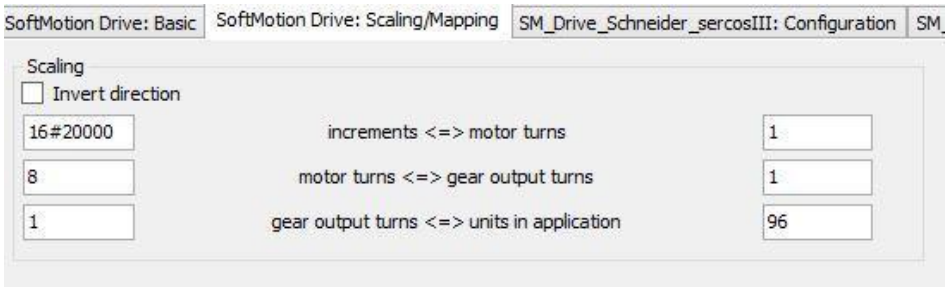
SoMove är ett användarvänligt installationsprogram för Windows som används till att ställa in Schneider Electric motorstyrenheter. Programvaran innehåller olika funktioner för enhetens startinställningar, såsom att konfigurera, installera samt underhålla motorstyrenheter.

Programmet användes för att konfigurera inställningar på servoenheten. I projektet användes SoMove till att skriva motorernas nuvarande position till kodavläsaren. Det borde räcka med att skriva till kodavläsaren en gång men under vissa omständigheter fick det göras igen. Detta berodde på att timingen mellan servoenheten och styrenheten inte stämde överens vid uppstart.

4.2 SoMachine

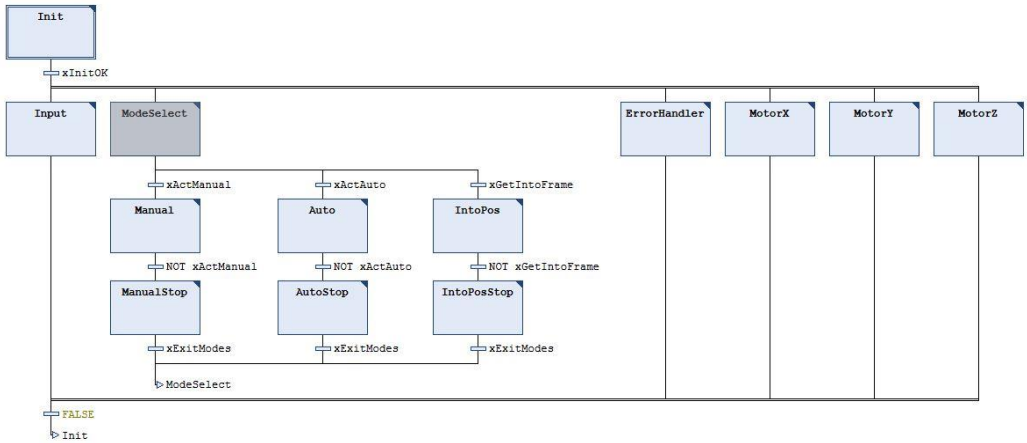
SoMachine är ett program som hanterar styrenheten. För att kommunikationen mellan dator och styrsystem ska fungera måste dessa vara i samma subnät. Användargränssnittet visas genom en surfplatta och för att möjliggöra detta krävdes en accesspunkt. För att underlätta inloggningen till styrenheten specificerades en IP adress till styrenheten med hjälp av MAC-adressen (Media Access Control).

SoMachine erbjuder möjligheten att kunna skala ner positionerna för att enheterna i programmet skall vara i millimeter. Detta är något som lägger grunden för att plocka-och-släpp-rörelsen skall fungera.



Figur 4.1 Visar hur skalningen på motorerna ställts in

Programmet är uppdelat i tre delar där motorstyrningen skrevs i SFC enligt figur 4.2, energimätarens avläsning skrevs i CFC och användargränssnittet skapades med hjälp av applikationen WebVisualization.



Figur 4.2 Beskriver strukturen på programmet som styr motorn

4.2.1 Init

Init programmerades i strukturerad text. Fördelen med Init är att koden exekveras en gång vid uppstart för att sätta startvärden. Variablerna som används sätts till false för att de flesta variablerna kräver en stigande flank när de skall aktiveras.

```
diBootState := SEC.FC_GetBootState();

IF diBootState = 1 THEN
    xLMC078BootOk := TRUE;
ELSE
    xLMC078BootOk := FALSE;
END_IF

IF SERCOSIII.State =4 AND xLMC078BootOk THEN
    xInitOK := TRUE;
ELSE
    xInitOK:=FALSE;
END_IF
```

Figur 4.3 Illustrerar en liten del av koden skriven i Init

4.2.2 Input

Även här används strukturerad text för att en av uppgifterna var att konstant skriva värden till variabler för att undvika oavsiktliga förändringar. Även variabler som håller rätt på om motorn är stilla stående eller om knapparna som joggar motorerna skulle visas, specificerades här.

```
12  //DECIDES RANGE ON AXISES
13      lrMinXAxis:= -5 ;
14      lrMaxXaxis:= 390;
15      lrMinYAxis:= 0;
16      lrMaxYaxis:= 200;
17      lrMinZAxis:= -1;
18      lrMaxZaxis:= 125;
19
20      sLMCDiag:= LMC078CECS20T.DiagMsg;
21      // Creating Variable for knoving when machine standstill.
22
23      IF  (   eAxisStatMotorX = SM3_Basic.SMC_AXIS_STATE.standstill OR eAxisStatMotorX = SM3_Basic.SMC_AXIS_STATE.errorstop) AND
24          (   eAxisStatMotorY = SM3_Basic.SMC_AXIS_STATE.standstill OR eAxisStatMotorY = SM3_Basic.SMC_AXIS_STATE.errorstop) AND
25          (   eAxisStatMotorZ = SM3_Basic.SMC_AXIS_STATE.standstill OR eAxisStatMotorZ = SM3_Basic.SMC_AXIS_STATE.errorstop) THEN
26          xMachineStandstill := TRUE;
27      ELSE
28          xMachineStandstill := FALSE;
29      END_IF
30
```

Figur 4.4. Illustrerar delar av koden under Input

4.2.3 ModeSelect

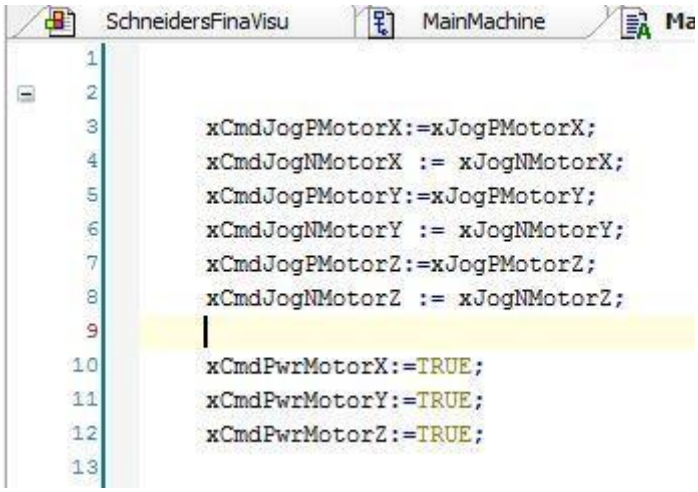
Under ModeSelect bryts spänningen till motorerna samt återställs värden som ändrats under Auto, Manual samt IntoPos för att kunna få en stigande flank när de återfinner sig i stegen.

```
xCmdStop:= FALSE;

xCmdPwrMotorX:=FALSE;
xCmdPwrMotorY:=FALSE;
xCmdPwrMotorZ:=FALSE;

xCmdMovAbsMotorX := FALSE;
xCmdMovAbsMotorY := FALSE;
xCmdMovAbsMotorZ := FALSE;
xExitModes:=FALSE;

MainMachine.xJogPMotorX:= FALSE;
MainMachine.xJogNMotorX:=FALSE;

MainMachine.xJogPMotorY:= FALSE;
MainMachine.xJogNMotorY:=FALSE;

MainMachine.xJogPMotorZ:= FALSE;
MainMachine.xJogNMotorZ:=FALSE;
```

Figur 4.5 Koden som tillhör ModeSelect

4.2.4 Manual

Under Manual ges spänning till motorerna samt tilldelar knapparna som används i användargränssnittet. Blocket som joggar motororena finns under motorernas egna steg. Det som sker när motorerna joggar är att motoraxeln rör sig långsamt åt ett specifikt håll i en långsam hastighet. Variablerna tilldelas värdena under användargränssnittet när programmet befinner sig i manual. Med användargränssnittet ska det gå att jogga motorn åt båda hållen.



Figur 4.6 Ett exempel på hur koden ser ut.

4.2.5 Auto

En plocka-och-släpp-rörelse ska skapas och det gjordes främst genom process satser. Steget använder sig av block ifrån motorernas steg. För att möjliggöra total kontroll har variabler satts som ingångar till blocken under motorernas steg. Absolutrörelse är lämplig för att göra denna typ av uppgifter då den använder sig av positioner för att ta reda på vart den ska röra sig till. Variablerna som används deklareras under motorernas steg. För att kunna återuppta automatiska processen skapas en ny variabel vars uppgift är att komma ihåg senaste processen.

```
        IF fbSM3MoveAbsMotorX.Done AND fbSM3MoveAbsMotorY.Done AND fbSM3MoveAbsMotorZ.Done  THEN
                xCmdMovAbsMotorX:=FALSE;
                xCmdMovAbsMotorY:=FALSE;
                xCmdMovAbsMotorZ:=FALSE;
                diState:=5;
        END_IF
    5:
    //Reset innan vi kör igång
        diOldDiState:=5;
        xCmdMovAbsMotorX:=FALSE;
        xCmdMovAbsMotorY:=FALSE;
        xCmdMovAbsMotorZ:=FALSE;

        diState:=10;
    10:
        diOldDiState:=10;
        lrSetPosMovAbsMotorZ:=10;
        xCmdMovAbsMotorZ:=TRUE;
        IF fbSM3MoveAbsMotorZ.Done THEN
            xCmdMovAbsMotorZ:=FALSE;
            diState:=15;
        END_IF

    15:
        diOldDiState:=15;
        lrSetPosMovAbsMotorZ:=10;
        xCmdMovAbsMotorZ:=TRUE;

        IF fbSM3MoveAbsMotorZ.Done THEN
            xCmdMovAbsMotorZ:=FALSE;
            diState:=20;
        END_IF
```

Figur 4.7 Illustrerar delar av hur Auto koden ser ut

4.2.6 IntoPos

IntoPos anropas av användaren genom användargränssnittet när positionen är utanför de tillåtna gränserna. Uppgiften som utförs är att motorn körs till en position innanför de uppsatta gränserna för att kunna köra motorn manuellt eller automatiskt.

4.2.7 Stop

Variabeln xCmdStop anropas för att stoppa motorerna, det är en ingång till blocket ”MC_Stop” som deklareras under motorernas steg. En kontroll av att motorn står stilla görs och när motorn är stillastående kan steget lämnas.

4.2.8 ErrorHandler

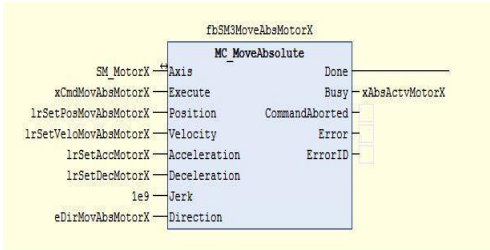
Skulle fel uppstå i programmet kommer det att kontrolleras samt hanteras i ErrorHandler. Funktionsblocket som läser felmeddelanden kallas “MC_ReadAxisError”. Det skapades en boolesk variabel som kontrollerar ifall någon av motorerna fått fel. Ifall felmeddelande uppstått på någon av motorerna stoppas alla motorer tills problemet lösts.

Fel kan ske på många olika sätt, det kan vara externa fel som kopplingen eller interna fel som uppstått i motorerna. För att hålla uppsikt på om ett fel på anslutningen har skett användes det en Sercos funktion som finns i SoMachine. Med hjälp av Sercos går det att se vilket tillstånd motorerna befinner sig i och om Sercos står på tillstånd fyra är allt i sin ordning. Detta utnyttjades genom att när tillståndet inte var fyra skulle motorerna stoppas och booleska variabeln sätts till sant.

En knapp från användargränssnittet styr variabeln som återställer felmeddelanden när ett fel har uppstått. Även mjukvaru-begränsning har satts upp för att skydda motorerna.

4.2.9 Motorer

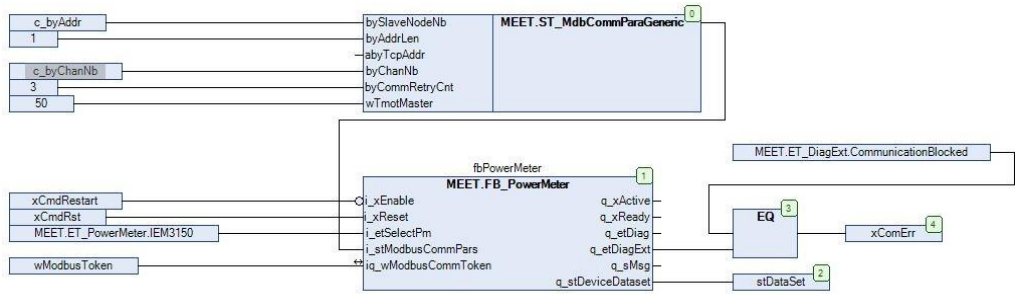
Motorerna skrevs i FBD för att få en överblick på motorernas funktioner. Det infördes block som kunde spänningssätta, stoppa, återuppta, starta om, jogga, läsa av positionen, absolut förflyttning samt läsa parametrar på motorerna. Motorerna har samma funktioner men namnet skiljer dem åt.



Figur 4.8 Illustrerar hur absolut blocket ser ut

4.2.10 PowerMeter

Energimätarens program är skrivet i CFC. Med hjälp av biblioteken MEET och MED läses variablerna ut med en sträng. För att hämta ut data från energimätaren användes en variabel som hette stDataset som sedan visas genom användargränssnittet. En väsentlig del att ha i åtanke var att energimätaren krävde mindre anrop per sekund vilket modifieras under fliken enheter. Energimätaren körs i MAST vilket skiljer sig från resten av programmet som skrevs i cykeln MOTION.



Figur 4.9 Visar hur delar av energimätarens funktionsblock ser ut

4.2.11 WebVisualization

SoMachine inkluderar en applikation, WebVisualization som även kan gå under namnet WebVisu. Genom att använda WebVisu skapas en hemsida med syftet att agera som ett användargränssnitt för att styra roboten.

För att komma åt användargränssnittet på styrenheten krävs IP-adressen 192.168.0.101:8080/webvisu.htm samt samma nätverk. Användargränssnittet är uppdelat i tre sidor, en startsida, en körlägessida och en informationssida.

Startsidan visar positionerna på motorerna samt två knappar som tar dig till kör läget respektive information läget. Den innehåller även en textruta som visas vid fel på motorn, annars är textrutan dold.

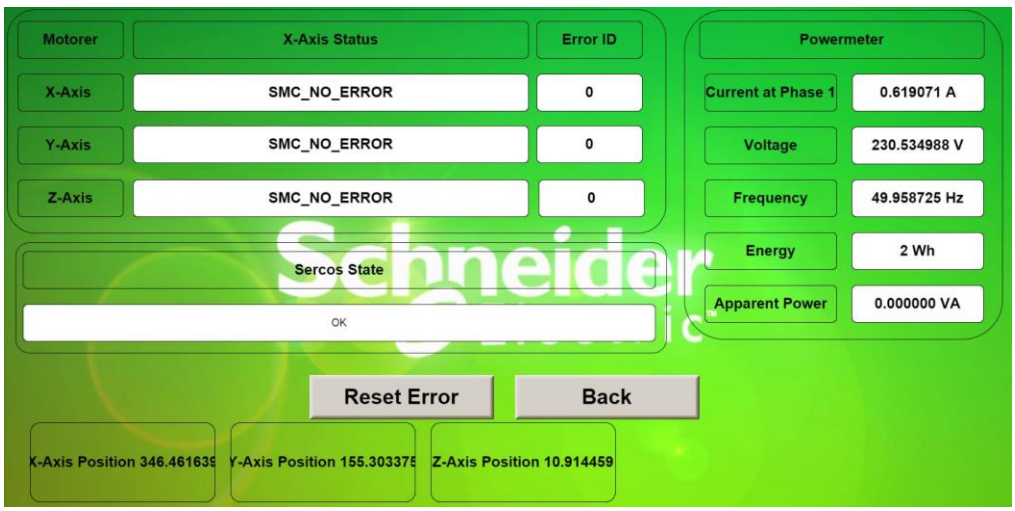
Figur 4.10. Illustrerar startsidan på användargränssnittet vid fel meddelanden

Sidan som kör motorerna innehåller information om deras positioner samt en dold textruta som dyker upp vid fel. Det finns även tre knappar som tillåter användaren att gå tillbaka till första sidan eller välja vad som önskas att köra, Auto eller Manual.

Vid Automatisk körning utförs en plocka-och-släpp-rörelse som repeteras tills stopp önskas. Däremot på Manual körning kan motorerna jogga med hjälp av extra styrknappar som dyker upp under Manual körning. Ifall någon av motorerna joggas utanför uppsatta gränser kommer knapparna som joggar motorerna att försvinna och ersättas av knappen IntoPos.

Figur 4.11: Illustrerar hur användargränssnittet ser ut när joggning av motorn sker utan fel

Information sidan innehåller information om vad som händer i roboten. Där visas energimätarens mätvärde samt felmeddelande i motorerna. Det förekommer även en knapp som rensar bort felen för att kunna återuppta arbetet. Felen måste först rättas till innan felmeddelanden kan försvinna.



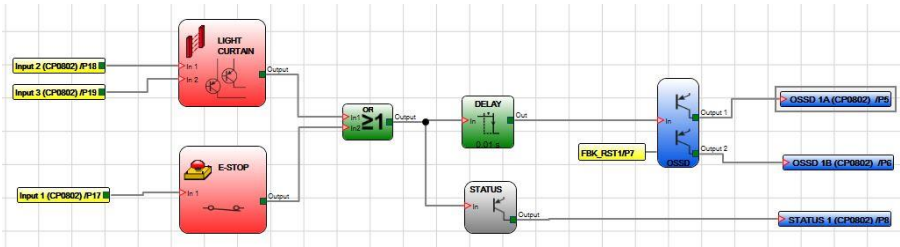
Figur 4.12. Illustrerar informations sida

4.3 SoSafe

SoSafe är ett användarvänligt program för Windows som används för att ställa in Schneider Electrics säkerhetsstyrenheter som är

konfigurerbara. Programvaran innehåller olika funktioner som användaren själv bestämmer upplägget för.

SoSafe programmeras i språket CFC och användes för att programmera säkerhetsstyrenheten. För att kunna ansluta till säkerhetsstyrenheten krävdes ett lösenord, lösenordet är “SAFEPASS” med versaler. När programmet skrevs specificerades det vilka ingångar samt utgångar som ska användas på säkerhetsstyrenheten.



Figur 4.13 En bild på hur projektet på SoSafe ser ut

4.4 Vanligt förekommande mjukvaruproblem

Det händer att LMC078 överbelastas och då bryts anslutningen mellan datorn och styrenheten vilket i sin tur kräver en mekanisk omstart för att kunna återuppta anslutningen.

IP adressen på datorn måste ändras manuellt för att styrenheten inte har DHCP (Dynamic Host Configuration Protocol) för att tilldela rätt IP. Även Sercos tappar adressen till och från och detta löses genom att SoMachine används för att göra en Sercos avsökning som tilldelar rätt adress till varje enhet. Det kan hända att nedskalningen på motorns växel glöms bort vilket måste ställas in igen.

Det händer att motorerna tappar sina positioner och då krävs det att positionerna ställs om med hjälp av SoMove.

Ibland kan det uppstå fel på Sercos enheterna som kräver en mekanisk omstart innan enheterna fungerar normalt.

När motorerna skulle jogga var det tänkt att knappen skulle hållas nedtryckt på surfplattan för att köras och när den släpptes så skulle den stoppas. Surfplattan klarade inte av att köras på detta vis utan problemet löstes med en knapp som bytte färg vid nedtryckning som sedan återställdes vid nästa nedtryckning.

5. Slutsats

I detta kapitel redogörs egna erfarenheter och slutsatser.

5.1 TVDA

Elskåpets konstruktion byggde på TVDA som användes under projektet främst före och under konstruktionen av elskåpet, den har varit väldigt användbar eftersom den innehöll information om varje moment som utfördes. Framförallt på grund av att den sparade en massa tid när tiden som fanns var väldigt begränsad.

Första steget var att välja ut vilka produkter som skulle användas till projektet. Därefter gjordes ändringar i elritningen på grund av att den innehöll komponenter som inte skulle användas exempelvis frekvensomvandlare. Slutligen används elritningen för att konstruera elskåpet. Konstruktionen av elskåpet gick bättre än förväntat, redan på första försöket lyckades elskåpet driftsättas.

En stor fördel med att använda TVDA är att den innehåller manualer för produkterna vilket leder till att det förenklas avsevärt att hitta dem. Senare vid programmering användes det TVDA mallar som fanns tillgängliga på SoMachine exempelvis mall för Power Meter iEM3155.

För personer med elbaskunskaper kan TVDA rekommenderas ifall de önskar att bygga ett elskåp. Den är användarvänlig och strukturen är enkel att följa och innehåller den allra viktigaste information som krävs för att bygga ett elskåp.

5.2 Testkörning av robot

Eftersom det var en gammal robot byttes de gamla motorerna ut till nya motorer som tillverkas av Schneider Electric. Motorerna fick provköras efter montering ifall den gamla roboten skulle fungera med dessa, vilket den gjorde.

Vid automatisk körning utfördes en plocka-och-släpp-rörelsen som planerat. Programmet håller reda på vad som skall utföras. På grund av detta klarar roboten manipuleringar inom utsatta mjukvarugränserna. På manuell körning var det önskat av Schneider Electric att kunna jogga motorerna med olika rörelser på roboten och detta lyckades med hjälp av användargränssnittet där motorerna styrs.

Gränssnittet var användarvänligt men hade sina begränsningar rent designmässigt då det t.ex. inte går att skräddarsy knapparna.

Under användargränssnittet går det att styra motorn i manuell eller automatisk mod och det går att läsa ut information från skåpet. Den kan visa energiförbrukningen eller felmeddelande på ett enkelt och tydligt sätt. Applikationen var väldigt användbar för styrning av robot.

5.3 Framtida arbete

Alla mål under projektets gång kunde inte uppnås på grund av olika anledningar, främst på grund av tidsbrist.

Roboten saknar förmågan att plocka upp föremål på grund av att elektromagneten inte har monterats på z-axeln. Ljusridån samt externa nödstopp till roboten är konfigurerade men inte färdigmonterade för användning. För att kunna köra motorn under testkörning användes 24-volt direkt istället för att ljusridån och nödstoppet skulle styra detta.

Ett av problemen som upptäcktes under testkörningen var att energimätaren fungerade till och från. Det handlade om tillfälligheter och problemet har ännu inte kunnat rättas till.

För framtida arbeten kan dessa problem åtgärdas. Möjligheten för vidareutveckling finns även eftersom roboten kan arbeta på ett 3D plan vilket leder till att roboten klarar av mera avancerade rörelser än enbart en rörelse som plockar och släpper. Det är exempelvis genomförbart att skapa spelet tre i rad där två personer möter varandra genom att använda WebVisualization som finns tillgänglig på SoMachine. Med hjälp av lite kreativitet kan det programmeras en hel del på roboten där gränsen dras av personen som styr roboten.

Dessutom kan elskåpet även vidareutvecklas genom att tillägga flera produkter som önskas av användaren. Det ser ljust ut för framtida utveckling med alla dessa möjligheter.

Terminologi

TVDA -Tested, Validated, Documented Architectures

HMI - Human Machine Interface

I/O - Input/output

WebVisualization - Ett grafiskt utvecklingsprogram som är inbyggd i SoMachine.

IL: Instruction List

LD - Ladder Diagram

ST - Structured text

FBD - Function Block Diagram

SFC -Sequential Function Chart

CFC - Continuous Function Chart

TCP - Transmission Control Protocol

IP -Internet Protocol

STO - Safe torque off

IEC - International Electrotechnical Commission

MAC - Media Access Control

DHCP - Dynamic Host Configuration Protocol

Referenser

Schneider Electric (2009). *Logo Schneider Electric*. http://www.schneider-electric.at/images/pictures/press-releases/2009_logo-schneider-electric.jpg (hämtad 28/5 2015)

Schneider Electric, 2014, *Modicon LMC078 Motion Controller Programming Guide*. Schneider Electric http://download.schneider-electric.com/files?p_File_Id=689586324&p_File_Name=EIO0000001909.00.pdf (hämtad 28/5 2015)

Schneider Electric, 2014, *LXM32S AC servo drive Product manual*. Schneider Electric http://download.schneider-electric.com/files?L=en&p=2302&p_docId=&p_docId=&p_Reference=0198441114060-EN&p_EnDocType=Userguide&p_File_Id=681841664&p_File_Name=LXM32S_Manual_V100_EN.pdf (hämtad 28/5 2015)

Schneider Electric, 2014, *XPSMCMCP0802x Instruction Sheet (Original Language).* Schneider Electric http://download.schneider-electric.com/files?p_File_Id=683695594&p_File_Name=EAV8285801.00.pdf (hämtad 28/5 2015)

Schneider Electric, 2014, *XPSMCMx Fieldbus Expansion Modules Instruction Sheet (Original Language)*. Schneider Electric http://download.schneider-electric.com/files?p_File_Id=683696311&p_File_Name=EAV8283001.00.pdf (hämtad 28/5 2015)

Schneider Electric, 2014, *iEM3110 / iEM3115 / iEM3135 / iEM3155 / iEM3165*. Schneider Electric http://download.schneider-electric.com/files?p_File_Id=792028222&p_File_Name=NHA15789-00.pdf (hämtad 28/5 2015)

Schneider Electric, 2014, *PowerPact H-, J-, and L-Frame Circuit Breakers*. Schneider Electric http://download.schneider-electric.com/files?p_File_Id=839996801&p_File_Name=0611CT1001-net.pdf (hämtad 28/5 2015)

Schneider Electric, *2014, ABL8RPS24100CE / ABL8WPS24p00CE.* Schneider Electric http://download.schneider-electric.com/files?L=en&p=&p_docId=&p_docId=&p_Reference=NHA34649&p_EnDocType=Instructionsheet&p_File_Id=687549060&p_File_Name=NHA34649_00.pdf (hämtad 28/5 2015)

Schneider Electric, 2014, *Modicon TM5 Sercos III Interface Hardware Guide*. Schneider Electric http://download.schneider-electric.com/files?p_File_Id=689586528&p_File_Name=EIO0000001941.00.pdf (hämtad 28/5 2015)

Schneider Electric, *2014, TM5SPS.* Schneider Electric http://download.schneider-electric.com/files?p_File_Id=697740159&p_File_Name=S1A12569_04.pdf (hämtad 28/5 2015)

Bilaga 1 – Koden som kör automatiska rörelsen

```
        xCmdPwrMotorX:=TRUE;
        xCmdPwrMotorY:=TRUE;
        xCmdPwrMotorZ:=TRUE;

CASE diState OF
        1:
                diOldDiState:=1;
                lrSetPosMovAbsMotorX :=0;
                xCmdMovAbsMotorX:=TRUE;

                lrSetPosMovAbsMotorY :=10;
                xCmdMovAbsMotorY:=TRUE;

                lrSetPosMovAbsMotorZ:=10;
                xCmdMovAbsMotorZ:=TRUE;

                IF fbSM3MoveAbsMotorX.Done AND
fbSM3MoveAbsMotorY.Done AND fbSM3MoveAbsMotorZ.Done
THEN
                                xCmdMovAbsMotorX:=FALSE;
                                xCmdMovAbsMotorY:=FALSE;
                                xCmdMovAbsMotorZ:=FALSE;
                                diState:=5;
                END_IF
        5:
        //Reset innan vi kör igång
                diOldDiState:=5;
                xCmdMovAbsMotorX:=FALSE;
                xCmdMovAbsMotorY:=FALSE;
                xCmdMovAbsMotorZ:=FALSE;
```

```
            diState:=10;
        10:
            diOldDiState:=10;
            lrSetPosMovAbsMotorZ:=110;
            xCmdMovAbsMotorZ:=TRUE;

            IF fbSM3MoveAbsMotorZ.Done THEN
                xCmdMovAbsMotorZ:=FALSE;
                diState:=15;
            END_IF

        15:
            diOldDiState:=15;
            lrSetPosMovAbsMotorZ:=10;
            xCmdMovAbsMotorZ:=TRUE;

            IF fbSM3MoveAbsMotorZ.Done THEN
                xCmdMovAbsMotorZ:=FALSE;
                diState:=20;
            END_IF


        20:
            diOldDiState:=20;
            lrSetPosMovAbsMotorX :=350;
            xCmdMovAbsMotorX:=TRUE;
            lrSetPosMovAbsMotorY :=160;
            xCmdMovAbsMotorY:=TRUE;

            IF fbSM3MoveAbsMotorX.Done AND
fbSM3MoveAbsMotorY.Done THEN
                        xCmdMovAbsMotorX:=FALSE;
                        xCmdMovAbsMotorY:=FALSE;
```

```
                    diState:=25;
            END_IF
        25:
            diOldDiState:=25;
            lrSetPosMovAbsMotorZ:=110;
            xCmdMovAbsMotorZ:=TRUE;
            IF fbSM3MoveAbsMotorZ.Done THEN
                xCmdMovAbsMotorZ:=FALSE;
                diState:=30;
            END_IF

        30:
            diOldDiState:=30;
            lrSetPosMovAbsMotorZ:=10;
            xCmdMovAbsMotorZ:=TRUE;

            IF fbSM3MoveAbsMotorZ.Done THEN
                xCmdMovAbsMotorZ:=FALSE;
                diState:=35;
            END_IF

        35:
            diOldDiState:=35;
            lrSetPosMovAbsMotorX :=0;
            xCmdMovAbsMotorX:=TRUE;

            lrSetPosMovAbsMotorY :=10;
            xCmdMovAbsMotorY:=TRUE;
            IF fbSM3MoveAbsMotorX.Done AND
fbSM3MoveAbsMotorY.Done THEN
                    xCmdMovAbsMotorX:=FALSE;
                    xCmdMovAbsMotorY:=FALSE;
                    diState:=5;
```

```
            END_IF
        999:
            xCmdMovAbsMotorX:=FALSE;
            xCmdMovAbsMotorY:=FALSE;
            xCmdMovAbsMotorZ:=FALSE;

        IF      xMachineStandstill AND xActAuto THEN
            diState := diOldDiState;
        END_IF
END_CASE
```

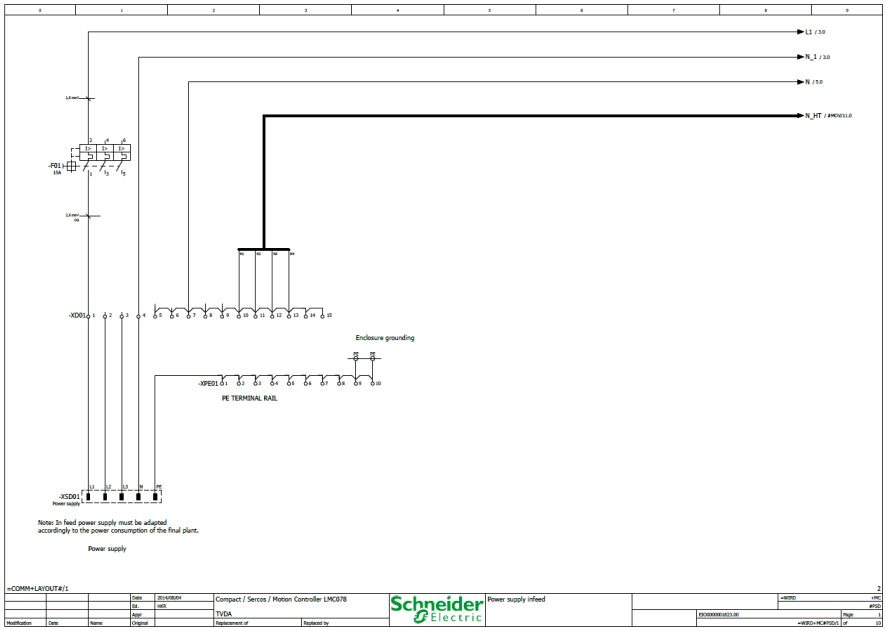
Bilaga 2 – Koden som behandlar fel

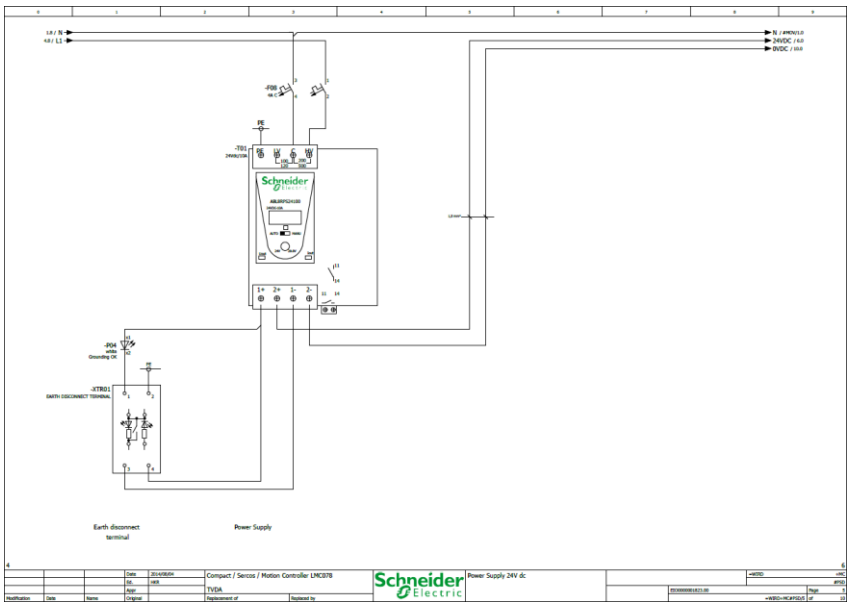
```
fbSM3ReadErrMotorX(     Axis := SM_MotorX,
                        Enable:= TRUE,
                        AxisError => xErrMotorX,
                        AxisErrorID => dwErrIdMotorX);

fbSM3ReadErrMotorY(     Axis := SM_MotorY,
                        Enable:= TRUE,
                        AxisError => xErrMotorY,
                        AxisErrorID => dwErrIdMotorY);

fbSM3ReadErrMotorZ(     Axis := SM_MotorZ,
                        Enable:= TRUE,
                        AxisError => xErrMotorZ,
                        AxisErrorID => dwErrIdMotorZ);
```

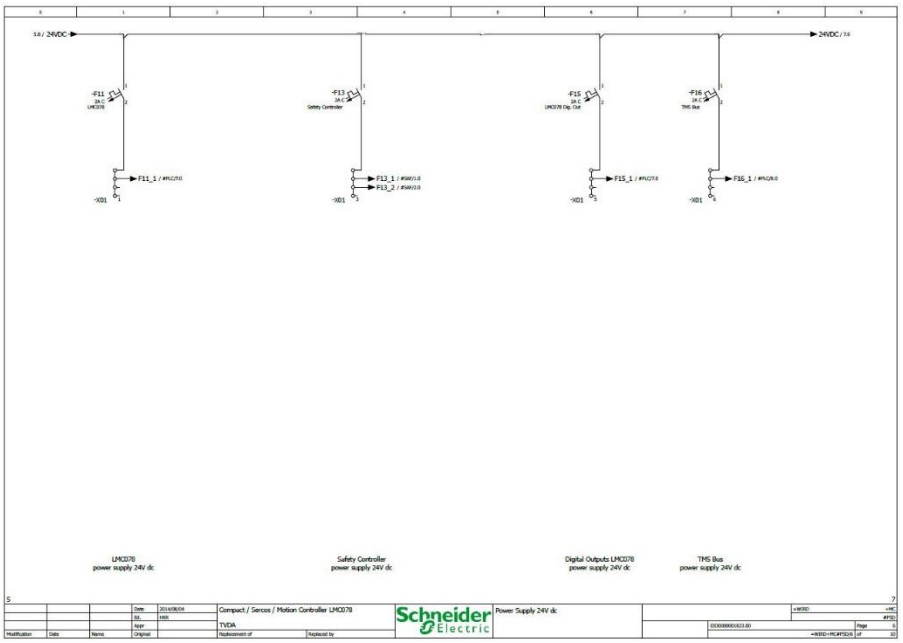
```
    IF xErrMotorX OR xErrMotorY OR xErrMotorZ THEN
fbSM3MoveAbsMotorY.Error OR fbSM3MoveAbsMotorZ.Error
THEN
            xActAuto:=FALSE;
            xActManual:=FALSE;
            xGetIntoFrame:=FALSE;
    END_IF
// Should be the opposit. Decides the visibility of the Error window.
    IF xMachineErr OR SercosIII.State <> 4 OR xErrMotorX OR
xErrMotorY OR xErrMotorZ THEN
            xErrorMotor:=FALSE;
    ELSE
            xErrorMotor:=TRUE;
    END_IF
```

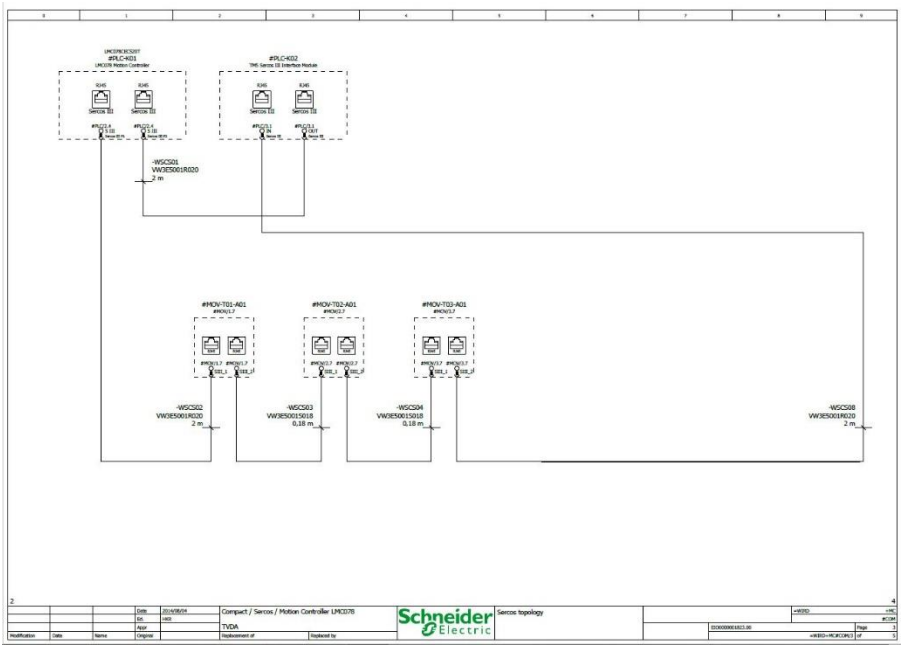
```
	IF SERCOSIII.State <> 4  THEN
		xActAuto:=FALSE;
	END_IF

	IF ( xMachineErr OR xLXM32sErrStat OR SERCOSIII.State
=11) AND xCmdRst THEN
		SEC.FC_DiagQuit(); // Blåa knappen på Somachine
		SERCOSIII.PhaseSet:=4;
	END_IF
	IF NOT xGetIntoFrame THEN
		IF	lrActPosMotorX < lrMinXAxis OR
			lrActPosMotorX >lrMaxXAxis OR
			lrActPosMotorY < lrMinYAxis OR
			lrActPosMotorY >lrMaxYAxis OR
			lrActPosMotorZ < lrMinZAxis OR
			lrActPosMotorZ >lrMaxZAxis THEN
				xCmdStop:=TRUE;
				xIntoFrame:=FALSE;
		IF xMachineStandstill OR xMachinePwrOff THEN
				xActManual:=FALSE;
				xCmdStop := FALSE;
				xExitModes:=TRUE;
			END_IF
		ELSE
				xIntoFrame:=TRUE;
		END_IF
	END_IF
```

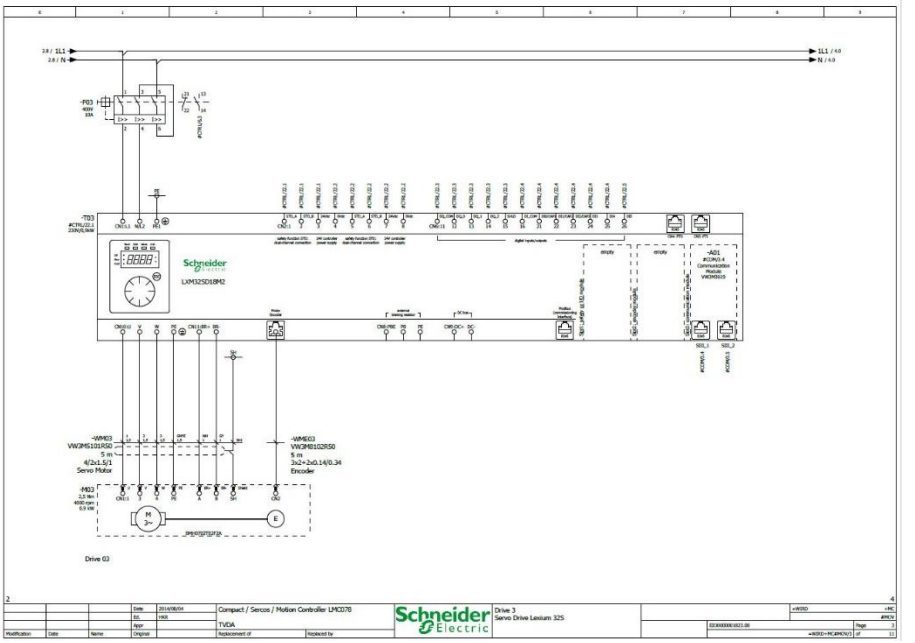
Bilaga 3 - Exempel på elritningar

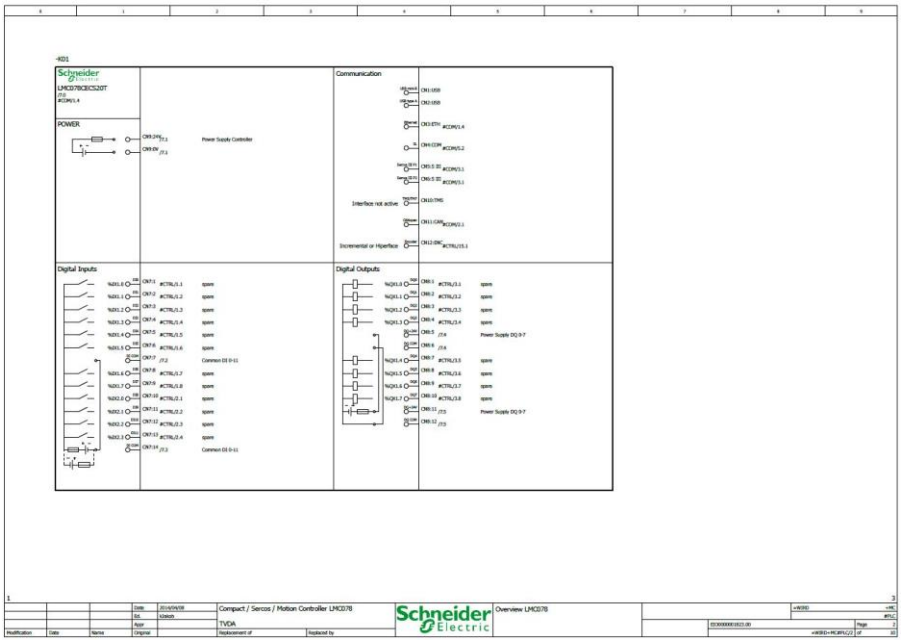





24VDC / 3.0
-F11
2A C
LMC078
-F13
2A C
Safety Controller
-F15
2A C
LMC078 Dig. Out
-F16
2A C
TMS Bus
-X01
F11_1 / #PLC7.0
F13_1 / #SAFI1.0
F13_2 / #SAFI2.0
F15_1 / #PLC7.0
F16_1 / #PLC8.0
LMC078
power supply 24V dc
Safety Controller
power supply 24V dc
Digital Outputs LMC078
power supply 24V dc
TMS Bus
power supply 24V dc
Compact / Sercos / Motion Controller LMC078
TVDA
Power Supply 24V dc
Schneider Electric


#PLC-A01
LMC078CBCS20T
LMC078 Motion Controller
#PLC-A02
TMS Sercos III Interface Module
-WSCS01
VW3E5001R020
2 m
#MOV-T01-A01
#MOV-T02-A01
#MOV-T03-A01
-WSCS02
VW3E5001R020
2 m
-WSCS03
VW3E5001S018
0,18 m
-WSCS04
VW3E5001S018
0,18 m
-WSCS08
VW3E5001R020
2 m
Compact / Sercos / Motion Controller LMC078
TVDA
Sercos topology
Schneider Electric


LXM32SD18M2
Drive 03
Compact / Sercos / Motion Controller LMC078
Drive 3
Servo Drive Lexium 32S
TVDA
Schneider Electric


-KD1
LMC078CBC520T
POWER
Communication
Digital Inputs
Digital Outputs
Compact / Sercos / Motion Controller LMC078
Overview LMC078
TVDA
Schneider Electric

Cable overview

Cable designation	from	to	cable type	Conductors all / used / free (shield)	ø	Length	Remark
+MC-WCAN_01	+MC#PLC-K01	+MC#COM-XCA01	TSXCANCBxx	2x2 / 0 / 4 (1)	24/22 mm²	1 m	
+MC-WCAN_F09	+MC#MOV-F09	+MC#COM-XCA01	TCSCCN4F3M1T	/ 0 / 0		1 m	
+MC-WCAN_T07	+MC#MOV-T07-A01	+MC#COM-XCA01	TCSCCN4F3M1T	/ 0 / 0		1 m	
+MC-WCAN_T08	+MC#MOV-T07-A01	+MC#MOV-T08-A01	VW3CANCARR10	/ 0 / 0		1 m	
+MC-WCAN_T10	+MC#MOV-T10-CN17	+MC#COM-XCA01	TSXCANCBxx	2x2 / 0 / 4 (1)	24/22 mm²	1 m	
+MC-WE01	+MC-XD01	+MC#PSD-E01	OELFLEX® classic 110 G	3 / 3 / 0	1,5 mm²	5 m	Cabinet light
+MC-WETH1	+MC#PLC-K01	+MC#COM-A01	490NTW00002	/ 0 / 0		2 m	Ethernet cable
+MC-WETH2	+MC#HMI-P01	+MC#COM-A01	490NTW00002	/ 0 / 0		2 m	Ethernet cable
+MC-WETH3	+MC-XSETH01	+MC#COM-A01	490NTW00002	/ 0 / 0		2 m	Ethernet cable
+MC-WM1	+MC-XD01	+MC#PSD-M1	OELFLEX® classic 110 G	3 / 3 / 0	1,5 mm²	4 m	Enclosure fan
+MC-WM01	+MC#MOV-T01	+MC#MOV-M01	VW3M5101R50	4/2 / 6 / 0 (1)	1.5/1 mm²	5 m	Servo Motor
+MC-WM02	+MC#MOV-T02	+MC#MOV-M02	VW3M5101R50	4/2 / 6 / 0 (1)	1.5/1 mm²	5 m	Servo Motor
+MC-WM03	+MC#MOV-T03	+MC#MOV-M03	VW3M5101R50	4/2 / 6 / 0 (1)	1.5/1 mm²	5 m	Servo Motor
+MC-WM07	+MC#MOV-T07	+MC#MOV-M07	OELFLEX® classic 110 G	4 / 4 / 0	1,5 mm²	5 m	Motor cable Drive 07
+MC-WM08	+MC#MOV-T08	+MC#MOV-M08	OELFLEX® classic 110 G	4 / 4 / 0	1,5 mm²	5 m	Motor cable Drive 08
+MC-WM09	+MC#MOV-F09	+MC#MOV-M09	OELFLEX® classic 110 G	4 / 4 / 0	1,5 mm²	5 m	Motorcable Drive 9
+MC-WMB01	+MC#PSD-P05	+MC#PLC-K01	VW3A8306D30	/ 0 / 0		3 m	Modbus cable / pre-configured
+MC-WME01	+MC#MOV-M01	+MC#MOV-T01	VW3M8102R50	3x2+2 / 0 / 0	0.14/0.34 mm²	5 m	Encoder
+MC-WME02	+MC#MOV-M02	+MC#MOV-T02	VW3M8102R50	3x2+2 / 0 / 0	0.14/0.34 mm²	5 m	Encoder
+MC-WME03	+MC#MOV-M03	+MC#MOV-T03	VW3M8102R50	3x2+2 / 0 / 0	0.14/0.34 mm²	5 m	Encoder
+MC-WP01	+MC-X01	+MC#CTRL-P01	OELFLEX® classic 110 G	7 / 6 / 1	1 mm²	3 m	Towerlight
+MC-WSCS01	+MC#PLC-K01	+MC#PLC-K02	VW3E5001R020	/ 0 / 0		2 m	Sercos patch cable
+MC-WSCS02	+MC#PLC-K01	+MC#MOV-T01-A01	VW3E5001R020	/ 0 / 0		2 m	Sercos patch cable
+MC-WSCS03	+MC#MOV-T01-A01	+MC#MOV-T02-A01	VW3E5001S018	/ 0 / 0		0,18 m	Sercos patch cable
+MC-WSCS04	+MC#MOV-T02-A01	+MC#MOV-T03-A01	VW3E5001S018	/ 0 / 0		0,18 m	Sercos patch cable
+MC-WSCS08	+MC#PLC-K02	+MC#MOV-T06-A01	VW3E5001R020	/ 0 / 0		2 m	Sercos patch cable
+MC-WT10.1	+MC#MOV-F10	+MC#MOV-T10-CN1	ÖLFLEX CLASSIC 100	3 / 3 / 0	1,5 mm²	5 m	Motorcable Drive 10
+MC-WT10.2	+MC-X01	+MC#MOV-T10-CN15	VW3M9405	/ 3 / 0		5 m	STO-cable Drive 10
+MC-WU01	+MC-X01	+MC#CTRL-U01-S01	ÖLFLEX CLASSIC 110-G	5 / 5 / 0	1 mm²	1 m	Field Buttons
+LOC-WB01	+MC#PLC-K01	+LOC#CTRL-B01	VW3E2097R050	8 / 1 / 8		5 m	Encoder cable LMC078 incremental
+LOC-WBUSK10	+MC#PLC-K10	+LOC#PLC-K01	TCSXCN2FNX10E	2x2 / 0 / 4	0.34/0.2 mm²	10 m	TM7 Bus cable
+LOC-WPWRK10	+MC-X01	+LOC#PLC-K01	TCSXCNEFNX10V	4 / 4 / 0	0,34 mm²	10 m	TM7 Power cable