# Utveckling av portalrobot

DOROTEA JONASON OCH MARTIN NORÉN
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY |
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

LUNDS UNIVERSITET
Lunds Tekniska Högskola

Schneider Electric

# Examensarbete

# Utveckling av portalrobot

Av
Dorotea Jonason och Martin Norén

Department of Electrical and Information Technology

Faculty of Engineering, LTH, Lund University

SE-221 00 Lund, Sweden

# Sammanfattning

Det här examensarbetet omfattar en uppdatering av hård- och mjukvara för en befintlig kartesisk portalrobot med tillhörande elskåp. Arbetet fokuseras huvudsakligen på att programmera en ny Programmable Logic Controller (PLC) för styrning och körning av roboten. För att möjliggöra körning i flera olika driftlägen implementeras ett nytt robotbibliotek. Examensarbetet undersöker även hur ett Human Machine Interface (HMI) för styrning av roboten kan implementeras för att uppnå både funktionalitet och användarvänlighet.

En uppdatering av konstruktionen i elskåpet utfördes också för att anpassa hårdvaran till den nya mjukvaran. Det utfördes genom att plocka bort de komponenter som inte längre fyllde någon funktion samt att byta ut den föråldrade utrustningen till den aktuella motsvarigheten.

Examensarbetet gjordes i samarbete med Schneider Electric i Lund där portalroboten var placerad och all hård- och mjukvara som användes i examensarbetet är därför Schneider Electrics egna produkter.

Nyckelord: Kartesisk robot, PLC, HMI, Robotbibliotek

# Abstract

This Bachelor thesis covers an update on hard- and software for an existent cartesian robot. The work is primarily focused on the programming of a Programmable Logic Controller (PLC) for control and run of the robot. To enable different run modes an implementation of a new robotic library is established. This thesis also explores how a Human Machine Interface (HMI) can be implemented to achieve both functionality and user-friendly-environment.

An update of the construction in the electrical cabinet was executed in order to match the hardware to the new software. This was executed by removing the components that no longer fulfilled any function and the old equipment was replaced with the current equivalent.

The Bachelor thesis was made in cooperation with Schneider Electric in Lund where the cartesian robot was based and therefore all the hard- and software that were used in this thesis are products owned by Schneider Electric.

Keywords: Cartesian Robot, PLC, HMI, Robotic library

# Innehållsförteckning

# Förord

Detta projekt har varit otroligt lärorikt och vi är väldigt tacksamma för att ha fått möjligheten att få genomföra examensarbetet under en tid så präglad av covid-19.

Vi har blivit väl omhändertagna av Schneider Electric och vi vill tacka alla som hjälpt oss, från August i receptionen till våra fantastiska handledare.

Tack till Lars Celano som utbildat och erbjudit oss sin expertis inom HMI.

Tack till Josef Christoffersson som försett oss med all möjlig hårdvara.

Stort tack till Johan Hult som var vecka hjälpt oss med allt mellan himmel och jord.

Och framförallt stort tack till Thomas Karlsson, Sveriges bästa programmerare inom strukturerad text, som tillsammans med oss suttit flera sena kvällar för att hjälpa oss genom projektet.

Vi vill även tacka vår handledare Mats Lilja och Examinator Christian Nyberg för deras arbete med examensarbetet och deras undervisning under vår tid på LTH, Campus Helsingborg.

Dorotea Jonason & Martin Norén

# 1. Inledning

Det första och inledande avsnittet beskriver bakgrunden för examensarbetet samt syfte, mål- och problemformulering och avgränsningar.

## 1.1 Bakgrund

Schneider Electric är en europeisk elektronikkoncern som är verksam över hela världen. De förser kunderna med lösningar, inom energi och automation, med mål att effektivisera och öka hållbarheten. Företaget riktar sig mot exempelvis vanliga hushåll men även större fastigheter och infrastrukturer.

Portalroboten med tillhörande elskåp, vars mjukvara och hårdvara ska uppdateras, är från början framtagen av två studenter från LTH i ett examensarbete på Schneider Electric 2015. Därefter har roboten genomgått olika iterationer med olika funktioner som lagts till och plockats bort vilket resulterat i en ostrukturerad grund med komponenter installerade som ej används.

Huvudproblemet som examensarbetet ska lösa är att portalroboten i dagsläget inte är uppdaterad till företagets senaste hård- och mjukvara. Då företaget genomför ett generationsbyte av sina egenutvecklade komponenter skall roboten uppdateras för att reflektera och demonstrera den nya tekniken. Problemet ska lösas genom att koppla in en ny PLC och även en touch-display med tillhörande PC som skall styra roboten och visa data.

Touch-displayen skall styras av ett HMI programmerat i “Operator Terminal Expert”. Denna mjukvara är utvecklad av Schneider Electric. HMI står för Human Machine Interface och gör kommunikation med en maskin möjlig genom ett grafiskt gränssnitt.

Roboten är inte uppkopplad mot molnet vilket är en målsättning från företaget. Detta skall lösas med Schneider Electrics molnverktyg ”Machine Advisor” där en digital tvilling skall skapas. Detta är en kopia av verkligheten som byggs upp i ett virtuellt verktyg. I detta fall kommer tvillingen att vara en identisk robot som byggs upp i Machine Advisor som kommer att nyttjas för att simulera och analysera roboten virtuellt.

Ett exempel på hur detta kan användas är att testa den digitala roboten under kritiska tillstånd för att analysera hur komponenterna blir påverkade utan att skada den verkliga fysiska roboten.

Portalrobotens drift kommer programmeras i Schneider Electrics mjukvara “Machine Expert”. Programmet innehåller ett nytt robotbibliotek vilket uppdaterats för att vara kompatibelt med företagets mindre modeller utav PLC som inte fanns när portalroboten byggdes 2015. Det gamla robotbiblioteket är inte kompatibelt med det nya så all drift och styrning skall programmeras på nytt.

All form av hårdvara och mjukvara kommer att tillhandahållas av Schneider Electric.

## 1.2 Syfte

Syftet med examensarbetet är att uppdatera den nuvarande portalroboten för att bättre demonstrera företagets mjukvara och hårdvara. Portalroboten förväntas att kunna styras av operatör via ett grafiskt gränssnitt, demonstrera olika typer av drift och kopplas upp till ett molnverktyg för att skapa en digital tvilling.

## 1.3 Målformulering

Examensarbetet ska utföras så att portalroboten uppdateras och kan styras via Schneider Electrics erhållna nya mjuk- och hårdvara.

Ett HMI ska programmeras som demonstrerar data och drift. Styrningen skall demonstrera referenskörning, joggning och olika rörelsemönster såsom cam-rörelser.

Programmeringen av roboten skall använda företagets uppdaterade robotbibliotek.

Molnverktyget som skall användas är “Machine Advisor” och skall användas för att skapa en digital tvilling. Därefter skall data från tvillingen analyseras och presenteras med hjälp av programmet.

I mån av tid skall elkonstruktionen ses över och dokumenteras.

## 1.4 Problemformulering

Examensarbetet kommer omfatta programmering av hårdvara och dessa följande problemfrågor ska besvaras:

1. Kan det nya robotbiblioteket implementeras på portalroboten?
2. Kan ett HMI implementeras så att roboten kan styras via detta?
3. Är det möjligt att köra roboten i flera olika driftlägen?
4. Är det möjligt att koppla upp roboten och skapa en digital tvilling i molnverktyget Machine Advisor?
5. Vilka värden kan avläsas från den digitala tvillingen?

## 1.5 Motivering av examensarbetet

Intresset för automationsteknik och viljan att arbeta inom området bidrog till valet av examensarbetet. Både mjukvara och hårdvara ska utvecklas och förbättras, vilket bidrar till en stor variation i arbetet. Ett befintligt styrsystem finns redan och mer fokus kan då läggas på vidareutveckling och förbättra systemet istället för att lägga tid på att starta upp ett helt nytt projekt.

Tanken är också att en uppdatering av både mjuk- och hårdvara ska bidra till att företaget kan använda och nyttja styrsystemet i framtiden, vilket tillför att det här examensarbetet känns extra viktigt.

## 1.7 Avgränsningar

Det grafiska gränssnittet (HMI) skall fokusera på funktion och mindre arbete skall ligga på den grafiska designen.

Endast mjukvara och hårdvara som tilldelas av Schneider Electric kommer att användas.

Portalroboten skall endast demonstrera drift och inte utföra arbete.

Arbetet kommer endast omfatta funktioner som är möjliga att implementera på den befintliga portalroboten.

## 1.8 Resurser

- Portalrobot
- HMI (Operator terminal expert)
- Ny hårdvara
- Mjukvara
- Datorer
- Molnverktyget Machine Advisor
- Lokal för arbete med inkoppling och tillgång till elkonstruktion av portalroboten.
- Utbildning inom mjuk- och hårdvara.

# 2. Teknisk bakgrund

Följande kapitel behandlar den huvudsakliga tekniska bakgrunden för den hårdvara och mjukvara som möjliggjort examensarbetet. Även kartesisk robot, styrsystem och HMI definieras.

## 2.1 Kartesisk robot

En kartesisk robot är en linjärrobot, också kallad kartesisk koordinatrobot, med tre ortogonala styraxlar. De tre styraxlarna (X, Y, Z) definierar längs med vilka riktningar som förflyttning kan ske, vilket bidrar till att den kartesiska roboten utför en koordinerad rörelse. Med hjälp av de tre axlarna kan roboten nå tredimensionella koordinater.  Varje axel är kopplad till en egen styrmotor som i sin tur är ansluten till en servoenhet. [1]

## 2.2 Styrsystem

Styrsystemet för roboten är en PLC med huvudsyftet att styra de tre axlarna på roboten och arbeta mot de anslutna komponenterna i systemet. PLC står för programmable logic controller som på svenska översätts till programmerbart styrsystem och används mestadels inom industriell automation med syfte att styra exempelvis industriella maskiner och motorer. Ett flertal olika programmeringsspråk finns tillgängliga för att programmera en PLC och av den anledningen har de vanligaste språken standardiserats i IEC-61131-3[2]. De fem standardiserade språken är;

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

Programmering av PLCn sker i Schneider Electrics egen mjukvara, EcoStruxure Machine Expert och programmeringsspråken som erbjuds i programmet är alla de fem standardiserade språken.

## 2.3 HMI

HMI är en förkortning av engelskans Human Machine Interface vilket på svenska kan översättas till Människa-Maskin Gränssnitt. HMI är skapat för att möjliggöra och realisera kommunikation mellan människa och maskin med hjälp av ett grafiskt gränssnitt. De grafiska gränssnitten visar information och data läsbart för människan samtidigt som det gör styrning av maskinen från HMI möjligt. Den grafiska designen samt programmering av HMI går att utföra med olika program och mjukvaror, exempelvis Schneider Electrics egna program Operator Terminal Expert.

## 2.4 Hårdvara

Följande segment kommer behandla hårdvaran som använts i projektet, PLCn Modicon M262, Servodrivenheterna tillsammans med servomotorerna och säkerhetsmodulerna Modicon Preventa. Fler komponenter har testats men ej varit kvar i den slutgiltiga konstruktionen.

Exempel på hårdvara som behandlar användarstyrning är RFID-systemet Osisense XG, Harmony hub ZBRN1 och HMI-styrning via Magelis pekskärm med industri-PC.

### 2.4.1 Modicon M262

PLCn som har använts i det här examensarbetet är Schneider Electrics PLC av modellen Modicon M262 motion controller. Modellen av PLC har en inbäddad sammanlänkning till molntjänster som ansvarar för säkerheten på nätet. M262 är utrustad med två enskilda Ethernet-nätverk där båda nätverken är försedda med stöd för Ethernet/IP och Modbus TCP. Dessa protokoll är ofta förekommande i industriell automationsteknik. Stöd för Sercos III finns i det första Ethernet-nätverket. PLCn är även utrustad med en seriell lina och 8 stycken I/O:s i hög hastighet, 4 digitala ingångar samt 4 digitala utgångar. En USB mini-B finns också tillgänglig med syftet att programmera mot.

PLC av modellen Modicon M262 erbjuder ett modulärt expansionssystem som ger en möjlighet att utöka med flera olika kompatibla moduler i direkt anslutning mot hårdvaran utan kablar. [3][4]

### 2.4.2 Servodrivenhet & AC servomotor

En servodrivenhet tillsammans med en sammankopplad servomotor möjliggör elektronisk rörelse med precision. Uppgiften hos servodrivenhet är att kalkylera rätt frekvens, spänning och ström som läggs på servomotorn för att ha en identisk rörelse under varierande arbete. Servodrivenheten är direkt ansluten till en PLC där programmering av önskade rörelser genomförs vilket slutligen når motorerna som utför den programmerade rörelsen. I det här examensarbetet används tre identiska servodrivenheter, Lexium 32S, som var och en styr en axel. Servomotorerna är av modell AC servomotor BMH0701P07A2A för X och Y-axlarna och BMH0701P07F2A för Z-axeln. Modell BMH0701P07F2A har ett inbyggt mekaniskt stopp vilket agerar som säkerhetsfunktion ifall Z-axeln, som utför arbete, inte ska röra sig i ett scenario då motorn blir spänningslös under arbetande drift. [5][6]

### 2.4.3 Modicon Preventa Safety Modules

Schneider Electric utvecklar säkerhetssystemet Modular safety controller Modicon MCM, tidigare benämnt Preventa safety modules. Preventa safety modules är det aktuella säkerhetssystemet för examensarbetet och programmeras i en Schneider Electric säkerhetsmjukvara kallad SoSafe. Modulerna behandlar nödstopp från de installerade säkerhetsrelä i form av nödstoppsknapp och ljusridå. Detta system har varit förinstallerat, förprogrammerat och kommer inte att fördjupas under rapporten.

### 2.4.4 RFID-system

Förkortningen RFID står för Radio Frequency Identification och är ett identifikationssystem som är vanligt förekommande i samhället. Märket på RFID-systemet i examensarbetet är Schneider Electrics Telemecanique sensor. Systemet består av antingen en läsare eller smart antenn som båda har tillhörande taggar eller kort med identifikationen för objektet vilket den representerar. Kombinationen utav en läsare eller smart antenn med taggar och kort utgör hårdvaran för RFID-systemet. Hur den ska uppträda programmeras i mjukvara med stöd för RFID. I examensarbetet används ett system som är uppbyggt av en smart antenn av modell Osisense XG med tillhörande kort och taggar. [7]

### 2.4.5 Harmony Hub (ZBRN1)

Samlingsnamnet Harmony omfattar bland annat tråd- och batterilösa apparater där mottagare kommunicerar via en extern antenn med trådlösa tryckknappar som genererar en spänning vid nedtryck. Mottagaren kommunicerar med PLCn via Modbus TCP-kommunikation, vilket leder till att variabler i programkoden kan kopplas till de trådlösa tryckknapparna.

Modellen som är aktuell för examensarbetet är den trådlösa mottagaren ZBRN1. Det maximala avståndet för kommunikation mellan mottagaren och tryckknapparna, med en Harmony ZBRN1, är 100 m, med en extern antenn mellan signalen och mottagaren ökas det maximala avståndet till 250 m.[8]

### 2.4.6 Acti 9 och iEM3000 energimätare

Acti 9 är ett kommunikationssystem kompatibelt med Modbus med stöd för energimätning. Energimätaren iEM 3055 är fullt kompatibel med Acti 9 och tillsammans ger de detaljerad information angående energianvändning och total förbrukning. Aktiv och reaktiv effekt kan läsas ut då energimätaren använder sig av 4-kvadrantsmätning. [9][10]

### 2.4.7 Magelis- pekskärm och industri-PC

En industri-PC, förkortat iPC, är en dator som används inom industrin för styrning och datahantering. Till skillnad från en vanlig PC har en iPC högre krav att fungera felfritt utan underhåll i mer utsatta miljöer som kan förekomma i industrin. Till en industri-PC kan en pekskärm installeras för att visa bild och göra inmatning utan mus och tangentbord möjlig. Dessa pekskärmar är konstruerade för att vara tåliga och kunna användas med handskar vilket kan vara ett krav för säkerheten där skärmen är installerad.

Hårdvaran för HMI-styrningen är uppbyggd av det modulära systemet Magelis utvecklat av Schneider Electric. Modellen HMIDMA521 är en 22 tums LED-skärm med kapacitivt baserad multipekfunktion som monteras direkt på iPCn av modell HMIBMPSI74D280.Denna iPC är Windows 10 baserad. [11]

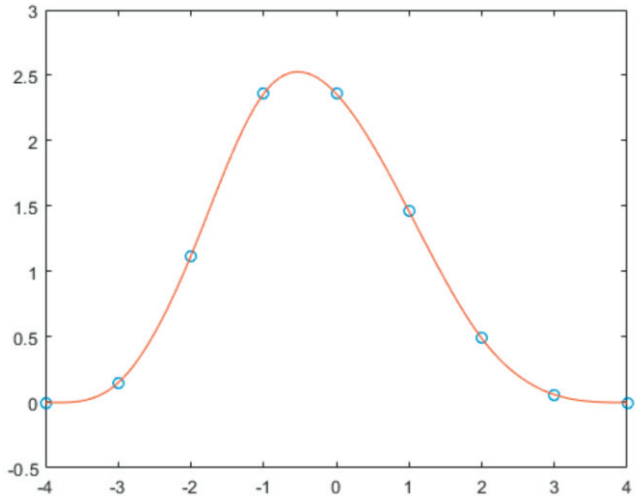
## 2.5 Sercos

Sercos står för Serial Real-time Communication System och är ett öppet och globalt digitalt gränssnitt standardiserat med IEC. Användningsområdet för Sercos är realtidskommunikation mellan I/O-system och utrustning för industriella system.

Sercos III är en version av Sercos för kommunikation med automatiserade lösningar. För användning av automatiserade system krävs enkla universella sammankopplingar, Sercos III som är ett Ethernetbaserat kommunikationssystem uppfyller dessa krav. [12]

## 2.6 Spline

Spline är en term för att beskriva en matematisk funktion som är bitvis uppbyggd av flera polynom. Termen spline refererar också till den kurvan som den matematiska funktionen bildar, exempel på en spline demonstreras i Figur 1. Flera mindre delar av funktioner som tillsammans bildar en annan funktion kallas för spline. Tangeringspunkten i kurvan kallas för apex.[13]




*Figur 1. Spline [14]*

## 2.7 Encoder

Utrustning för konvertering av position och rörelse till digitala eller analoga utsignaler kallas för encoders. Servomotorerna i examensarbetet använder sig av absolut rotationsencoders för konvertering. Dessa agerar vinkelgivare då den önskade utsignalen är positionen på axlarna. En absolut encoder behåller värdet på positionen när strömmen till den bryts. [15]

## 2.8 Robotbibliotek

Ett robotbibliotek är ett bibliotek med färdigimplementerade metoder, funktioner och strukturer med syftet att användas för programmering av hårdvara. Två bibliotek används i det här examensarbetet för robotstyrning, Schneider Electrics robotbibliotek som är kompatibelt med hårdvaran samt det standardiserade biblioteket PLCopen. [16]

## 2.5 Mjukvara

Nästföljande del förklarar mjukvaran som använts i examensarbetet vilket är EcoStruxure Machine Expert som behandlar PLC-programmering och EcoStruxure Operator Terminal Expert som behandlar HMI-programmering.

### 2.5.1 EcoStruxure Machine Expert

EcoStruxure Machine Expert är Schneider Electrics självutvecklade mjukvara för PLC-programmering. Mjukvaran används för programmering, konfiguration och uppkoppling mot PLC för att drivas i realtid. I programmet byggs styrsystemet upp med virtuell hårdvara som speglar och synkar med den faktiska hårdvaran. Fullständiga bibliotek är tillgängliga och kan implementeras, både specifika bibliotek utvecklade av företaget och öppna bibliotek som stöds av codesys. Då Machine Expert är codesys-baserat stöds alla de fem programmeringsspråken som är standardiserade i IEC-61131-3. [17][18]

### 2.5.2 EcoStruxure Operator Terminal Expert

EcoStruxure Operator Terminal Expert är en mjukvara utvecklad av Schneider Electric för programmering av HMI. Variabler och alarm skapade i programkoden för styrsystemet går att koppla till Operator Terminal Expert via en symbollista för att styra PLC. Variablerna knyts samman med knappar, lampor och mätare i gränssnittet som sedan logik från Machine Expert hanterar. Logiken omfattar hur och vad som ska påverkas vid ändringar av data, tryck från användare etc. Operator Terminal Expert arbetar i samspel med Machine Expert vid simulering

av gränssnittet innan programmet sparas till en fil som exekveras av HMI-hårdvaran. [19]

### 2.5.3 SoMove

SoMove är ett installationsprogram med huvudsyfte att konfigurera styrenhets-utrustning. Styrenheterna vilket SoMove har stöd för är Schneider Electrics egna produkter, där styrenheten Lexium ingår. [20]

### 2.6.1 EcoStruxure Machine Advisor

EcoStruxure Machine Advisor är Schneider Electrics egna molnverktyg med molnbaserade och digitala tjänster skapade för både maskintillverkare och maskinoperatörer. [21]

# 3. Metod

I följande avsnitt redogörs de olika faserna i examensarbetet samt beskrivs hur inhämtning av information har skett. Kommande avsnitt kommer även behandla källorna för examensarbetet och varför de anses som pålitliga. Inför examensarbetets start togs beslutet att dela in arbetet i fem olika faser för att uppnå ett effektivt arbetssätt. Den första fasen ägnades åt förberedelser och planering för att starta examensarbetet medan utbildningsfasen fördelades på olika tillfällen. Detta resulterade i att utbildningen delvis ägde rum under den tredje och fjärde fasen där konstruktions- och programmeringsarbetet utfördes.

## 3.1 Förberedelser

En initialbeskrivning inför examensarbetet skrevs där den grundläggande informationen från företaget samlades, med mål att återanvända i den slutgiltiga rapporten. Därefter bokades ett uppstartsmöte in där den praktiska informationen kring examensarbetet bestämdes. Resultatet från uppstartsmötet var att examensarbetet skulle ske på plats på Schneider Electrics kontor i Lund med avstämningar varje vecka via digitala möten.

## 3.2 Utbildning

För att möjliggöra examensarbetet var det viktigt att skapa en väsentlig och grundläggande förståelse för Schneider Electrics unika mjukvaror, Ecostruxure Machine Expert och EcoStruxure Operator Terminal Expert. Utbildning gavs av två olika handledare vid två separata tillfällen och ägde rum på Schneider Electrics kontor i Lund.

Utbildningen för EcoStruxure Machine Expert genomfördes först med syftet att programmeringen av roboten skulle påbörjas i tid. Utbildningen inkluderade i huvudsak hur ett nytt projekt skapas och hur koden skrivs i de olika språken tillgängliga i programmet. Installation av mjukvara och aktivering av licenser var essentiella för att påbörja programmeringen. Därefter demonstrerades hur en förbindelse mellan hårdvara och mjukvara skulle upprättas med målsättningen att få en förståelse för hur de olika komponenterna tillhörande roboten skulle samverka med mjukvaran.

För att få ut så mycket som möjligt av HMI-utbildningen togs beslutet att genomföra den vid ett senare tillfälle när fungerande kod för sammanlänkning av variabler fanns tillgänglig. Utbildningen för EcoStruxure Operator Terminal

Expert utfördes för att få kunskap i hur ett HMI skapas och hur sammankopplingen med EcoStruxure Machine Expert fungerar.

## 3.3 Konstruktion

En befintlig portalrobot med tillhörande elskåp fanns redan på plats innan examensarbetets start. Ett beslut togs att uppgraderingen av elskåpet skulle påbörjas innan en installation av den nya roboten utfördes. Syftet med beslutet var möjligheten att kunna färdigställa elskåpet och testköra kod på den gamla roboten innan den nya installerades. Då den äldre inte skulle användas mer kunde denna utnyttjas som test för att undvika risken att skada den nya roboten.

Innan någon hårdvara byttes ut byggdes roboten upp i Machine Expert, där förbindelser med hårdvaran skapades successivt. Den hårdvara som först implementeras i Machine Expert var ZBRN2 och Sercos master med tillhörande drivenheter. I följd kopplades M262 in och implementering av den påbörjades. ZBRN2 och Sercos Master kopplades till två olika ingångar på PLCn. Arbetet fortsatte med implementering av RFID-systemet där kartläggning av ingångarna samt taggar och kort utfördes. Kartläggningen som visas i Tabell 1, genomfördes med anledning att använda dem i koden för start och stopp av roboten då ett fungerande HMI inte fanns tillgängligt förrän senare i examensarbetet.

*Tabell 1. Kartlagda ingångar i RFID-systemet*

| Tagnamn: | Tagtyp: | Tag System Flag: | UID[1] : | UID[2] : | UID[4] : |
|---|---|---|---|---|---|
| 8 | Kort | 265 | 19721 | 31252 | 2016 |
| Jump | Kort | 265 | 19465 | 31252 | 2016 |
| Pick Place | Kort | 265 | 19209 | 31252 | 2016 |
| 8 | Knapp | 257 | 18885 | 14380 | 1248 |
| Pick Place | Knapp | 257 | 51963 | 14380 | 1248 |

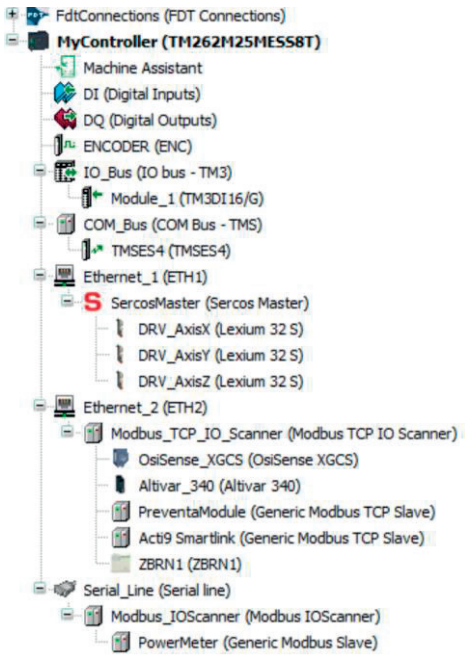
Under implementeringen av M262 skapades förbindelser med majoriteten av den resterande hårdvaran. Dessa komponenter var kommunikationssystemet Acti9, säkerhetssystemet preventa samt Altivar 340. Dessa anslöts till den andra ethernetingången på PLCn via en switch. Därefter utfördes flashning av drivenheterna Lexium 32 S via programmet SoMove då en uppdatering av firmware krävdes för att dessa skulle kunna drivas med den nyare PLC-modellen.

Nästa steg i konstruktionen var att färdigställa implementeringen av M262 för att kunna påbörja programmeringen i form av ett Sequential Function Chart (SFC).

ZBRN2 byttes ut mot en liknande modul, ZBRN1 med Ethernetstöd. Detta då den ursprungliga modulen endast hade stöd för seriell lina och det fanns en önskan att övergå helt till ethernetkommunikation i konstruktionen. Fler in- och utgångar till PLCn behövdes och två kompatibla moduler, TM3DI16 OCH TM3DQ8R, anslöts till M262 för att realisera det.

Under konstruktionens gång skedde även ett byte utav M262. Den ersattes med en uppdatering av samma modell som var kompatibel med det önskade robotbiblioteket då den första saknade detta med trolig anledning till brist på minne. Uppbyggnaden av roboten i Machine Expert visas i figur 2.

En energimätare med syftet att samla data för energiförbrukning i skåpet kopplades upp till roboten i Machine Expert och dess utgångar kartlades i programmet. Färdiga funktioner var tillgängliga för korrekt beräkning av de värdena energimätaren genererade, vilket även kom till användning senare när HMI skapades. Dessa funktioner, som var skrivna i strukturerad text, tilldelades av handledare och var en del av originalprojektet 2015. Energimätaren kopplades via en seriell ingång på PLCn.




*Figur 2. Device tree i Machine Expert*

En del av konstruktionsarbetet omfattade även baklängeskonstruktion på grund av det faktum att det ärvda projektet inte varit fullt dokumenterat. Elkonstruktionen saknade dokumentation sett till kablage samt att det gamla I/O-systemet inte var kartlagt. Detta arbete skedde främst tillsammans med handledare från företaget då

endast de hade tillgång till den föråldrade mjukvara som krävdes. Tillsammans diskuterades och testades olika lösningar för att ta reda på vad de olika ingångar och utgångar använts till och om de i den då aktuella konstruktionen fortfarande användes.

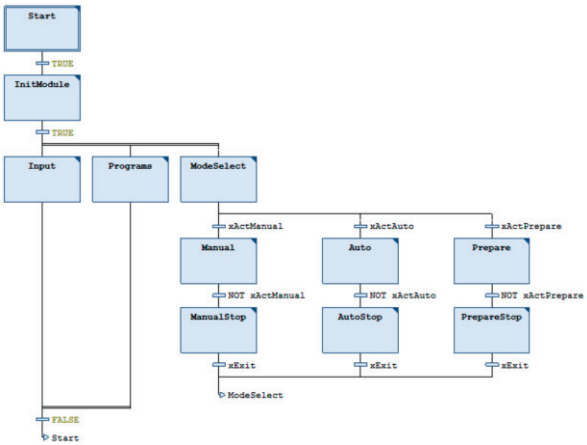
## 3.4 Programmering

Programmeringen påbörjades tillsammans med handledare under utbildningsfasen. Första iterationerna skrevs utan att vara uppkopplad mot PLCn och programmerades för att få en förståelse för variabler och hur strukturen fungerade. Därefter skrevs kortare program och laddades upp till styrsystemet via Ethernet med ett syfte att uppskatta hur koden exekverades i PLCn.

Nästa fas i programmeringen omfattade test i form av drift av motorerna och samverkan mellan komponenterna. Olika körlägen och rörelsemönster testades i programmet tillsammans med styrning som start och stopp via RFID-systemet och ZBRN1. Programmeringen skrevs i strukturerad text (ST) under utbildnings- och testfaserna.

Efter de första två faserna i programmeringen och konstruktionen av elskåpet färdigställdes skrevs koden i form av ett Sequential Function Chart (SFC), vilket illustreras i figur 3. Strukturen i SFCn grundades i robotens tre olika körlägen och för varje block skrevs enskild kod i programmeringsspråket Strukturerad Text (ST). Styrning av roboten utfördes i början av examensarbetet med camstyrning vilket inte producerade de resultat som förväntades. En omstrukturering utfördes och styrningen ändrades till att programmeras med Schneider Electrics robotbibliotek. Programmering av Human Machine Interface (HMI) gjordes efter att en övervägande del av robotprogrammeringen var fullbordad.

Implementering och installation av en 360-graders kamera påbörjades under programmeringsfasen som senare fick avbrytas då tiden inte räckte till och den tilldelade kamerans föråldrade mjukvara inte gick att implementera. Beslutet togs att lämna kameran till vidareutveckling i framtiden.



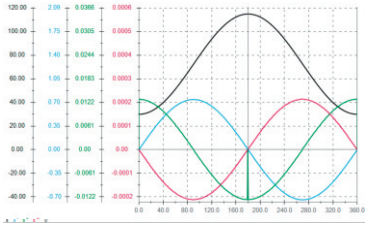

*Figur 3. SFC SR_Main*

### 3.4.1 Camstyrning

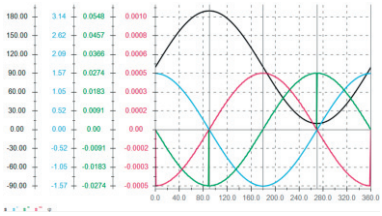
Implementering av camstyrning i det här examensarbetet var primärt i utbildningssyfte för att förstå kalkyleringen bakom rörelsen på axlarna. En virtuell axel skapades i Machine Expert, se figur 4, med syftet att agera som en master till de tre verkliga axlarna. För att realisera camstyrningen nyttjades metoder från det öppna biblioteket PLCOpen (PLCO) vilka gjorde det möjligt att programmera rörelser till axlarna. Master-axeln kom till användning när camdiagram, för de övriga axlarna, skapades då en referensaxel behövdes när rörelsekurvorna bildades.


*Figur 4. SercosMaster med axlar för camstyrning*

Camdiagram för X- och Y-axeln skapades i Machine Expert för att etablera en samtidig och koordinerad rörelse på styraxlarna. För att säkerställa att axlarna rörde sig under samma tidpunkt agerade masteraxeln som referensaxel vilket i dessa fallen var X-axeln i kurvorna. Figur 5 och Figur 6 demonstrerar hur diagrammen ser ut i programmet. Den svarta kurvan visar det önskade rörelsemönster robotens axel ska utföra medan de andra kurvorna visar de beräknade värdena för acceleration (grön), hastighet (blå) och ryck (röd) som servomotorn kommer genomföra för att uppnå rörelsen.



*Figur 5. Camdiagram för X-axeln*



*Figur 6. Camdiagram för Y-axeln*

### 3.4.2 Robotbibliotek

I början av arbetet utnyttjades PLCO-biblioteket för all programmering av robotrörelserna och till större delen skrevs koden med linjära rörelser för var axel. Det hindrade arbetet med att få effektiva rörelser genom det tredimensionella planet där axlarna samverkade. CAM-styrning via PLCO var inget alternativ och detta löstes genom att tillföra ett mer avancerat robotbibliotek som erbjöd kalkylerade rörelser. Biblioteket som nyttjades här var Schneider Electrics egna robotbibliotek för PLC med kompatibilitet för modellen M262. Stödet för att nyttja biblioteket i M262 är en ny funktion vilket resulterade i att dokumentationen var begränsad. I samband med programmering med robotbiblioteket tillämpades och implementerades larm för felsökning under initiering och drift samt till säkerhetssystemet, både för nödstopp och ljusridå.

### 3.4.3 HMI

Programmering av Human Machine Interface (HMI) inleddes när programmeringsarbetets avslutande fas påbörjades. Först utfördes den grafiska designen och därefter kopplades variabler från Machine Expert till Operator Terminal Expert via en länk, för att sedan skapa förbindelser med de interaktiva verktygen på skärmen. En larmgrupp konstruerades med de färdigkodade alarmen och sammankopplades med den grafiska larmlistan i gränssnittet. Totalt implementerades 4 olika skärmar som skulle utgöra HMI, dessa var huvudskärmen, skärmen för larmlistan, val av körläge samt en skärm för förberedning av roboten.

## 3.5 Robot

Den avslutande fasen för examensarbetet hanterade uppbyggnad och inkoppling av den nya roboten. Anslutningen mellan drivenheterna och styrmotorerna på den äldre roboten avbröts och den nya sattes på plats och en anslutning upprättades. Då den nya roboten var utrustad med andra motorer behövde drivstegen informeras om det och för att drivenheterna skulle uppfylla de nya motorernas specifikationer

flashades de om på nytt. Avslutningsvis placerades Magelis-skärmen med HMI på plats intill roboten.

## 3.6 Inhämtning av information

Inhämtning av både information och data har skett löpande under hela examensarbetet där de huvudsakliga källorna har varit i form av manualer och expertis av handledare. Den tekniska specifikationen för den aktuella hårdvaran har varit väsentlig både för konstruktion och programmering av portalroboten. Manualerna har varit lättillgängliga eftersom hårdvaran är Schneider Electrics egna produkter.

Mycket information har varit tilldelad av handledarna genom projektets gång, både under utbildningen och med deras tillgänglighet. Utöver utbildningstillfällena var handledarna tillgängliga vid fysiska möten på kontoret där arbete utfördes i form av installering av komponenter, rensning av ej använda komponenter och programmering med testkörning.

## 3.7 Källkritik

Källorna för teknisk bakgrund för en kartesisk bakgrund, referens [1], och standarden IEC-61131-3, referens [2] är artiklar upplagda på IEEE vilket då också betyder är de är granskade, därför anses dessa källor pålitliga.

Figur 1, referens [14], hämtades från mathworks och är framtagen i MATLAB. Källan anses pålitlig då MATLAB används vid undervisning på Lunds universitet.

Den vetenskapliga artikeln om splines, referens [13], anses pålitlig då författarna som har publicerat den tillhör University of Oslo samt University of Rome Tor Vergata.

Teknisk specifikation om den hård- och mjukvaran som har använts under examensarbetet har tagits ur Schneider Electrics manualer, referens [3][5][10][11][15][17]. Informationen från dessa källor anses pålitliga då det är företagets egna manualer med innehåll som företaget står bakom. Information och data från Schneider Electrics officiella webbsida, referens, [4][6][8][9][18][19][20][21], anses mycket pålitliga av samma anledning.

Robotbiblioteket finns tillgängligt på internet via Schneider Electric och information hämtad därifrån till förklaring av vad ett robotbibliotek är, referens [16], och beskrivning av vad en spline är, referens [22], anses då pålitliga.

Källan till den tekniska bakgrunden om RFID-systemet, referens [7], är den officiella hemsidan för hårdvaran, vilken då känns mycket tillförlitlig. Generell information om Sercos och Sercos III, referens [12], anses pålitlig då källan är Sercos officiella hemsida och kan antas innehålla korrekt information.

# 4. Analys

I följande kapitel berörs de val som togs för att uppnå målen för examensarbetet. En diskussion kring de problem som uppstod, med lösningar, behandlas även i den här delen.

## 4.1 Hårdvara

Vilken hårdvara som skulle behållas och vilken som skulle bytas ut bestämdes i samspel med handledare för examensarbetet på Schneider Electric. Besluten bidrog till att robotens föregående styrkontroll, LMC 078, byttes ut till den nyare och modernare PLC-kontrollen M262. Även ett nytt I/O-system implementerades och ersatte det äldre, då stora delar av det inte användes och platsen i elskåpet behövdes till annan hårdvara. Bytet som skedde mellan de två olika M262 utfördes med anledning av att den första kontrollen inte var kompatibel med det robotbibliotek som användes, följden blev att den ersattes av en kontroll som uppfyllde kraven. Behovet av fler in- och utgångar bidrog till sammankoppling med de två kompatibla modulerna, ett behov som i senare skede inte behövdes. Beslutet att behålla modulerna grundades i framtida behov vid eventuell vidareutveckling av styrsystemet.

Upplösningen i Harmony-modulen ZBRN2 uppfyllde inte kraven för examensarbetet då den inte registrerade knapptryckningar i den önskade frekvensen. För bättre upplösning krävdes en liknande modul med en snabbare ingång och utifrån de önskade kriterierna valdes en ZBRN1. Den stora skillnaden mellan modulerna är ingången som bidrar till hastigheterna, ZBRN2 går via en seriell lina medan ZBRN1 går via Ethernet. Förbindelsen med Altivar 340 skapades med anledning av expansionsmöjligheter, den användes aldrig i examensarbetet. Beslutet att inte koppla ur den från elskåpet togs med tanke på framtida utvecklingsmöjligheter och då utrymmet inte behövdes till övriga komponenter lämnades den kvar i konstruktionen och sammankopplad med Machine Expert. Resterande hårdvara fungerade och uppfyllde den nödvändiga prestandan vilket resulterade i att den inte behövde ersättas med nya komponenter. Beslutetet bidrog till att eventuell tid som var avsedd för konstruktion istället kunde läggas på programmering.

Uppbyggnaden och implementeringen av den nya roboten var sedan examensarbetets start bestämt. Grundtanken var att uppbyggnaden av den skulle ske under konstruktionsfasen men då förseningar uppstod kom den på plats under och efter den avslutande delen av programmeringsfasen. Samtidigt installerades skärmen och industri-PCn, där HMI skulle uppvisas.

## 4.2 Val av programmeringsspråk och struktur

Två olika programmeringsspråk, med olika syften, används i det här examensarbetet. Sequential Function Chart (SFC) var nödvändig för att skapa en grundläggande överblick över hela programmet. En väsentlig fördel med valet av SFC-strukturen var att koden även kunde följas visuellt för att garantera att koden inte tillät drift utanför de tillåtna driftstegen. Beslutet angående tillämpningen och användningen av Structured Text (ST) var primärt personligt, då det språket var det lämpligaste för examensarbetarna.

Robotstyrningen byggdes upp med en tillståndsmaskin i form av case-satser där rörelsesekvenserna och stopphantering programmerades. Denna metod valdes då exekveringen av koden lättare kunde följas.

## 4.3 Rörelsesekvenser

Tidigt under programmeringsfasen togs beslutet att implementering av två olika körlägen av roboten skulle ske, med syftet att tidigt möjliggöra anpassning av grundstrukturen på koden. Ett manuellt körläge tillämpades för manuell styrning av varje enskild axel. Det automatiska körläget försågs med tre olika rörelsesekvenser med olika huvudsyften för varje sekvens. Syftet med den första sekvensen, “Pick and place” var att demonstrera en linjär rörelse där samtliga axlar var delaktiga. Ytterligare en “Pick and place”-sekvens implementerades där X och Y-axlarna utförde en linjär rörelse medan Z-axeln rörde sig i en halvellips, med ändamål att visa effektiviteten i säljande syfte. Resultatet blev en rörelse med spline-kurvor. Den sista rörelsesekvensen skapades för att demonstrera en cirkulär rörelse och uppvisades i form som ett oändlighetstecken.

## 4.4 HMI

Målet med framställning av ett Human Machine Interface (HMI) var att skapa ett grafiskt gränssnitt som både var funktionellt och användarvänligt. En huvudskärm implementerades där värdena från energimätaren visualiserades med syftet att informera om energiförbrukningen. En skärm skapades också för larmlistan där de aktiverade och utlösta alarmen skrevs ut på enskilda rader med tillhörande datum och tid. En återställningsknapp implementerades på larmskärmen med möjligheten att åtgärda det utlösta larmet och återställa det på ett säkert sätt. För de olika körlägena togs beslutet att illustrera dem med enkla knappar för att göra styrningen lättförståelig och begriplig. Bredvid knapparna vilka representerade de tre automatiska rörelsesekvenserna placerades lampor som agerade indikatorer för den aktiva sekvensen.

Den sista skärmen visualiserar initieringssteget “prepare” i programkoden. För att initieringen ska ske korrekt placerades en informativ textruta med nödvändig information inför starten för att motverka onödiga felkoder. Numeriska rutor placerades ut för att underlätta positionering av styraxlarna inför initiering, dessutom placerades lampor ut för att indikera var i initiering koden befann sig, med syfte att informera användaren.

## 4.5 Problem

### 4.5.1 Uppkoppling till internet

När PLCn skulle kopplas upp till nätet uppdagades det att det inte var möjligt då förhöjd säkerhet inom företaget förhindrade det. Påföljden blev att PLCn inte fick åtkomst till internet vilket orsakade att funktioner inte kunde nyttjas i examensarbetet. Eftersom det inte etablerades någon internetuppkoppling kunde inte HMI köras via funktionen för webbaserad simulering vilket PLCn hade stöd för.

Problemen som uppstod med uppkopplingen till nätet bidrog också till att implementeringen av en digital tvilling i det digitala molnverktyget Machine Advisor förhindrades detta då internet var ett krav för tjänsten. Med anledning av den förhöjda säkerheten inom företaget samt tidsbrist fanns det inte utrymme för att hitta en lösning på problemet, inom tiden för utförandet av examensarbetet.

### 4.5.2 Programmering

Då det här examensarbetet har omfattat väldigt specifik hård- och mjukvara i form av Schneider Electrics egna produkter har tillgängligheten av information angående programmeringen varit väldigt begränsad. Det här resulterade i att mycket av projektets tid gick åt till testning av kod via den information och de definitioner som fanns tillgängliga om biblioteken.

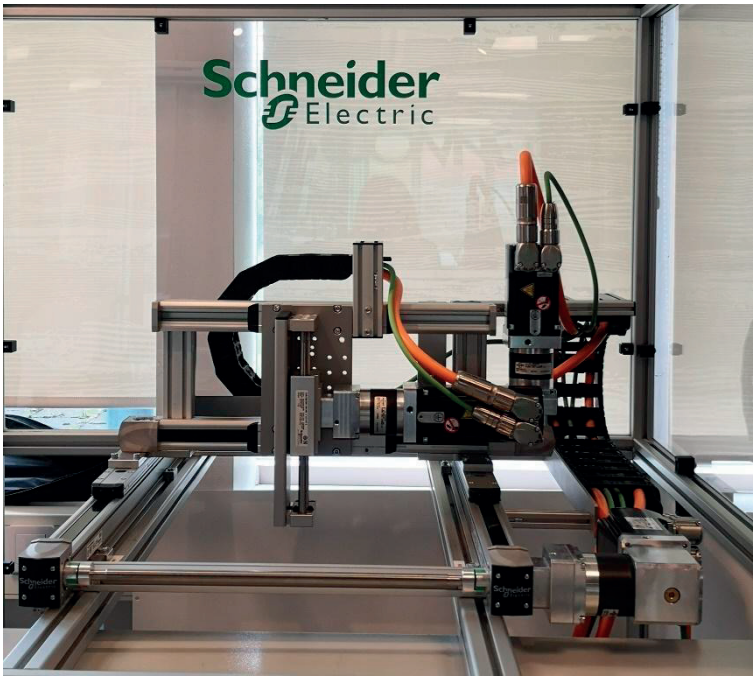
# 5. Resultat

Kommande avsnitt presenterar den resulterande hårdvaran för examensarbetet, även en detaljerad beskrivning av programkoden för portalroboten framförs här. Kapitlet avslutas med en redogörelse för funktionerna i det skapade Human Machine Interface (HMI).

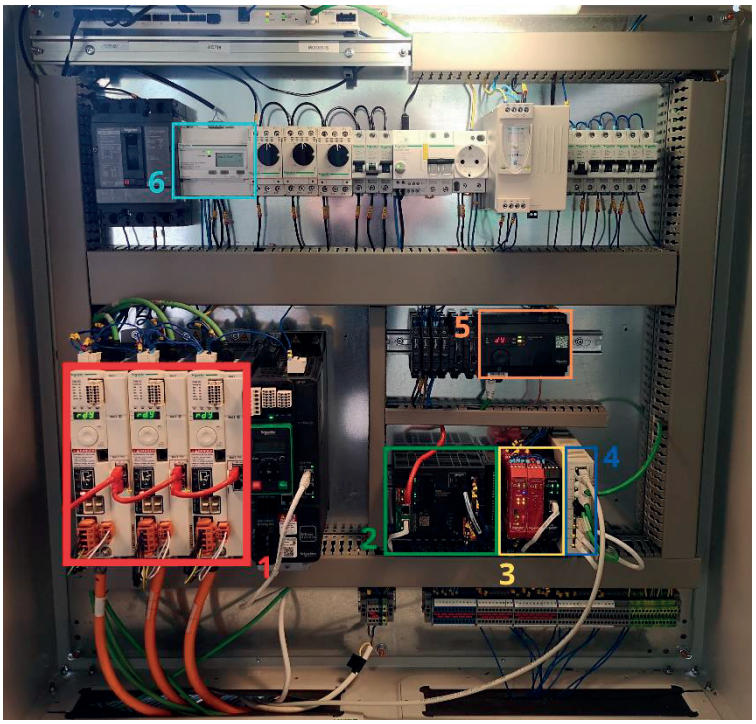
## 5.1 Hårdvara

Den resulterande hårdvaran består av den nya portalroboten samt ett uppdaterat elskåp med tillhörande komponenter, både ny utrustning men även återanvänd utrustning. Portalroboten är utrustad med nya servomotorer av typen BMH0701P07A2A på X- och Y-axeln och typen BMH0701P07F2A på Z-axeln. Servomoterna är anslutna till styrenheterna av typen Lexium 32 S, vilka är sammankopplade till styrenheten M262 via modulens Sercos III-ingång. Hårdvaran för RFID-systemet (Osisense XG), säkerhetssystemet (Modicon Preventa Safety Modules), kommunikationssystemet (Acti 9), Harmony Hub (ZBRN1) och Altivar 340 är anslutna till PLC-kontrollen via en switch vilken är ansluten till M262 ethernet 1-port. Energimätaren är ansluten till styrenhetens seriella ingång, vilket är en separat ingång. Två kompatibla moduler till M262, TM3DI16 OCH TM3DQ8R, är anslutna i följd på sidan av kontrollen. Hårdvaran för HMI består av en Magelis-skärm med tillhörande iPC. I figur 7 visas roboten och i figur 8 visas det färdiga elskåpet med de olika komponenterna markerade.

1. Servodrivenheterna
2. PLC-modulen M262
3. Säkerhetssystemet Preventa
4. Switch
5. Harmony Hub ZBRN1
6. Energimätare




*Figur 7. Roboten*




*Figur 8. Elskåpet*

## 5.2 SR_Main

Grundstrukturen i programmet som styr portalroboten etableras i SR_Main. SR_Main består av ett SFC och en variabellista, se figur 9, som styr stegbyte, stopphantering och uppbyggnaden av en strukturlista. Typen av strukturlistan innehåller strukturen ST_MODULETOMAIN och heter astModules. Varje steg i SFCn (Figur 2) exekverar ett eget program som enbart körs när steget är aktivt. Dessa program är skrivna i strukturerad text.

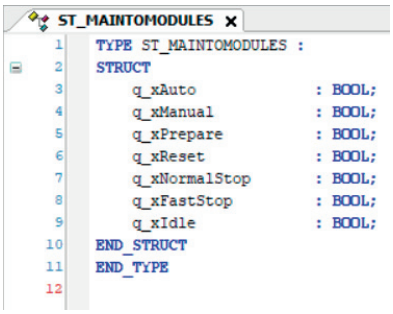
SR_Main

```
1  PROGRAM SR_Main
2  VAR
3      xActManual      : BOOL;
4      xActAuto        : BOOL;
5      xActPrepare     : BOOL;
6      xExit           : BOOL;
7
8      xStart          : BOOL;
9      xReset          : BOOL;
10     xNormalStopDone : BOOL;
11     xFastStopDone   : BOOL;
12     xPrepareDone    : BOOL;
13
14     stMainToModules : ST_MAINTOMODULES;
15
16     astModules      : ARRAY[1..5] OF ST_MODULETOMAIN;
17
18     iInputLoop      : INT;
19
20     iSercosReset    : INT;
21     xSercosReset    : BOOL;
22
23     fbLoggerPoint   : APL.FB_LoggerPoint;
24
25     etDiag          : GD.ET_Diag;
26     etDiagExtAPL    : APL.ET_DiagExt;
27
28     xStopiding      : BOOL;
29     axAlarm         : ARRAY[1..8] OF BOOL;
30     asAlarmMsg      : ARRAY[1..8] OF STRING;
31 END_VAR
32
```

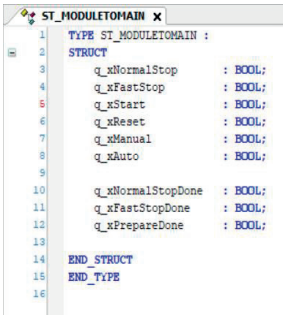
*Figur 9. Variabellista SR_Main*

Koden är uppbyggd på det sätt att de installerade komponenterna kan ses som en modul och är konstruerade som strukturer av typen ST_MODULETOMAIN. På det viset behandlar main alla moduler på ett generellt sätt men ger rum till olika funktioner hos komponentens egna skrivna program. Kommunikationen mellan main och modul är baserad på två strukturer, ST_MAINTOMODULES (Figur 10) och nämnda strukturen ST_MODULETOMAIN (Figur 11). Dessa är utformade genom att main kommunicerar till varje modul om vilket SFC-steg som är aktuellt, om ett stopp är efterfrågat eller om en återställning är initierad. På ett liknande sätt svarar varje modul med information till main. Varje modul har i grunden möjlighet att kalla på ett normal- eller snabbstopp och även berätta att den slutfört ett normal- eller snabbstopp via utmatningen q_xNormalStopDone respektive

q_xFastStopDone. Skillnaden på stoppvariablerna är att normalstopp hanterar en vanlig stoppbegäran, vilket slutför den aktuella rörelsen medan snabbstopp hanterar stopp begärda från säkerhetsreläerna och andra inmatningar med krav på omedelbar inbromsning. Modulerna kan även begära att starta roboten och ändra drift mellan manuell och auto.

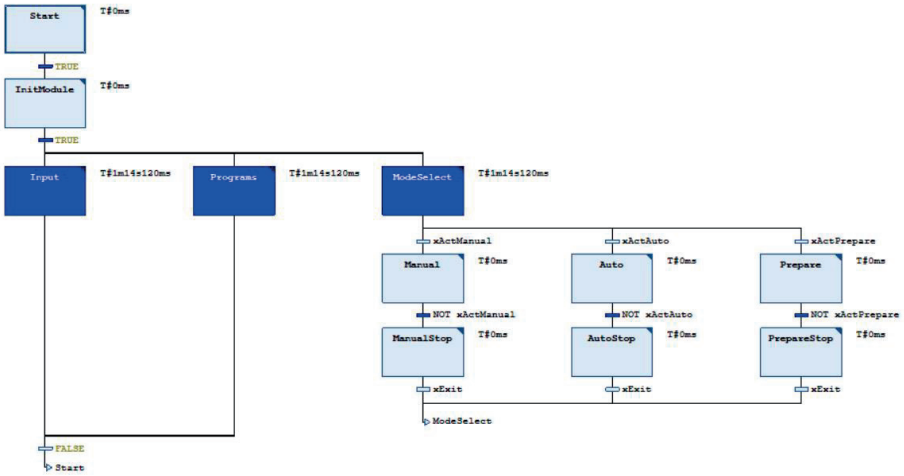



*Figur 10. ST_MAINTOMODULES*




*Figur 11. ST_MODULETOMAIN*

Vid uppstart faller SR_Main genom stegen Start och InitModule. Dessa SFC-steg är en del av initieringen och körs enbart vid uppstart. Efter det här stannar SFCn i det som kommer att kallas “Idle”. I idle väntar roboten på kommandon om vad som ska utföras. Beroendet på när roboten är i idle är om SFCn befinner sig i ModeSelect. I idle är även Programs och Input aktiva som kan ses i figur 12. Dessa tre steg uppdateras sekventiellt men körs parallellt med varandra i SFCn. SR_Main uppdateras cykliskt var 10 ms. Aktiveras ett start från någon av modulerna lämnas ModeSelect-steget till valt driftläge.




*Figur 12. SFC under Idle, blåmarkerade är aktiva steg.*

## 5.3 Start

Startsteget i sekvensdiagrammet körs vid uppstart av roboten och dess mål är att säkerställa att de variabler som vid hög flank aktiverar ett stegbyte är låga. När detta exekverats lämnar SFC:n startsteget och övergår direkt till InitModule-steget då beroendet till hopp alltid är TRUE. Figur 13 visar koden i Start stegets programkod.

SR_Main.Start_active

```
xActManual      := FALSE;
xActAuto        := FALSE;
xActPrepare     := FALSE;
xExit           := FALSE;
xStart          := FALSE;
xReset          := FALSE;
xNormalStopDone := FALSE;
xFastStopDone   := FALSE;
```

*Figur 13. Programkod i Start, återställning av variabler*

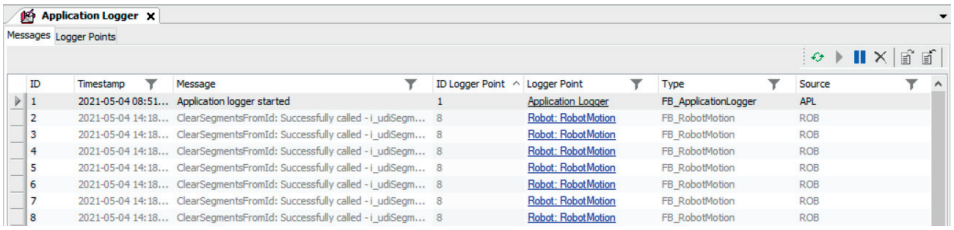
## 5.4 InitModule

InitModule används i koden för att aktivera en applikationslogg under initiering av programmet. Denna loggbok används under drift för möjligheten att följa specifika punkter i programmet, i detta fall registreras variablerna etDiag och etDiagExtAPL. Variablen etDiagExt varierar i programmet och är specifik till var i programmet den avläses. I Figur 14 heter variabeln etDiagExtAPL då den för en logg direkt i anslutning till applikationsloggen. Detta ändras om loggpunkten förs i anslutning till en komponent såsom roboten. Dessa felkoder kan endast avläsas med uppkoppling mot styrsystemet med en dator, ett exempel på hur loggen kan se ut med loggpunkter finns i figur 15.

SR_Main.InitModule_active

```
APL.G_ifApplicationLogger.RegisterCommunicationService(q_etDiag=> etDiag, q_etDiagExt=> etDiagExtAPL);

IF etDiag <> GD.ET_Diag.Ok THEN
    axAlarm[1] := TRUE;
    asAlarmMsg[1] := 'Failed to initialize application logger...';
END_IF


fbLoggerPoint.RegisterLoggerPoint(  i_ifParent      := APL.G_ifApplicationLogger,
                                    i_sName         := 'SR_Main',
                                    i_sType         := 'SR_Main',
                                    i_sSource       := 'Project',
                                    q_etDiag        => etDiag,
                                    q_etDiagExt     => etDiagExtAPL);

IF etDiag <> GD.ET_Diag.Ok THEN
    axAlarm[1] := TRUE;
    asAlarmMsg[1] := 'Failed to initialize application logger...';
END_IF
```

*Figur 14. InitModule, ApplicationLogger konstruktion*




| ID | Timestamp | Message | ID Logger Point | Logger Point | Type | Source |
|---|---|---|---|---|---|---|
| 1 | 2021-05-04 08:51... | Application logger started | 1 | Application Logger | FB_ApplicationLogger | APL |
| 2 | 2021-05-04 14:18... | ClearSegmentsFromId: Successfully called - i_udiSegm... | 8 | Robot: RobotMotion | FB_RobotMotion | ROB |
| 3 | 2021-05-04 14:18... | ClearSegmentsFromId: Successfully called - i_udiSegm... | 8 | Robot: RobotMotion | FB_RobotMotion | ROB |
| 4 | 2021-05-04 14:18... | ClearSegmentsFromId: Successfully called - i_udiSegm... | 8 | Robot: RobotMotion | FB_RobotMotion | ROB |
| 5 | 2021-05-04 14:18... | ClearSegmentsFromId: Successfully called - i_udiSegm... | 8 | Robot: RobotMotion | FB_RobotMotion | ROB |
| 6 | 2021-05-04 14:18... | ClearSegmentsFromId: Successfully called - i_udiSegm... | 8 | Robot: RobotMotion | FB_RobotMotion | ROB |
| 7 | 2021-05-04 14:18... | ClearSegmentsFromId: Successfully called - i_udiSegm... | 8 | Robot: RobotMotion | FB_RobotMotion | ROB |
| 8 | 2021-05-04 14:18... | ClearSegmentsFromId: Successfully called - i_udiSegm... | 8 | Robot: RobotMotion | FB_RobotMotion | ROB |

*Figur 15. InitModule, Exempel på Application Logger*

## 5.5 Input

Input är ett steg som alltid hålls aktivt i programmet efter att initieringen är slutförd. Vid varje uppdateringscykel söker main igenom om någon modul efterfrågar ett normalstopp, snabbstopp, återställning eller start via for-loopar baserade på antalet moduler. Om detta är fallet uppdaterar main sin variabel knuten till aktionen och förmedlar den till resterande moduler. Figur 16 visar hur koden är uppbyggd mellan kommunikationen från moduler till main och figur 17 visar kommunikationen mellan main och modulerna.

SR_Main.Input_active

```
1  stMainToModules.q_xFastStop := FALSE;
2  FOR iInputLoop := 1 TO 5 DO
3      IF astModules[iInputLoop].q_xFastStop THEN
4          xStopiding := TRUE;
5          stMainToModules.q_xFastStop := TRUE;
6      END_IF
7  END_FOR
8
9  stMainToModules.q_xNormalStop := FALSE;
10 IF NOT stMainToModules.q_xFastStop THEN
11     FOR iInputLoop := 1 TO 5 DO
12         IF astModules[iInputLoop].q_xNormalStop THEN
13             xStopiding := TRUE;
14             stMainToModules.q_xNormalStop := TRUE;
15         END_IF
16     END_FOR
17 END_IF
18
19 xReset := FALSE;
20 FOR iInputLoop := 1 TO 5 DO
21     IF astModules[iInputLoop].q_xReset THEN
22         xReset := TRUE;
23     END_IF
24 END_FOR
25
26 xStart := FALSE;
27 FOR iInputLoop := 1 TO 5 DO
28     IF astModules[iInputLoop].q_xStart THEN
29         xStart := TRUE;
30     END_IF
31 END_FOR
```

*Figur 16. Input, kommunikation mellan moduler till main*

```
52
53  stMainToModules.q_xIdle := ModeSelect._x;
54  stMainToModules.q_xAuto := xActAuto;
55  stMainToModules.q_xManual := xActManual;
56  stMainToModules.q_xPrepare := xActPrepare;
57  stMainToModules.q_xReset := xReset;
```

*Figur 17. Input, kommunikation mellan main och moduler*

Input-steget hanterar även återställning av servodrivenheterna genom en tillståndsmaskin som kan skådas i figur 18. Om ett fel uppstått där servodrivenheternas kommunikationstillstånd rapporterar ett fel krävs en omstart av detta. Tillståndsmaskinen aktiveras när en återställning begärs samtidigt som det finns ett kommunikationsproblem.

```
32
33  CASE iSercosReset OF
34
35  0:
36      IF xReset AND SercosMaster.SercosPhaseChanger.ActualState = S3M.ET_SercosState.Error OR
37      SR_RobotNGSFC.xSercosReset AND SercosMaster.SercosPhaseChanger.ActualState = S3M.ET_SercosState.Error THEN
38          iSercosReset := 10;
39      END_IF
40  10:
41      SercosMaster.SercosPhaseChanger.DesiredPhase := S3M.ET_SercosPhase.NRT;
42      IF SercosMaster.SercosPhaseChanger.ActualState = S3M.ET_SercosPhase.NRT THEN
43          iSercosReset := 20;
44      END_IF
45  20:
46      SercosMaster.SercosPhaseChanger.DesiredPhase := S3M.ET_SercosPhase.Phase4;
47      IF SercosMaster.SercosPhaseChanger.ActualState = S3M.ET_SercosPhase.Phase4 THEN
48          iSercosReset := 0;
49      END_IF
50
51  END_CASE
52
```

*Figur 18. Tillståndsmaskinen för återställning av sercos.*

## 5.6 Programs

SFC-steget Programs används för att anropa programmen som ska köras, vilket visas i figur 19. Detta steg uppdateras varje cykel och lämnas aldrig. Var modul körs som ett program med en generell kod för modul-till-main kommunikation samt unik kod baserat på vad det är för modul. Här tilldelas modulerna i modulstrukturlistan astModules.

SR_Main.Programs_active

```
1  SR_Harmony(iq_stMainToModule:= stMainToModules, iq_stModuleToMain:=astModules[1] );
2  SR_RFID(iq_stMainToModule:= stMainToModules, iq_stModuleToMain:= astModules[2]);
3  SR_Preventa(iq_stMainToModule:= stMainToModules, iq_stModuleToMain:= astModules[3]);
4  SR_RobotNGSFC(iq_stMainToModule:= stMainToModules, iq_stModuleToMain:= astModules[4]);
5  SR_HMI(iq_stMainToModule:= stMainToModules, iq_stModuleToMain:= astModules[5]);
```

*Figur 19. Programs, anrop*

### 5.6.1 Harmony

Programmet SR_Harmony hanterar inmatningen till ZBRN1-modulen och består av tre delar. En struktur, ST_Harmony, är uppbyggd för att hantera variablerna som är direkt mappade mot hårdvaran. Figur 20 visar exempel på denna mappning. Variabler blir TRUE när en knapp sammanlänkad till modulen trycks ned.

| Variable | Mapping | Channel | Address | Type |
|---|---|---|---|---|
| Inputs | | | | |
| Application.SR_Harmony.stHarmony.wHarmony | | IW0 | %IW181 | WORD |
| Application.SR_Harmony.stHarmony.axHarmonyInput[0] | | I0 | %IX362.0 | BOOL |
| Application.SR_Harmony.stHarmony.axHarmonyInput[1] | | I1 | %IX362.1 | BOOL |

*Figur 20. Exempel på mappning.*

Funktionsblocket FB_Harmony är skapat och översätter inmatningen från strukturen och matar ut de som används för att vara enklare att använda i koden. Figur 21 visar hur funktionsblocket anropas i SR_Harmony.

```
fbHarmony(      iq_stHarmony:= stHarmony,
                q_iHarmonyValue=> ,
                q_xLimitTrig=> ,
                q_xGreenButton=> ,
                q_xRedButton=> );
```

*Figur 21. Funktionsblocket FB_Harmony*

SR_Harmony hanterar sedan vad inmatningen ska genomföra och vad som kommuniceras med main. Figur 22 demonstrerar hur main meddelas och hur utmatningen från funktionsblocket används för en stoppknapp. Anledningen till att knappen aktiverar ett larm och inte direkt kommuniceras till main är för att knappen är momentärt sann vid trycket. För att stoppet inte ska gå missat i den sekventiella uppdateringen triggas ett larm som ligger högt tills stoppet är hanterat, vilket figur 22 visar.

```
//Main to module 1 variables
iq_stModuleToMain.q_xFastStopDone := iq_stMainToModule.q_xFastStop;
iq_stModuleToMain.q_xNormalStopDone := iq_stMainToModule.q_xNormalStop;
iq_stModuleToMain.q_xStart := fbHarmony.q_xGreenButton;
iq_stModuleToMain.q_xNormalStop := axAlarm[1];
iq_stModuleToMain.q_xFastStop := axAlarm[2];
iq_stModuleToMain.q_xReset := axAlarm[3];

//Stop button
IF fbHarmony.q_xRedButton THEN
    axAlarm[1] := TRUE;
END_IF
```

*Figur 22. Kommunikation till main och larm som aktiveras av stoppknapp.*

### 5.6.2 RFID

SR_RFID hanterar inmatning från RFID-systemet. Konstruktionen är uppbyggd på samma sätt som Harmony programmet med en struktur, ett funktionsblock och huvudprogrammet. Funktionsblocket är konstruerat och anropat men är ej färdigutvecklat och används inte då tiden valdes att slutföra andra funktioner. Istället används RFID-taggarnas siffervärde direkt i koden via strukturen. Alla

taggarnas värde och typ återfinns i tabell 1. Figur 23 visar koden som är implementerad för att styra roboten med taggarna. Variablerna xPnP, xInfinity och xPnPSpline är beroenden till startutmatningen i SR_RFIDs kommunikation till main, går någon av dessa höga efterfrågas en start av robotdriften. Var enskild variabel är också ett beroende till vilken sekvens i autodrift som aktiveras.

```
14  xPnP := FALSE;
15  xInfinity := FALSE;
16  xPnPSpline := FALSE;
17  IF stRFID.wTagSystemFlags > 0 THEN
18      IF stRFID.awUID[1] = 19209 THEN
19          xPnP := TRUE;
20      ELSIF stRFID.awUID[1] = 19721 OR stRFID.awUID[1] = 18885 THEN
21          xInfinity := TRUE;
22      ELSIF stRFID.awUID[1] = 51963 THEN
23          xPnPSpline := TRUE;
24      END_IF
25  END_IF
```

*Figur 23. Specifika kort hanterar olika rörelsesekvenser.*

### 5.6.3 Preventa

SR_Preventa har samma struktur som de tidigare modulerna. Detta program hanterar utlösningar från säkerhetsreläerna som är installerade mot preventa hårdvaran. Modulerna är färdigprogrammerade från tidigare projekt och då vi inte haft full tillgång till detta har baklängeskonstruktion gett upphov till den kritiska variabeln xSafetyReset. Denna variabel tilldelas samma värde som variabeln för återställning från main och kan ses i figur 24.

```
40  //Reset emergency stop
41  stPreventa.xSafetyReset := iq_stMainToModule.q_xReset;
```

*Figur 24. Variabeln xSafetyReset*

När ett säkerhetsrelä utlöses stoppar preventamodulerna all drift av servodrivenheterna tills stoppet är bekräftat och åtgärdat. Denna bekräftelse är direkt beroende av PLC utgången Q0 vilket uppdagades via baklängeskonstruktionen. Detta resulterade i att variabeln xSafteyReset skapades och kartlades till utgången som kan ses i figur 25 för att användas som bekräftelse och återställning.

| | | | | |
|---|---|---|---|---|
| Outputs | | | | |
| | | QB0 | %QB0 | BYTE |
| Application.SR_Preventa.stPreventa.xSafetyReset | | Q0 | ~~%QX0.0~~ | BOOL |

*Figur 25. xSafteyReset mappat till utgång Q0 på PLC.*

### 5.6.4 HMI

HMI:t hanteras som en modul av main men konstruktionen skiljer sig från resterande moduler. Alla variabler som används i HMI-programmering är skapade i en struktur med global åtkomst då strukturen anropas i en global variabellista. Detta underlättar sammanlänkningen med Operator Terminal Expert och de beroenden kring inmatning då variablerna som behövs i HMI är åtkomliga från endast en plats. I SR_HMI sammanlänkas de variablerna synliga i HMI:t mot deras motsvarighet i PLC-programmeringen och logiken kring de interaktiva variablerna definieras.

## 5.7 ModeSelect

ModeSelect är steget i mains SFC då roboten inväntar inmatning från en användare, figur 26 visar vilka beroenden som ska vara uppfyllda för att lämna steget till auto, manuell eller förberedelse. När ett stegbyte sker lämnar SFC-programmet ModeSelect för det valda steget. “Manual”, “Auto” och “Prepare”-stegen i sekvensdiagrammet har funktionen att meddela modulerna vilket steg som är aktuellt för att visa typen av drift som är tillåten. Avläsningen sker via programmet i Input som visas i figur 17. Programkoden i de olika läges-stegen är att lämna tillståndet när någon form av stopp anropas.

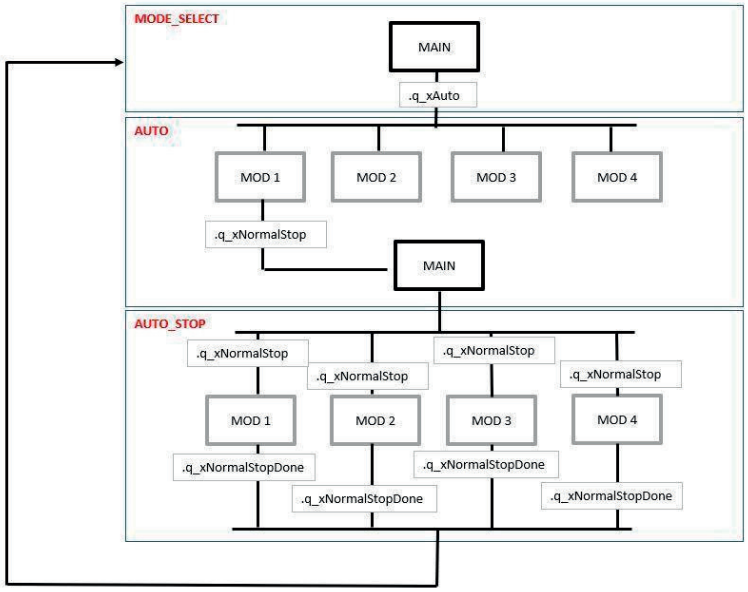
```
IF G_stHMI.xManualStart AND NOT stMainToModules.q_xFastStop AND NOT stMainToModules.q_xNormalStop THEN
    xActManual := TRUE;
END_IF

IF xStart AND NOT stMainToModules.q_xFastStop AND NOT stMainToModules.q_xNormalStop THEN
    xActAuto := TRUE;
END_IF

IF G_stHMI.xPrepareStart AND NOT stMainToModules.q_xFastStop AND NOT stMainToModules.q_xNormalStop THEN
    xActPrepare := TRUE;
END_IF
```

*Figur 26. Beroenden för stegbyte.*

När ett av de tre lägena är aktiva och ett stopp anropas utav någon av modulerna lämnas läget i diagrammet och går till efterföljande steg. Figur 27 visar flödet genom SFCn vid ett normalstopp och sekvensen lyder på följande sätt:

1. Main är i ModeSelect och får inmatningen att aktivera driftläget auto. Detta utlöser hoppet till Auto-steget.
2. I auto körs roboten i någon utav de olika rörelsesekvenserna tills en användare anropar ett normalstopp via modul 1. Modul 1 meddelar stoppet till main som därefter förmedlar vidare budskapet till alla anslutna moduler i listan astModules. När main kommunicerar att ett stop är efterfrågat triggas programkoden i steget Auto. Koden genomför hoppet till nästa steg i sekvensdiagrammet, i detta fall AutoStop.

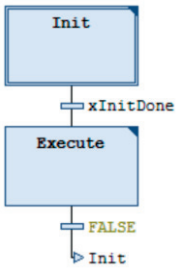
3. Steget AutoStop är aktivt tills alla moduler rapporterar att de genomfört ett stopp, när detta genomförts återgår SFCn till idle i ModeSelect.




*Figur 27. Beskrivning av flödet genom SFC i ModeSelect vid normalstopp*

## 5.8 RobotNGSFC

Programmet SR_RobotNGSFC består av tre delar. Huvuddelen är ett SFC-program, se figur 28, som i sin tur består av ett initieringsprogram, Init, och det aktiva programmet Execute som innehåller all kod involverande robotstyrning. SR_RobotNGSFC har en variabellista som är tillgänglig i både Init och Execute.



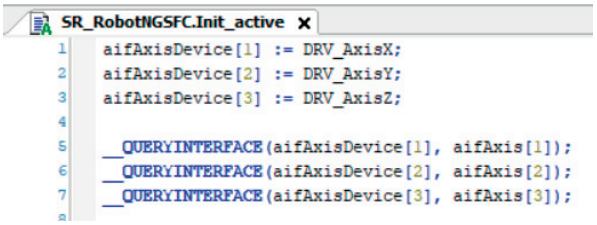

*Figur 28. SFC för robotbibliotek*

### 5.8.1 Init

Initieringen Init är skriven i tre faser:

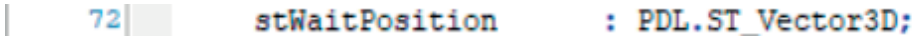
1. Initiering av axlarna
2. Konstruktion av rörelsekoordinater och kalkylering av rörelsemönster
3. Uppbyggnad av robotparametrar

Axlarna initieras genom att lagra axelenheterna i två olika listor, aifAxisDevice och aifAxis. aifAxisDevice pekar på servodrivenheten som helhet och aifAxis pekar via ett __queryinterface specifikt på axeln i enheten. Genom att nyttja __queryinterface behövs hårdvaran inte definieras flera gånger i koden vilket underlättar vid byte av hårdvara. Figur 29 visar hur värdena tilldelas.

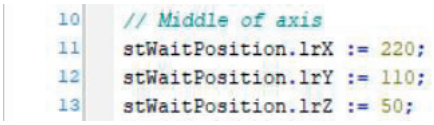



*Figur 29. Initiering av axlar*

Konstruktionen av rörelsekoordinater byggs upp via lagring av värden som pekar på en tredimensionell punkt. Denna punkt är av typen ST_Vector3D och de tre planen i punkten är de tre axlarna, X, Y och Z. Figur 30 är ett exempel på hur en vektor är konstruerad och i figur 31 visas hur koordinaterna tilldelas till vart plan.



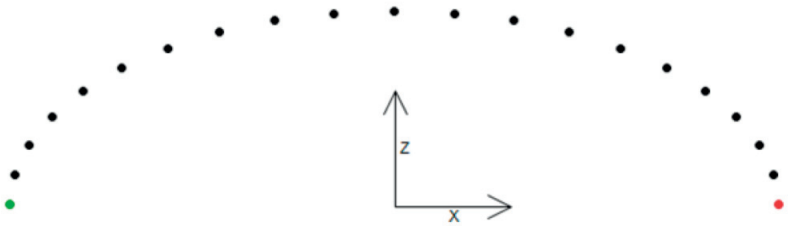

*Figur 30. ST_Vector3d variabelkonstruktion*




*Figur 31. Tilldelning av koordinater*

Dessa vektorer används i robotbibliotekets metoder för rörelse och genom att lagra dem i förhand kan rörelsemönster byggas upp användarvänligt.

En rörelse som inte är linjär behöver beräknas för att vara synkroniserad mellan de tre axlarna. I projektet kalkyleras spline-funktioner för att skapa ellipsrörelser med funktionsblocket FB_EllipticSpline från robotbiblioteketet. För att spara beräkningskraft kalkyleras detta en gång under initieringen istället för att göra kalkyleringen var uppdateringscykel.




*Figur 32. Splinekurva [22]*

Funktionsblockets inparametrar är tredimensionella vektorer för start och mål samt avståndet till ellipsens apex. Detta funktionsblock kalkylerar enbart en rörelse baserad på Z-axelns apex. Utparametern som används i projektet är den kalkylerade rörelsen från start till mål. Denna rörelse är av typen ST_SplineTable som är en tabell med beräknade tredimensionella vektorer som axlarna ska passera för att skapa en halvellips. Figur 32 visar ett exempel på en kurva och dess beräknade punkter från tillhörande splinetabellen och figur 33 visar hur funktionsblocket används i koden med ett apex på distansen 100 användarenheter.

```
56    fbSpline.CalcFullSpline(        i_stStart:= stPnPPick1,
57                                    i_stTarget:= stPnPPick2,
58                                    i_lrAbsHeight:= 100,
59                                    q_etDiag=> etDiag,
60                                    q_etDiagExt=> etDiagExtROB,
61                                    q_sMsg=> ,
62                                    q_stForward=> stSpline12,
63                                    q_stReverse=> ,
64                                    q_udiApexIndex=> ,
65                                    q_lrApexScalingFactor=> ,
66                                    q_lrSplineLength=> );
```

*Figur 33. Exempel på splinekalkylering*

Efter att rörelseparametrarna konstruerats, definieras roboten i funktionsblocket FB_Robot. Först skapas en kartesisk robot och dess axlar tilldelas, Figur 34, till följt av att hantering av nödbroms definieras, Figur 35. Uppstår ett problem under initieringen av dessa parametrar flaggas ett larm med ett meddelande för var felet uppstått. Dessa larm är synliga i applikationsloggen. Uppstår inga larm under initieringen lämnar programmet Init-steget och övergår till Execute.

```
111   fbRobot.ifConfiguration.Cartesian3Ax(   i_ifDriveA:= aifAxis[1].Axis,
112                                           i_ifDriveB:= aifAxis[2].Axis,
113                                           i_ifDriveC:= aifAxis[3].Axis,
114                                           q_etDiag=> etDiag,
115                                           q_etDiagExt=> etDiagExtROB,
116                                           q_sMsg=> sMsg);
```

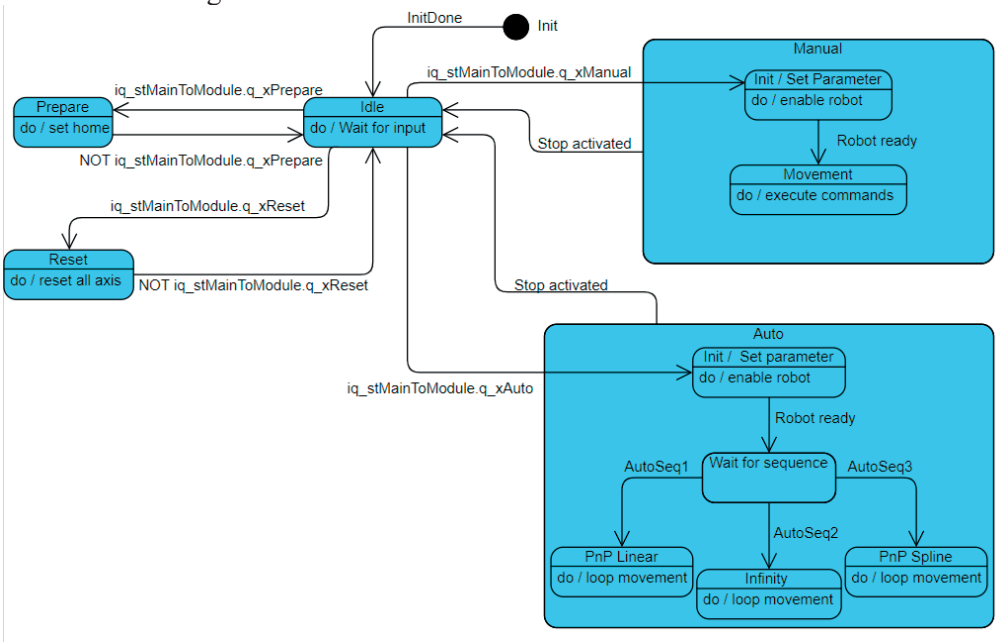
*Figur 34. Init, cartesian*

```
IF etDiag = GD.ET_Diag.Ok THEN

    fbRobot.ifConfiguration.SetEmergencyParameter(  i_lrMaxDeceleration:= 30000,
                                                    i_lrRamp:= 1.0,
                                                    q_etDiag=> etDiag,
                                                    q_etDiagExt=> etDiagExtROB,
                                                    q_sMsg=> sMsg);
    IF etDiag <> GD.ET_Diag.Ok THEN
        axAlarm[8] := TRUE;
        asAlarmMsg[1] := 'SetEmergencyParameter failed...';
    END_IF

END_IF
```

*Figur 35. Init, set emergency*

## 5.8.2 Execute

Execute är uppbyggt med en tillståndsmaskin som följer tillståndsdiagrammet i figur 36. När initieringen av SFC-programmen i main och i robotNGSFC är slutförd väntar tillståndsmaskinen i "Idle" på kommandon kommunicerat från de andra modulerna via main. Maskinen återkommer till idle när SFCn i main återvänder till steget ModeSelect.




*Figur 36. Tillståndsdiagram för tillståndsmaskin*

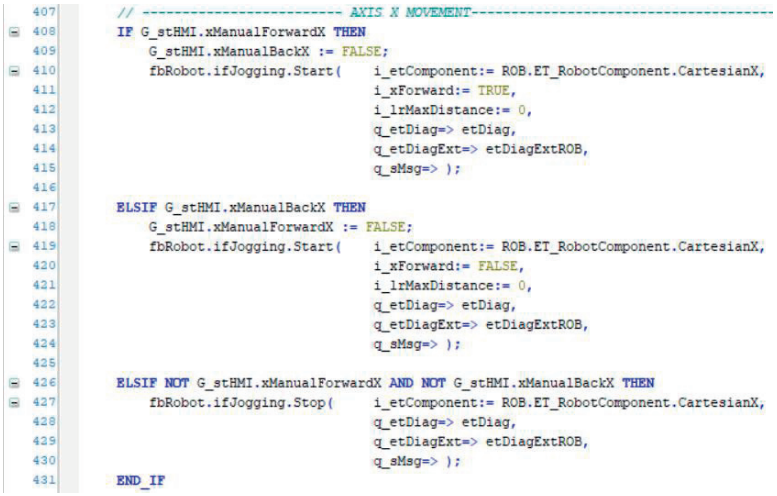
Vid första uppstart måste en förberedelse av roboten köras som genomförs i tillståndet “Prepare”. Prepares funktion är att synkronisera axlarna och definiera deras position i dess encoder, om detta inte utförs kan roboten skadas vid drift då inmatade koordinater kan finnas utanför axlarnas fysiskt möjliga område. Förberedelsen initieras via HMI:t efter att axlarna positionerats i ett läge som ska tolkas som koordinaten “1”. Positioneringen genomförs fysiskt eller via manuell drift i HMI. När förberedelsen är startad övergår SFCn i main från steget ModeSelect till Prepare och när förberedelsen är slutförd återgår tillståndsmaskinen till idle när ModeSelect är aktivt i main igen.

Tillståndet “Reset” återställer de tre axlarna genom ett funktionsblock, MC_Reset från biblioteket PLCO. En återställning krävs efter inträffat nödstopp eller annat utlöst larm och initieras via RFID-systemet, Harmony huben eller HMI. När återställningen är genomförd återvänder programmet till tillståndet idle.

Skiftet till “Manual” och “Auto” i tillståndsdiagrammet sker när inmatning av användaren utlöser ett stegskifte från ModeSelect till “Manual” respektive “Auto” i main. Manual är endast körbart via HMI medans auto kan aktiveras via start- och stoppknapp, HMI och RFID-taggar. Vilken utav de tre sekvenserna som aktiveras är beroende på valt läge i HMI, specifikt läge knutit till tagg eller föregående kört läge om auto aktiveras med startknapp.

Innan manuell och autodrift påbörjas initieras axlarna genom att servodrivenheterna aktiveras för spänningssättning av servomotorerna. Vid lyckad aktivering matas hastighet- och accelerationsparametrar in till axlarna som sedan går vidare för att invänta driftkommandon då föregående steg genomförts felfritt.

Vid manuell drift används joggning för drift. Joggning tillåter en rörelse mot en riktning utan behovet av specifika koordinater som inmatning för rörelse. I detta läge arbetar var axel utan synkronisering mot de andra och riktning av joggning är beroende på inmatning från HMI. Exempel på hur driften är programmerad för styrning via HMI till en axel demonstreras i figur 37.



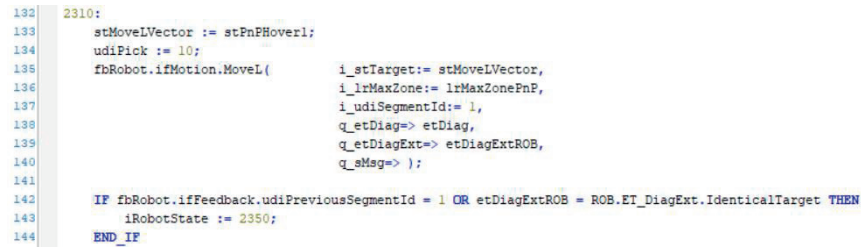
```
// ------------------------ AXIS X MOVEMENT-------------------------------------
IF G_stHMI.xManualForwardX THEN
    G_stHMI.xManualBackX := FALSE;
    fbRobot.ifJogging.Start(    i_etComponent:= ROB.ET_RobotComponent.CartesianX,
                                i_xForward:= TRUE,
                                i_lrMaxDistance:= 0,
                                q_etDiag=> etDiag,
                                q_etDiagExt=> etDiagExtROB,
                                q_sMsg=> );

ELSIF G_stHMI.xManualBackX THEN
    G_stHMI.xManualForwardX := FALSE;
    fbRobot.ifJogging.Start(    i_etComponent:= ROB.ET_RobotComponent.CartesianX,
                                i_xForward:= FALSE,
                                i_lrMaxDistance:= 0,
                                q_etDiag=> etDiag,
                                q_etDiagExt=> etDiagExtROB,
                                q_sMsg=> );

ELSIF NOT G_stHMI.xManualForwardX AND NOT G_stHMI.xManualBackX THEN
    fbRobot.ifJogging.Stop(     i_etComponent:= ROB.ET_RobotComponent.CartesianX,
                                q_etDiag=> etDiag,
                                q_etDiagExt=> etDiagExtROB,
                                q_sMsg=> );
END_IF
```

*Figur 37. Manuell drift X axel.*

En rörelsesekvens är uppbyggd med flertalet anrop av metoder för att skapa ett mönster. Som exempel, om en rörelse ska arbeta fram och tillbaka mellan två punkter anropas först en rörelse från start till önskad destination. Dessa koordinater matas in i metoden tillsammans med ett unikt identifikationsnummer för sekvensen. När rörelsen slutförts matas nästa rörelse in som är från destinationen till startkoordinaterna med ett nytt ID-nummer. Detta återupprepas sedan för att få en sömlös rörelse.

De olika rörelsesekvenserna i driftläget auto är skrivet med tre olika metoder från robotbiblioteket.

1. Den linjära plocka och placera sekvensen är skriven med metoden MoveL som visas enligt figur 38. MoveL genomför en linjär rörelse mot målkoordinaterna. Denna rörelse är synkroniserad i de tre axlarna på det sätt att hastigheten är kalkylerad för att var axel ska nå målet från vardera startposition vid samma tidpunkt. Inparametern i_lrMaxZone definierar felmarginalen rörelsen tillåts ha mot målkoordinaterna i en sömlös rörelse.



```
2310:
    stMoveLVector := stPnPHover1;
    udiPick := 10;
    fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
                                    i_lrMaxZone:= lrMaxZonePnP,
                                    i_udiSegmentId:= 1,
                                    q_etDiag=> etDiag,
                                    q_etDiagExt=> etDiagExtROB,
                                    q_sMsg=> );

    IF fbRobot.ifFeedback.udiPreviousSegmentId = 1 OR etDiagExtROB = ROB.ET_DiagExt.IdenticalTarget THEN
        iRobotState := 2350;
    END_IF
```

*Figur 38. MoveL*

2. Rörelsen som efterliknar ett oändlighetstecken i X- och Y-led är programmerat med metoden MoveC. Denna metod är skriven som i figur 39. MoveC beräknar en rörelse från start till mål med definierade punkter rörelsen ska passera via en cirkulär rörelse. Denna metod kallas på nytt med nya koordinater fyra gånger med skärningspunkten som utgångspunkt, en gång för var halvcirkel i formen av en "8".

```
2510:
    lrCx := 60.0;
    lrTx := lrCx * 2;

    stCircular := stWaitPosition;
    stCircular.lrX := stCircular.lrX - lrCx;
    stCircular.lrY := stCircular.lrY + lrCx;

    stMoveCVector := stWaitPosition;
    stMoveCVector.lrX := stMoveCVector.lrX - lrTx;
    stMoveCVector.lrY := stMoveCVector.lrY + 0.0;

    fbRobot.ifMotion.MoveC( i_stTarget:= stMoveCVector,
                            i_stCircular:= stCircular,
                            i_lrMaxZone:= lrMaxZone,
                            i_udiSegmentId:= 1,
                            q_etDiag=> etDiag,
                            q_etDiagExt=> etDiagExtROB,
                            q_sMsg=> sMsg);

    iRobotState := 2520;
```

*Figur 39. MoveC*

3. Plocka och placera rörelsen med elliptisk kurva i Z-led är kodad med metoden MoveS. Metodens inmatning är målkoordinater och en splinetabell vilket visas i figur 40. Denna tabell kalkylerades i Init steget och kombineras med matchande mål för att genomföra rörelsen mellan start och mål.

```
2710:
    fbRobot.ifMotion.MoveS(         i_stTarget:= stPnPPick2,
                                    i_stSplineTable:= stSpline12,
                                    i_lrMaxZone:= lrMaxZonePnP,
                                    i_udiSegmentId:= 1,
                                    q_etDiag=> etDiag,
                                    q_etDiagExt=> etDiagExtROB,
                                    q_sMsg=> );

    IF fbRobot.ifFeedback.udiPreviousSegmentId = 1 THEN
        iRobotState := 2720;
    END_IF
```

*Figur 40 MoveS.*

Utanför tillståndsmaskinen finns även kod i SR_RobotNGSFC som hanterar larm och stopp, felhantering av axlar och kommunikationen till main. För att uppdateras i varje cykel i programmet måste dessa vara placerade utanför alla tillstånd. Stopphanteringen är uppbyggd enligt figur 41 och varierar beroende på vilken typ

av stopp som är efterfrågad. Ett efterfrågat snabbstopp avaktiverar roboten vilket under rörelse aktiverar nödinbromsningen som definierades i Init. När detta skett hoppar tillståndet till idle i tillståndsmaskinen och modulen, som i detta fall är roboten, meddelar main att snabbstoppet är genomfört. Ett normalstopp låter den aktuella rörelsen arbeta färdigt innan den hoppar till idle och meddelar att den är färdig, ingen inbromsningsåtgärd sker.

```
489  IF iq_stMainToModule.q_xFastStop THEN
490      fbRobot.xEnable := FALSE;
491      IF NOT fbRobot.ifFeedback.xInMotion THEN
492          fbRobot.xStart := FALSE;
493          xPowerEnableAll := FALSE;
494
495          iq_stModuleToMain.q_xFastStopDone := TRUE;
496          iRobotState := 0;
497      END_IF
498
499  ELSIF iq_stMainToModule.q_xNormalStop THEN
500
501      IF NOT fbRobot.ifFeedback.xInMotion THEN
502          iq_stModuleToMain.q_xNormalStopDone := TRUE;
503
504          iRobotState := 0;
505      END_IF
506  END_IF
```

*Figur 41. Kod för stopphantering.*

## 5.9 PowerMeter

Energimätaren ärvdes av det befintliga projektet och då det fanns problem med kommunikationen via Acti9 fick energimätaren kopplas upp via seriell anslutning och byggas upp från grund i Machine Expert. I programmets hårdvaruträd installerades en generisk modul och för att kunna avläsa rätt värden användes mätarens datablad [10] för att ta reda på vilka bitar som visade energiavläsning. Figur 42 visar hur sex olika kanaler är uppbyggda för att avläsa olika bitområden i den seriella kommunikationen från energimätaren.

| | Name | Access Type | Trigger | READ Offset | Length | Error Handling |
|---|---|---|---|---|---|---|
| 0 | Channel 1 | Read Holding Registers (Function Code 03) | Cyclic, t#500ms | 16#0BB7 | 6 | Keep last Value |
| 1 | Channel 2 | Read Holding Registers (Function Code 03) | Cyclic, t#500ms | 16#0BCB | 14 | Keep last Value |
| 2 | Channel 3 | Read Holding Registers (Function Code 03) | Cyclic, t#500ms | 16#0BED | 8 | Keep last Value |
| 3 | Channel 4 | Read Holding Registers (Function Code 03) | Cyclic, t#500ms | 16#0C0B | 2 | Keep last Value |
| 4 | Channel 5 | Read Holding Registers (Function Code 03) | Cyclic, t#500ms | 16#0C25 | 2 | Keep last Value |
| 5 | Channel 6 | Read Holding Registers (Function Code 03) | Cyclic, t#500ms | 16#0C83 | 4 | Keep last Value |

*Figur 42. Avläsningskanaler för energimätaren.*

Detta gjorde det möjligt att via programmet SR_PowerMeter kartlägga avläsningarna för att beräkna värdena för energiavläsningen. Värdena beräknas med två funktioner som tilldelades av handledare och är hämtade från den gamla programvaran för projektet. Möjliga värden att avläsa är:

- Fasspänning
- Spänning mellan fas till fas
- Genomsnittlig spänning
- Fasström
- Aktiv effekt per fas
- Reaktiv effekt per fas
- Effektfaktor
- Frekvens
- Totalt förbrukad energi

Roboten är enbart installerad på en fas men alla avläsningar implementerades för möjligheten av expansion.

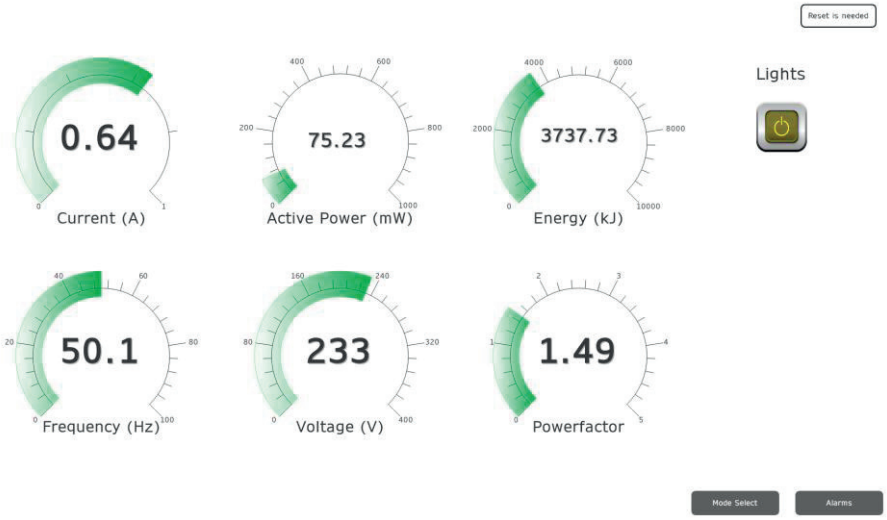
För att effektivisera koden något ändrades uppdateringsfrekvensen för SR_PowerMeter. Kravet på uppdateringen av energivärdena är ej lika hög som resterande program i koden då variationen på värdena ej fluktuerar mycket. SR_PowerMeter är programmerat med en uppdateringscykel på 1 s skilt från de 10 ms som resterande program uppdateras i.

## 5.10 HMI

Det grafiska gränssnittet, Human Machine Interface (HMI), består av fyra skärmar där varje skärm uppfyller en individuell funktion. Navigering av skärmarna är implementerad på vardera skärm med enkla knappar tillsammans med en beskrivande text som informerar vilken skärm knappen refererar till. En indikation, för att en återställning av systemet är nödvändig, är utplacerad på samtliga skärmar.

### 5.10.1 MainScreen

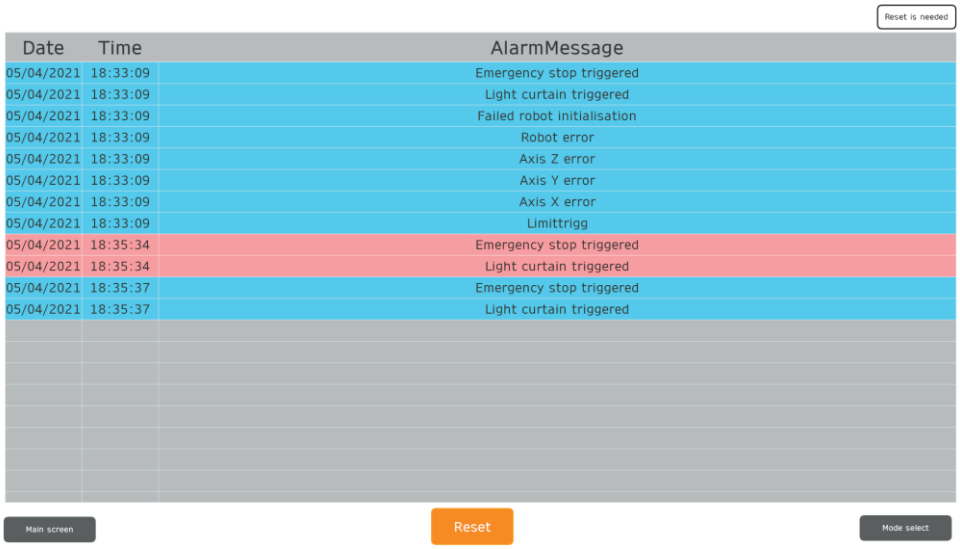
Visualisering av energiförbrukning och övriga data för elskåpet visas på huvudskärmen, vilket kan ses i figur 43. I översta högra hörnet lokaliseras indikationen för återställning av systemet, här är även en av- och påknapp utplacerad för påslagning av ljusbelysning.




*Figur 43. Mainscreen*

### 5.10.2 AlarmList

En larmlista över aktiverade och utlösta larm är skapad för den andra skärmen. De olika färgerna på larmmeddelandena symboliserar ett larm som är aktivt respektive ett larm som är utlöst, vilken illustreras i figur 44. Larmmeddelandena i blå färg refererar till de aktiverade larmen och de röda meddelandena indikerar de utlösta larmen. När ett larm utlöses ska orsaken åtgärdas för att sedan använda återställningsknappen för återaktivering av larmen.

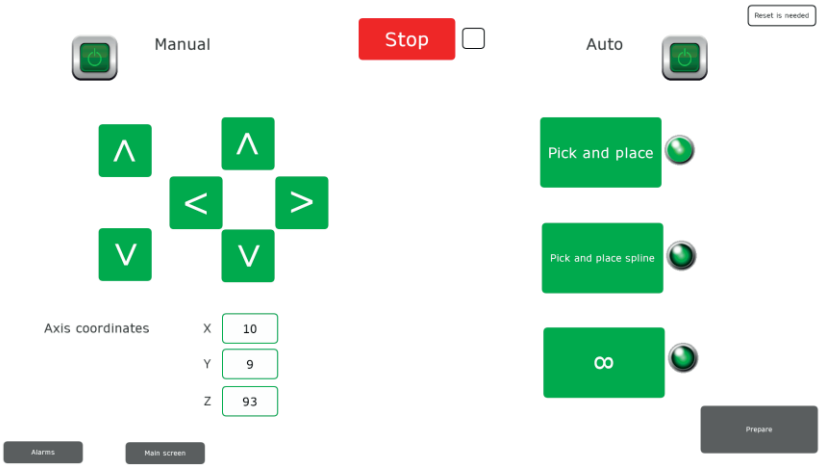



*Figur 44. Larmlistan*

### 5.10.3 ModeSelect

Valmöjligheter för styrning i manuellt respektive automatiskt läge är implementerat i den tredje skärmen. Manuell körning startas genom en av- och påknapp och önskad rörelse på axlarna bestäms genom knapptryckningar på pilarna, vilka refererar till riktningarna på samtliga axlar. Vid manuell körning visualiseras även styraxlarnas aktuella koordinater.

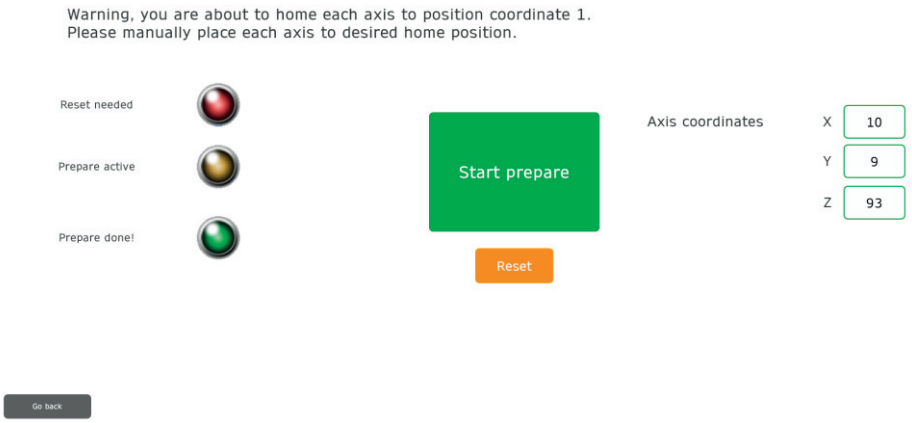
Det automatiska körläget initieras med en separat av- och påknapp, därefter följer val av rörelsesekvens. Den valda sekvensen markeras med en lysande lampa. Figur 45 visar den händelsen i fallet där användaren har valt rörelsesekvensen “pick and place”. En gemensam stoppknapp för både det manuella och automatiska läget är placerad på skärmen med en indikator med funktionen att blinka från och med att en stoppsekvens är initierad till och med att den är avslutad. Skärmen fyller ytterligare en funktion, alternativet att utföra en prepare-sekvens vilket förklaras mer detaljerat i avsnitt 5.10.4.




*Figur 45. Modeselect*

## 5.10.4 Prepare

Den fjärde och sista skärmen är implementerad med prepare-funktionen. En informativ text beskriver vad som kommer att ske under initieringen, med ändamål att göra användaren medveten. Styraxlarna ska ändra position till de önskade koordinaterna och detta sker med hjälp av de numeriska rutorna med robotens aktuella koordinater. Tre lampor är utplacerade och agerar indikatorer för tre olika scenarion. Den första röda lampan indikerar att en återställning av systemet måste ske innan en prepare kan initieras, den gula indikerar att initiering är påbörjad och aktiv, den gröna lyser när metoden är färdig. Det här kan ses i figur 46.




*Figur 46. Prepare*

# 6. Slutsats

I följande avsnitt presenteras slutsatsen för hela examensarbetet där svar på problemformuleringarna från avsnitt 1.4 ges.  Även en kort reflektion över de relevanta etiska aspekterna redogörs. Här framförs även en diskussion kring framtida utvecklingsmöjligheter för examensarbetet.

En portalrobot har programmerats med implementering av det nya robotbiblioteket och styrning av den sker via ett egenkonstruerat Human Machine Interface (HMI). Portalroboten är körbar i flera olika driftlägen och rörelsemönster, vilket går att styra via det grafiska gränssnittet. Från resultatet går det att svara på tre av de fem originala problemformuleringarna presenterade i den inledande delen av rapporten för examensarbetet.

En problemformulering var om det var möjligt att implementera det nya robotbiblioteket på den nya portalroboten, vilket i resultatet påvisades att så var fallet. Det är möjligt att implementera det nya robotbiblioteket om styrsystemet är kompatibelt mot det, vilket realiserades när bytet av PLC-modul tog plats. I resultatet, avsnitt 5.10, presenteras slutresultatet och det färdigimplementerade gränssnitten, Human Machine Interface (HMI), vilket svarar på frågan om ett HMI kan implementeras. Styrning via ett HMI är möjligt att implementera och styrning via det har realiserats.

Problemet angående internetuppkopplingen, vilket redovisades i avsnitt 4.5.1, orsakade att de två resterande problemformuleringarna saknar svar. Utan internet anslutet till konstruktionen omöjliggjordes uppkoppling mot det digitala molnverktyget, Machine Advisor. Det medförde att skapandet av en digital tvilling var inte var genomförbart.

## 6.1 Reflektion över etiska aspekter

Ett lyckat examensarbete med en fullt fungerande portalrobot bidrar till att företaget kan använda den i utbildningssyfte som en demorobot. En demorobot kan bidra till en förståelse för hur industrirobotar fungerar och gör det även möjligt för eventuella kunder och studenter att personligen få testköra den. Det kan också bidra till att ett intresse för automationsteknik växer. En blick in i den automatiserade världen bidrar i sin tur till en bredare förståelse för den automatisering som sker i samhället idag.

## 6.2 Framtida utvecklingsmöjligheter

Potentialen för framtida utvecklingsmöjligheter är stor då många beslut under examensarbetet har tagits med den tanken närvarande. Koden är skriven med möjligheten för enkel expansion i framtiden, då flera olika rörelsemönster kan implementeras och realiseras. Hårdvara har även bibehållits i elskåpets konstruktion för att inte förhindra eventuell expansion av robot, med tillhörande styrsystem, i framtiden.

Internetuppkoppling är något som kommer att behöva implementeras för framtida användning av systemet då flera nya funktioner blir tillgängliga för användning. Uppkoppling mellan PLC och nätet bidrar till att implementeringen av den digitala tvillingen kan realiseras med molnverktyget Machine Advisor.

Ett försök till installation av en 360-graders kamera påbörjades under programmeringsfasen som senare fick avbrytas då modellen på den tilldelade kameran var föråldrad. Problemet resulterade i att kamerans mjukvara för videoströmning inte gick att implementera i projektet och tiden inte räckte till för att finna en lösning eller införskaffa en nyare kamera. Beslutet togs att lämna idén om en kamera till vidareutveckling i framtiden.

För kontroll och övervakning kan en ny 360-graders kamera installeras ovanför roboten, i den skyddande buren. Därefter kan en livestream etableras i HMI för övervakning av roboten i realtid.

# 7. Terminologi

| | |
|---|---|
| HMI | Human Machine Interface |
| PLC | Programmable Logic Controller |
| PLCO | PLC open |
| I/O-system | Input/Output-system |
| SFC | Sequential Function Block |
| ST | Strukturerad Text |

# 8. Källförteckning

[1] G. S. Bell and W. J. Wilson, "Coordinated controller design for position based robot visual servoing in Cartesian coordinates," Proceedings of IEEE International Conference on Robotics and Automation, 1996, pp. 1650-1655 vol.2, doi: 10.1109/ROBOT.1996.506949.

[2] E. Tisserant, L. Bessard and M. de Sousa, "An Open Source IEC 61131-3 Integrated Development Environment," 2007 5th IEEE International Conference on Industrial Informatics, 2007, pp. 183-187, doi: 10.1109/INDIN.2007.4384753.

[3]Schneider Electric(2021), *Modicon M262 IIoT-ready logic/motion controller for performance machines.*
URL:https://download.schneider-electric.com/files?p_enDocType=Catalog&p_File_Name=Catalog+Modicon+M262+-+IIoT-ready+logic+%26+motion+controller+for+performances+machines+-+May+2021.pdf&p_Doc_Ref=DIA3ED2180503EN (Hämtad: 2021-05-18).

[4]Schneider Electric(2021), *Modicon M262.*
URL:https://www.se.com/ww/en/product-range-presentation/65771-modicon-m262/ (Hämtad: 2021-05-18).

[5]Schneider Electric(2019), *Lexium 32S Servo Drive User Guide.*
URL:https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=0198441114060.02.pdf&p_Doc_Ref=0198441114060-EN
(Hämtad: 2021-05-18).

[6]Schneider Electric(2021), *Lexium 32 & Motors*
URL:https://www.se.com/ww/en/product-range-presentation/2302-lexium-32-%26-motors/?selected-node-id=12367263866#tabs-top (Hämtad: 2021-05-18).

[7] Schneider Electric(u.å.), *RFID and Inductive Identification Systems.*
URL:https://tesensors.com/se/en/products/rfid-systems?range=1471&node=12145659794 (Hämtad: 2021-05-18).

[8]Schneider Electric(2021), *Harmony, Harmony Hub, wireless to modbus/TCP, ethernet gateway, 24...240 V AC/DC.*
URL:https://www.se.com/ww/en/product/ZBRN1/harmony-hub%2C-wireless-to-modbus-tcp%2C-ethernet-gateway%2C-24...240-v-ac-dc/ (Hämtad: 2021-05-18).

[9]Schneider Electric(2021), *Acti 9 iEM3000.*
URL:https://www.se.com/se/sv/product-range-presentation/61273-acti-9-iem3000/?parent-subcategory-id=4125&filter=business-4-l%C3%A5gsp%C3%A4nningsprodukter-och-system&selected-node-id=12146168476#tabs-top
(Hämtad: 2021-05-18).

[10]Schneider Electric(2019), *iEM3000series Energy meters User manual.*
URL:https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=DOCA0005EN-12.pdf&p_Doc_Ref=DOCA0005EN
(Hämtad: 2021-05-18).

[11]Schneider Electric(2016), *Magelis industrial PC.*
URL:https://download.schneider-electric.com/files?p_Doc_Ref=Magelis+iPC+e.book+-+New (Hämtad: 2021-05-18).

[12]Sercos International e.V.(2021), *What is Sercos?*
URL:https://www.sercos.org/technology/what-is-sercos/ (Hämtad: 2021-05-18).

[13] Lyche T., Manni C., Speleers H. (2018) Foundations of Spline Theory: B-Splines, Spline Approximation, and Hierarchical Refinement. In: Lyche T., Manni C., Speleers H. (eds) Splines and PDEs: From Approximation Theory to Numerical Linear Algebra. Lecture Notes in Mathematics, vol 2219. Springer, Cham. https://doi.org/10.1007/978-3-319-94911-6_1 (Hämtad: 2021-05-18).

[14]The MathWorks, Inc.(2021), *spline.*
URL:https://www.mathworks.com/help/matlab/ref/spline.html
(Hämtad: 2021-05-18).

[15]Schneider Electric(2019), *Lexium 32 and Motors Lexium 32 servo drives, BMH and BSH servo motors.*
URL:https://download.schneider-electric.com/files?p_enDocType=Catalog&p_File_Name=Catalog+Lexium+32+and+Motors++Lexium+32+servo+drives+BMH+and+BSH+servo+motors+-+March+2019.pdf&p_Doc_Ref=DIA7ED2140501EN (Hämtad: 2021-05-18).

[16]Schneider Electric(2020), *RoboticModule Library Guide.*
URL:PD.Lib.RoboticModule/index.htm#t=PD.Lib.RoboticModule%2Ffront%2Ffront-1.htm (Hämtad: 2021-05-18).

[17]Schneider Electric(2019), *EcoStruxureTM Machine Expert A single software environment to automate machines.*
URL:https://download.schneider-electric.com/files?p_enDocType=Catalog&p_File_Name=Catalog+EcoStruxure+Machine+Expert+A+single+software+environment+to+automate+machines.pdf&p_Doc_Ref=DIA3ED2180701EN
(Hämtad: 2021-05-18)

[18]Schneider Electric(2021), *EcoStruxure Machine Expert (SoMachine).*
URL:https://www.se.com/ww/en/product-range-presentation/2226-ecostruxure-machine-expert-%28somachine%29/?selected-node-id=14435955187#tabs-top
(Hämtad: 2021-05-18)

[19]Schneider Electric(2021), *EcoStruxure™ Operator Terminal Expert.*
URL:https://www.se.com/ww/en/product-range-presentation/62621-ecostruxure%E2%84%A2-operator-terminal-expert/ (Hämtad: 2021-05-18)

[20]Schneider Electric(), *SoMove.*
URL:https://www.se.com/se/sv/product-range-presentation/2714-somove/
(Hämtad: 2021-05-18)

[21]Schneider Electric(2021), *EcoStruxure™ Machine Advisor.*
URL:https://www.se.com/se/sv/work/services/field-services/industrial-automation/oem/machine-advisor.jsp (Hämtad: 2021-05-18)

[22]Schneider Electric(2020), *Robotic Library Guide, FB_EllipticSpline - CalcFullSpline (Method).*
URL:https://product-help.schneider-electric.com/Machine%20Expert/V1.1/en/PD.Lib.Robotic/index.htm#t=PD.Lib.Robotic%2FFB_EllipticSpline%2FFB_EllipticSpline-3.htm&rhsearch=ellip&rhsyns=%20&SUBSTRSRCH=1
(Hämtad: 2021-05-18)

# 9. Appendix

## 9.1 SR_RobotNGSFC

### 9.1.1 Variabellista

```
1 PROGRAM SR_RobotNGSFC
2 VAR_IN_OUT
3   iq_stMainToModule   : ST_MAINTOMODULES;
4   iq_stModuleToMain   : ST_MODULETOMAIN;
5 END_VAR
6 VAR
7   fbRobot             : ROB.FB_Robot;
8   fbSpline            : ROB.FB_EllipticSpline;
9
10  ifJogging           : ROB.IF_RobotJogging;
11
12  xInitDone           : BOOL;
13  xTest               : BOOL;
14  xPowerEnableAll     : BOOL;
15  xResetAll           : BOOL;
16  xSercosReset        : BOOL;
17
18  xAutoSeq1           : BOOL := TRUE;
19  xAutoSeq2           : BOOL;
20  xAutoSeq3           : BOOL;
21
22  etDiag              : GD.ET_Diag;
23  etDiagExtROB        : ROB.ET_DiagExt;
24  etDiagExtAPL        : APL.ET_DiagExt;
25
26  sMsg                : STRING;
27
28  axAlarm             : ARRAY[1..8] OF BOOL;
29  asAlarmMsg          : ARRAY[1..8] OF STRING;
30  xAlarmInit          : BOOL;
31
32  fbLoggerPoint       : APL.FB_LoggerPoint;
33
34  afbSetPos           : ARRAY[1..3] OF
FB_SetPositionMultiturn_SIII;
35  afbAxis             : ARRAY[1..3] OF FB_SEAxis;
36  aifAxis             : ARRAY[1..3] OF DAL.IF_AxisAccess;
37    aifAxisDevice     : ARRAY[1..3] OF DAL.IF_DeviceAccess;
38
```

```
39  iAxis               : INT;
40  iRobotState         : INT;
41  iLoop               : INT;
42
43  udiPick             : UDINT;
44
45  rHomePosition       : REAL;
46
47  lrCx                : LREAL;
48  lrTx                : LREAL;
49  lrMaxZone           : LREAL := 10;
50  lrMaxZonePnP        : LREAL := 0;
51
52  lrRefAxisX          : LREAL;
53  lrRefAxisY          : LREAL;
54  lrRefAxisZ          : LREAL;
55
56  xMaxZonePnPOFF      : BOOL := TRUE;
57  xMaxZonePnPON       : BOOL;
58
59  stMoveLVector       : PDL.ST_Vector3D;
60  stMoveCVector       : PDL.ST_Vector3D;
61  stMoveSVector       : PDL.ST_Vector3D;
62
63  stSpline12          : ROB.ST_SplineTable;
64  stSpline23          : ROB.ST_SplineTable;
65  stSpline34          : ROB.ST_SplineTable;
66  stSpline41          : ROB.ST_SplineTable;
67
68
69  stSplineTest        : PDL.ST_Vector3D;
70
71  stCircular          : PDL.ST_Vector3D;
72  stWaitPosition      : PDL.ST_Vector3D;
73
74  stPnPHover1         : PDL.ST_Vector3D;
75  stPnPHover2         : PDL.ST_Vector3D;
76  stPnPHover3         : PDL.ST_Vector3D;
77  stPnPHover4         : PDL.ST_Vector3D;
78
79  stPnPPick1          : PDL.ST_Vector3D;
80  stPnPPick2          : PDL.ST_Vector3D;
81  stPnPPick3          : PDL.ST_Vector3D;
82  stPnPPick4          : PDL.ST_Vector3D;
83
84  stPnPHoverZ         : PDL.ST_Vector3D;
85  stPnPPickZ          : PDL.ST_Vector3D;
```

```
86 END_VAR
```

### 9.1.2 Init_active

```
1  aifAxisDevice[1] := DRV_AxisX;
2  aifAxisDevice[2] := DRV_AxisY;
3  aifAxisDevice[3] := DRV_AxisZ;
4
5  __QUERYINTERFACE(aifAxisDevice[1], aifAxis[1]);
6  __QUERYINTERFACE(aifAxisDevice[2], aifAxis[2]);
7  __QUERYINTERFACE(aifAxisDevice[3], aifAxis[3]);
8
9  //------------- Movement coordinates -----------------
10 // Middle of axis
11 stWaitPosition.lrX := 220;
12 stWaitPosition.lrY := 110;
13 stWaitPosition.lrZ := 50;
14
15 // Homing coordinates
16 rHomePosition := 1.0;
17
18 // Pick and place coordinates
19 stPnPHoverZ.lrZ := 99;
20 stPnPPickZ.lrZ := 5;
21
22 stPnPHover1.lrX := 10;
23 stPnPHover1.lrY := 10;
24 stPnPHover1.lrZ := stPnPHoverZ.lrZ;
25
26 stPnPHover2.lrX := 400;
27 stPnPHover2.lrY := 200;
28 stPnPHover2.lrZ := stPnPHoverZ.lrZ;
29
30 stPnPHover3.lrX := 100;
31 stPnPHover3.lrY := 120;
32 stPnPHover3.lrZ := stPnPHoverZ.lrZ;
33
34 stPnPHover4.lrX := 380;
35 stPnPHover4.lrY := 30;
36 stPnPHover4.lrZ := stPnPHoverZ.lrZ;
37
38 stPnPPick1.lrX := 10;
39 stPnPPick1.lrY := 10;
40 stPnPPick1.lrZ := stPnPPickZ.lrZ;
41
42 stPnPPick2.lrX := 400;
43 stPnPPick2.lrY := 200;
```

```
44 stPnPPick2.lrZ := stPnPPickZ.lrZ;
45
46 stPnPPick3.lrX := 100;
47 stPnPPick3.lrY := 120;
48 stPnPPick3.lrZ := stPnPPickZ.lrZ;
49
50 stPnPPick4.lrX := 380;
51 stPnPPick4.lrY := 30;
52 stPnPPick4.lrZ := stPnPPickZ.lrZ;
53
54 IF etDiag = GD.ET_Diag.Ok THEN
55
56  fbSpline.CalcFullSpline(        i_stStart:= stPnPPick1,
57                                  i_stTarget:= stPnPPick2,
58                                  i_lrAbsHeight:= 100,
59                                  q_etDiag=> etDiag,
60                                  q_etDiagExt=> etDiagExtROB,
61                                  q_sMsg=> ,
62                                  q_stForward=> stSpline12,
63                                  q_stReverse=> ,
64                                  q_udiApexIndex=> ,
65                                  q_lrApexScalingFactor=> ,
66                                  q_lrSplineLength=> );
67
68  fbSpline.CalcFullSpline(        i_stStart:= stPnPPick2,
69                                  i_stTarget:= stPnPPick3,
70                                  i_lrAbsHeight:= 100,
71                                  q_etDiag=> etDiag,
72                                  q_etDiagExt=> etDiagExtROB,
73                                  q_sMsg=> ,
74                                  q_stForward=> stSpline23,
75                                  q_stReverse=> ,
76                                  q_udiApexIndex=> ,
77                                  q_lrApexScalingFactor=> ,
78                                  q_lrSplineLength=> );
79
80  fbSpline.CalcFullSpline(        i_stStart:= stPnPPick3,
81                                  i_stTarget:= stPnPPick4,
82                                  i_lrAbsHeight:= 100,
83                                  q_etDiag=> etDiag,
84                                  q_etDiagExt=> etDiagExtROB,
85                                  q_sMsg=> ,
86                                  q_stForward=> stSpline34,
87                                  q_stReverse=> ,
88                                  q_udiApexIndex=> ,
89                                  q_lrApexScalingFactor=> ,
90                                  q_lrSplineLength=> );
```

```
91
92  fbSpline.CalcFullSpline(        i_stStart:= stPnPPick4,
93                                  i_stTarget:= stPnPPick1,
94                                  i_lrAbsHeight:= 100,
95                                  q_etDiag=> etDiag,
96                                  q_etDiagExt=> etDiagExtROB,
97                                  q_sMsg=> ,
98                                  q_stForward=> stSpline41,
99                                  q_stReverse=> ,
100                                 q_udiApexIndex=> ,
101                                 q_lrApexScalingFactor=> ,
102                                 q_lrSplineLength=> );
103
104 IF etDiag <> GD.ET_Diag.Ok THEN
105     axAlarm[8] := TRUE;
106     asAlarmMsg[1] := 'Failed initialisation...';
107 END_IF
108 END_IF
109
110 IF etDiag = GD.ET_Diag.Ok THEN
111 fbRobot.ifConfiguration.Cartesian3Ax(   i_ifDriveA:=
aifAxis[1].Axis,
112                                         i_ifDriveB:=
aifAxis[2].Axis,
113                                         i_ifDriveC:=
aifAxis[3].Axis,
114                                         q_etDiag=> etDiag,
115                                         q_etDiagExt=>
etDiagExtROB,
116                                         q_sMsg=> sMsg);
117
118 IF etDiag <> GD.ET_Diag.Ok THEN
119     axAlarm[8] := TRUE;
120     asAlarmMsg[1] := 'Failed initialisation...';
121 END_IF
122
123 END_IF
124
125 IF etDiag = GD.ET_Diag.Ok THEN
126 fbLoggerPoint.RegisterLoggerPoint(  i_ifParent:=
SR_Main.fbLoggerPoint,
127                                     i_sName:= 'SR_RobotNG',
128                                     i_sType:= '',
129                                     i_sSource:= '',
130                                     q_etDiag=> etDiag,
131                                     q_etDiagExt=>
etDiagExtAPL);
```

```
132
133 IF etDiag <> GD.ET_Diag.Ok THEN
134     axAlarm[8] := TRUE;
135     asAlarmMsg[1] := 'Failed initialisation...';
136 END_IF
137
138 END_IF
139
140 IF etDiag = GD.ET_Diag.Ok THEN
141 fbRobot.RegisterLoggerPoint(    i_ifParent:= fbLoggerPoint,
142                                 i_sName:= 'Robot',
143                                 q_etDiag=> etDiag,
144                                 q_etDiagExt=> etDiagExtROB);
145 IF etDiag <> GD.ET_Diag.Ok THEN
146     axAlarm[8] := TRUE;
147     asAlarmMsg[1] := 'Failed initialisation...';
148 END_IF
149
150 END_IF
151
152 IF etDiag = GD.ET_Diag.Ok THEN
153
154 fbRobot.ifConfiguration.SetEmergencyParameter(
i_lrMaxDeceleration:= 30000,
155                                                 i_lrRamp:= 1.0,
156                                                 q_etDiag=>
etDiag,
157                                                 q_etDiagExt=>
etDiagExtROB,
158                                                 q_sMsg=> sMsg);
159 IF etDiag <> GD.ET_Diag.Ok THEN
160     axAlarm[8] := TRUE;
161     asAlarmMsg[1] := 'SetEmergencyParameter failed...';
162 END_IF
163
164 END_IF
165
166 IF etDiag = GD.ET_Diag.Ok THEN
167
168 fbRobot.ifConfiguration.ConfigDone( q_etDiag=> etDiag,
169                                     q_etDiagExt=> etDiagExtROB,
170                                     q_sMsg=> sMsg);
171
172 IF etDiag <> GD.ET_Diag.Ok THEN
173     axAlarm[8] := TRUE;
174     asAlarmMsg[1] := 'Configuration failed...';
175 END_IF
```

```
176
177 END_IF
178
179
180
181 ifJogging := fbRobot.ifJogging;
182
183 xInitDone := NOT axAlarm[8];
```

### 9.1.3 execute_active

```
1  //--------------------- ROBOT MOTION --------------------------
2
3  (*
4  0: Idle
5  10-20: Reset
6  30-35: Normal stop
7  40-45: Emergency stop
8  50-55: Axis Error
9  1xx: Prepare mode
10 2xx: Auto mode
11  200-225: Init and sequence select
12      2310-2360: Pick n Place linear movement
13      2500-2545: XY infinity movement
14      2700-2745: Pick n Place spline movement
15 3xx: Manual mode
16  300-320: Intiate
17      330: Manual run
18 *)
19
20 CASE iRobotState OF
21 //----------------- IDLE -------------------------
22 0:
23  iq_stModuleToMain.q_xFastStopDone := FALSE;
24  iq_stModuleToMain.q_xNormalStopDone := FALSE;
25
26  IF iq_stMainToModule.q_xReset THEN
27      xResetAll := FALSE;
28      iRobotState := 10;
29
30  ELSIF iq_stMainToModule.q_xPrepare THEN
31      G_stHMI.xPrepareDone := FALSE;
32      iRobotState := 100;
33
34  ELSIF iq_stMainToModule.q_xAuto THEN
35      iRobotState := 200;
36
```

```
37  ELSIF iq_stMainToModule.q_xManual THEN
38      iRobotState := 300;
39
40  END_IF
41 //-------------------- RESET --------------------------
42 10:
43  xResetAll := TRUE;
44
45  IF afbAxis[1].q_xResetDone AND afbAxis[2].q_xResetDone AND
afbAxis[3].q_xResetDone THEN
46      xResetAll := FALSE;
47      iRobotState := 20;
48  END_IF
49 20:
50  IF NOT iq_stMainToModule.q_xReset THEN
51      iRobotState := 0;
52  END_IF
53
54 //---------------------- PREPARE MODE ------------------------
55 100:
56  fbRobot.xEnable := FALSE;
57  xPowerEnableAll := FALSE;
58
59  IF NOT afbAxis[1].i_xPowerEnable AND NOT
afbAxis[2].i_xPowerEnable AND NOT afbAxis[2].i_xPowerEnable THEN
60      iRobotState := 110;
61  END_IF
62
63 110:
64  FOR iAxis := 1 TO 3 DO
65      afbSetPos[iAxis].i_rPosition := rHomePosition;
66      afbSetPos[iAxis].i_xExecute := TRUE;
67  END_FOR
68
69  IF afbSetPos[1].q_xDone AND afbSetPos[2].q_xDone AND
afbSetPos[3].q_xDone THEN
70      FOR iAxis := 1 TO 3 DO
71          afbSetPos[iAxis].i_xExecute := FALSE;
72          aifAxis[iAxis].Axis.xIsHomed := TRUE;
73      END_FOR
74
75      iRobotState := 120;
76  END_IF
77 120:
78  IF SR_Main.axAlarm[2] THEN
79      G_stHMI.xPrepareDone := TRUE;
80      iq_stModuleToMain.q_xPrepareDone := TRUE;
```

```
81
82      iRobotState := 130;
83  END_IF
84 (*
85  xSercosReset := FALSE;
86  IF SercosMaster.SercosPhaseChanger.ActualState =
S3M.ET_SercosState.Error THEN
87      xSercosReset := TRUE;
88  ELSIF SercosMaster.SercosPhaseChanger.ActualState =
S3M.ET_SercosState.Phase4 THEN
89      G_stHMI.xPrepareDone := TRUE;
90      iq_stModuleToMain.q_xPrepareDone := TRUE;
91
92      iRobotState := 130;
93  END_I*)
94 130:
95  IF NOT iq_stMainToModule.q_xPrepare THEN
96      iRobotState := 0;
97  END_IF
98
99 //---------------------- AUTO MODE ---------------------------
100
101     //----------------------- INIT ----------------------
102  200:
103     xPowerEnableAll := TRUE;
104     fbRobot.ifMotion.ClearSegmentsFromId(i_udiSegmentId:= 0,
q_etDiag=> , q_etDiagExt=> , q_sMsg=> );
105     IF afbAxis[1].q_xStatus AND afbAxis[2].q_xStatus AND
afbAxis[3].q_xStatus THEN
106         fbRobot.xEnable := TRUE;
107         iRobotState := 210;
108     END_IF
109
110  210:
111     fbRobot.xStart := TRUE;
112     IF fbRobot.xReady THEN
113         iRobotState := 220;
114     END_IF
115
116     //--------------------- SET PARAMETER -> WAIT FOR CHOSEN
SEQUENCE ---------------------
117 220:
118     fbRobot.ifMotion.SetMotionParameter(
119                                             i_etComponent:=
ROB.ET_RobotComponent.All,
120                                             i_lrMaxVelocity:=
500,
```

```
121
i_lrMaxAcceleration:= 2000,
122
i_lrMaxDeceleration:= 2000,
123                                                 i_lrRamp:= 1,
124                                                 q_etDiag=> ,
125                                                 q_etDiagExt=> ,
126                                                 q_sMsg=> );
127     iRobotState := 225;
128
129 225:
130     IF xAutoSeq1 THEN
131         iRobotState := 2310;
132     ELSIF xAutoSeq2 THEN
133         iRobotState := 2500;
134     ELSIF xAutoSeq3 THEN
135         iRobotState := 2700;
136     END_IF
137
138     //----------------- PICK AND PLACE LINEAR -----------------
-------
139 2310:
140     stMoveLVector := stPnPHover1;
141     udiPick := 10;
142
143     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
144                                     i_lrMaxZone:= lrMaxZonePnP,
145                                     i_udiSegmentId:= 1,
146                                     q_etDiag=> etDiag,
147                                     q_etDiagExt=> etDiagExtROB,
148                                     q_sMsg=> );
149
150     IF fbRobot.ifFeedback.udiPreviousSegmentId = 1 OR
etDiagExtROB = ROB.ET_DiagExt.IdenticalTarget THEN
151         iRobotState := 2350;
152     END_IF
153 2320:
154     stMoveLVector := stPnPHover2;
155     udiPick := 20;
156     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
157                                     i_lrMaxZone:= lrMaxZonePnP,
158                                     i_udiSegmentId:= 2,
159                                     q_etDiag=> ,
160                                     q_etDiagExt=> ,
161                                     q_sMsg=> );
162
163     IF fbRobot.ifFeedback.udiPreviousSegmentId = 2 THEN
```

```
164         iRobotState := 2350;
165     END_IF
166 2330:
167     stMoveLVector := stPnPHover3;
168     udiPick := 30;
169     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
170                                     i_lrMaxZone:= lrMaxZonePnP,
171                                     i_udiSegmentId:= 3,
172                                     q_etDiag=> ,
173                                     q_etDiagExt=> ,
174                                     q_sMsg=> );
175
176     IF fbRobot.ifFeedback.udiPreviousSegmentId = 3 THEN
177         iRobotState := 2350;
178     END_IF
179 2340:
180     stMoveLVector := stPnPHover4;
181     udiPick := 40;
182     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
183                                     i_lrMaxZone:= lrMaxZonePnP,
184                                     i_udiSegmentId:= 4,
185                                     q_etDiag=> ,
186                                     q_etDiagExt=> ,
187                                     q_sMsg=> );
188
189     IF fbRobot.ifFeedback.udiPreviousSegmentId = 4 THEN
190         iRobotState := 2350;
191     END_IF
192 2350:
193     stMoveLVector.lrZ := stPnPPickZ.lrZ;
194
195     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
196                                     i_lrMaxZone:= lrMaxZonePnP,
197                                     i_udiSegmentId:= 5,
198                                     q_etDiag=> ,
199                                     q_etDiagExt=> ,
200                                     q_sMsg=> );
201
202         iRobotState := 2355;
203 2355:
204     stMoveLVector.lrZ := stPnPHoverZ.lrZ;
205
206     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
207                                     i_lrMaxZone:= lrMaxZonePnP,
208                                     i_udiSegmentId:= udiPick,
209                                     q_etDiag=> ,
210                                     q_etDiagExt=> ,
```

```
211                                     q_sMsg=> );
212
213     iRobotState := 2360;
214 2360:
215     IF fbRobot.ifFeedback.udiPreviousSegmentId = 10 THEN
216         iRobotState := 2320;
217     ELSIF fbRobot.ifFeedback.udiPreviousSegmentId = 20 THEN
218         iRobotState := 2330;
219     ELSIF fbRobot.ifFeedback.udiPreviousSegmentId = 30 THEN
220         iRobotState := 2340;
221     ELSIF fbRobot.ifFeedback.udiPreviousSegmentId = 40 THEN
222         iRobotState := 2310;
223     END_IF
224
225     //--------------------- XY INFINITY -----------------------
-
226 2500:
227     stMoveLVector := stWaitPosition;
228     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
229                                     i_lrMaxZone:= 0,
230                                     i_udiSegmentId:= 10,
231                                     q_etDiag=> etDiag,
232                                     q_etDiagExt=> etDiagExtROB,
233                                     q_sMsg=> );
234     iRobotState := 2505;
235 2505:
236     IF fbRobot.ifFeedback.udiPreviousSegmentId = 10 OR
etDiagExtROB = ROB.ET_DiagExt.IdenticalTarget THEN
237         iRobotState := 2510;
238     END_IF
239
240 2510:
241     lrCx := 60.0;
242     lrTx := lrCx * 2;
243
244     stCircular := stWaitPosition;
245     stCircular.lrX := stCircular.lrX - lrCx;
246     stCircular.lrY := stCircular.lrY + lrCx;
247
248     stMoveCVector := stWaitPosition;
249     stMoveCVector.lrX := stMoveCVector.lrX - lrTx;
250     stMoveCVector.lrY := stMoveCVector.lrY + 0.0;
251
252     fbRobot.ifMotion.MoveC( i_stTarget:= stMoveCVector,
253                             i_stCircular:= stCircular,
254                             i_lrMaxZone:= lrMaxZone,
255                             i_udiSegmentId:= 1,
```

```
256                             q_etDiag=> etDiag,
257                             q_etDiagExt=> etDiagExtROB,
258                             q_sMsg=> sMsg);
259
260     iRobotState := 2520;
261
262 2520:
263     stCircular := stWaitPosition;
264     stCircular.lrX := stCircular.lrX - lrCx;
265     stCircular.lrY := stCircular.lrY - lrCx;
266
267     stMoveCVector := stWaitPosition;
268
269     fbRobot.ifMotion.MoveC( i_stTarget:= stMoveCVector,
270                             i_stCircular:= stCircular,
271                             i_lrMaxZone:= lrMaxZone,
272                             i_udiSegmentId:= 2,
273                             q_etDiag=> etDiag,
274                             q_etDiagExt=> etDiagExtROB,
275                             q_sMsg=> sMsg);
276
277     iRobotState := 2530;
278
279 2530:
280     stCircular := stWaitPosition;
281     stCircular.lrX := stCircular.lrX + lrCx;
282     stCircular.lrY := stCircular.lrY + lrCx;
283
284     stMoveCVector := stWaitPosition;
285     stMoveCVector.lrX := stWaitPosition.lrX + lrTx;
286     stMoveCVector.lrY := stWaitPosition.lrY + 0.0;
287
288     fbRobot.ifMotion.MoveC( i_stTarget:= stMoveCVector,
289                             i_stCircular:= stCircular,
290                             i_lrMaxZone:= lrMaxZone,
291                             i_udiSegmentId:= 3,
292                             q_etDiag=> etDiag,
293                             q_etDiagExt=> etDiagExtROB,
294                             q_sMsg=> sMsg);
295
296     iRobotState := 2540;
297
298 2540:
299     stCircular := stWaitPosition;
300     stCircular.lrX := stCircular.lrX + lrCx;
301     stCircular.lrY := stCircular.lrY - lrCx;
302
```

```
303     stMoveCVector := stWaitPosition;
304
305     fbRobot.ifMotion.MoveC( i_stTarget:= stMoveCVector,
306                             i_stCircular:= stCircular,
307                             i_lrMaxZone:= lrMaxZone,
308                             i_udiSegmentId:= 4,
309                             q_etDiag=> etDiag,
310                             q_etDiagExt=> etDiagExtROB,
311                             q_sMsg=> sMsg);
312
313     iRobotState := 2545;
314
315 2545:
316     IF fbRobot.ifFeedback.udiPreviousSegmentId = 4 THEN
317         iRobotState := 2510;
318     END_IF
319
320     //----------------- PICK AND PLACE SPLINE -----------------
-------
321 2700:
322     stMoveLVector := stPnPPick1;
323     fbRobot.ifMotion.MoveL(         i_stTarget:= stMoveLVector,
324                                     i_lrMaxZone:= 0,
325                                     i_udiSegmentId:= 10,
326                                     q_etDiag=> etDiag,
327                                     q_etDiagExt=> etDiagExtROB,
328                                     q_sMsg=> );
329     iRobotState := 2705;
330 2705:
331     IF fbRobot.ifFeedback.udiPreviousSegmentId = 10 OR
etDiagExtROB = ROB.ET_DiagExt.IdenticalTarget THEN
332         iRobotState := 2710;
333     END_IF
334 2710:
335     fbRobot.ifMotion.MoveS(         i_stTarget:= stPnPPick2,
336                                     i_stSplineTable:=
stSpline12,
337                                     i_lrMaxZone:= lrMaxZonePnP,
338                                     i_udiSegmentId:= 1,
339                                     q_etDiag=> etDiag,
340                                     q_etDiagExt=> etDiagExtROB,
341                                     q_sMsg=> );
342
343     IF fbRobot.ifFeedback.udiPreviousSegmentId = 1 THEN
344         iRobotState := 2720;
345     END_IF
346 2720:
```

```
347     fbRobot.ifMotion.MoveS(         i_stTarget:= stPnPPick3,
348                                     i_stSplineTable:=
stSpline23,
349                                     i_lrMaxZone:= lrMaxZonePnP,
350                                     i_udiSegmentId:= 2,
351                                     q_etDiag=> etDiag,
352                                     q_etDiagExt=> etDiagExtROB,
353                                     q_sMsg=> );
354
355     IF fbRobot.ifFeedback.udiPreviousSegmentId = 2 THEN
356         iRobotState := 2730;
357     END_IF
358 2730:
359     fbRobot.ifMotion.MoveS(         i_stTarget:= stPnPPick4,
360                                     i_stSplineTable:=
stSpline34,
361                                     i_lrMaxZone:= lrMaxZonePnP,
362                                     i_udiSegmentId:= 3,
363                                     q_etDiag=> etDiag,
364                                     q_etDiagExt=> etDiagExtROB,
365                                     q_sMsg=> );
366
367     IF fbRobot.ifFeedback.udiPreviousSegmentId = 3 THEN
368         iRobotState := 2740;
369     END_IF
370 2740:
371     fbRobot.ifMotion.MoveS(         i_stTarget:= stPnPPick1,
372                                     i_stSplineTable:=
stSpline41,
373                                     i_lrMaxZone:= lrMaxZonePnP,
374                                     i_udiSegmentId:= 4,
375                                     q_etDiag=> etDiag,
376                                     q_etDiagExt=> etDiagExtROB,
377                                     q_sMsg=> );
378     iRobotState := 2745;
379 2745:
380     IF fbRobot.ifFeedback.udiPreviousSegmentId = 4 THEN
381         iRobotState := 2710;
382     END_IF
383
384 //-------------------- MANUAL MODE ----------------------------
385
386     //------------------ INIT -------------------------------
387 300:
388     xPowerEnableAll := TRUE;
389
```

```
390     IF afbAxis[1].q_xStatus AND afbAxis[2].q_xStatus AND
afbAxis[3].q_xStatus THEN
391         fbRobot.xEnable := TRUE;
392         iRobotState := 310;
393     END_IF
394 310:
395     fbRobot.xStart := FALSE;
396
397     IF fbRobot.xReady THEN
398         iRobotState := 320;
399     END_IF
400
401     //---------------------- SET PARAMETERS MANUAL ------------
-------------------
402 320:
403     fbRobot.ifJogging.SetMotionParameter(
404                                             i_etComponent:=
ROB.ET_RobotComponent.All,
405                                             i_lrMaxVelocity:=
50,
406
i_lrMaxAcceleration:= 500,
407
i_lrMaxDeceleration:= 500,
408                                             i_lrRamp:= 1,
409                                             q_etDiag=> ,
410                                             q_etDiagExt=> ,
411                                             q_sMsg=> );
412     iRobotState := 330;
413
414 330:
415     // ------------------------- AXIS X MOVEMENT---------------
-----------------------
416     IF G_stHMI.xManualForwardX THEN
417         G_stHMI.xManualBackX := FALSE;
418         fbRobot.ifJogging.Start(    i_etComponent:=
ROB.ET_RobotComponent.CartesianX,
419                                     i_xForward:= TRUE,
420                                     i_lrMaxDistance:= 0,
421                                     q_etDiag=> etDiag,
422                                     q_etDiagExt=> etDiagExtROB,
423                                     q_sMsg=> );
424
425     ELSIF G_stHMI.xManualBackX THEN
426         G_stHMI.xManualForwardX := FALSE;
427         fbRobot.ifJogging.Start(    i_etComponent:=
ROB.ET_RobotComponent.CartesianX,
```

```
428                                     i_xForward:= FALSE,
429                                     i_lrMaxDistance:= 0,
430                                     q_etDiag=> etDiag,
431                                     q_etDiagExt=> etDiagExtROB,
432                                     q_sMsg=> );
433
434     ELSIF NOT G_stHMI.xManualForwardX AND NOT
G_stHMI.xManualBackX THEN
435         fbRobot.ifJogging.Stop(     i_etComponent:=
ROB.ET_RobotComponent.CartesianX,
436                                     q_etDiag=> etDiag,
437                                     q_etDiagExt=> etDiagExtROB,
438                                     q_sMsg=> );
439     END_IF
440
441     // ------------------------- AXIS Y MOVEMENT---------------
-----------------------
442     IF G_stHMI.xManualForwardY THEN
443         G_stHMI.xManualBackY := FALSE;
444         fbRobot.ifJogging.Start(    i_etComponent:=
ROB.ET_RobotComponent.CartesianY,
445                                     i_xForward:= TRUE,
446                                     i_lrMaxDistance:= 0,
447                                     q_etDiag=> etDiag,
448                                     q_etDiagExt=> etDiagExtROB,
449                                     q_sMsg=> );
450
451     ELSIF G_stHMI.xManualBackY THEN
452         G_stHMI.xManualForwardY := FALSE;
453         fbRobot.ifJogging.Start(    i_etComponent:=
ROB.ET_RobotComponent.CartesianY,
454                                     i_xForward:= FALSE,
455                                     i_lrMaxDistance:= 0,
456                                     q_etDiag=> etDiag,
457                                     q_etDiagExt=> etDiagExtROB,
458                                     q_sMsg=> );
459
460     ELSIF NOT G_stHMI.xManualForwardY AND NOT
G_stHMI.xManualBackY THEN
461         fbRobot.ifJogging.Stop(     i_etComponent:=
ROB.ET_RobotComponent.CartesianY,
462                                     q_etDiag=> etDiag,
463                                     q_etDiagExt=> etDiagExtROB,
464                                     q_sMsg=> );
465     END_IF
466
```

```
467     // ------------------------- AXIS Z MOVEMENT---------------
------------------------
468     IF G_stHMI.xManualUpZ THEN
469         G_stHMI.xManualDownZ := FALSE;
470         fbRobot.ifJogging.Start(    i_etComponent:=
ROB.ET_RobotComponent.CartesianZ,
471                                     i_xForward:= TRUE,
472                                     i_lrMaxDistance:= 0,
473                                     q_etDiag=> etDiag,
474                                     q_etDiagExt=> etDiagExtROB,
475                                     q_sMsg=> );
476
477     ELSIF G_stHMI.xManualDownZ THEN
478         G_stHMI.xManualUpZ := FALSE;
479         fbRobot.ifJogging.Start(    i_etComponent:=
ROB.ET_RobotComponent.CartesianZ,
480                                     i_xForward:= FALSE,
481                                     i_lrMaxDistance:= 0,
482                                     q_etDiag=> etDiag,
483                                     q_etDiagExt=> etDiagExtROB,
484                                     q_sMsg=> );
485
486     ELSIF NOT G_stHMI.xManualUpZ AND NOT G_stHMI.xManualDownZ
THEN
487         fbRobot.ifJogging.Stop(     i_etComponent:=
ROB.ET_RobotComponent.CartesianZ,
488                                     q_etDiag=> etDiag,
489                                     q_etDiagExt=> etDiagExtROB,
490                                     q_sMsg=> );
491     END_IF
492
493 END_CASE
494
495 //-------------------- ALARM/STOP HANDLING --------------------
---
496 IF iq_stMainToModule.q_xFastStop THEN
497     fbRobot.xEnable := FALSE;
498     IF NOT fbRobot.ifFeedback.xInMotion THEN
499         fbRobot.xStart := FALSE;
500         xPowerEnableAll := FALSE;
501
502         iq_stModuleToMain.q_xFastStopDone := TRUE;
503         iRobotState := 0;
504     END_IF
505
506 ELSIF iq_stMainToModule.q_xNormalStop THEN
507
```

```
508     IF NOT fbRobot.ifFeedback.xInMotion THEN
509         iq_stModuleToMain.q_xNormalStopDone := TRUE;
510 
511         iRobotState := 0;
512     END_IF
513 END_IF
514 
515 IF aifAxis[1].Axis.etAxisState = MOIN.ET_AxisState.ErrorStop
THEN
516     axAlarm[1] := TRUE;
517 END_IF
518 
519 IF aifAxis[2].Axis.etAxisState = MOIN.ET_AxisState.ErrorStop
THEN
520     axAlarm[2] := TRUE;
521 END_IF
522 
523 IF aifAxis[3].Axis.etAxisState = MOIN.ET_AxisState.ErrorStop
THEN
524     axAlarm[3] := TRUE;
525 END_IF
526 
527 IF fbRobot.etDiag <> GD.ET_Diag.Ok THEN
528     axAlarm[4] := TRUE;
529 END_IF
530 
531 IF iq_stMainToModule.q_xReset THEN
532     FOR iLoop := 1 TO 8 DO
533         axAlarm[iLoop] := FALSE;
534     END_FOR
535 END_IF
536 
537 //Main to Module 4 variables
538 iq_stModuleToMain.q_xFastStop := axAlarm[1] OR axAlarm[2] OR
axAlarm[3] OR axAlarm[4];
539 iq_stModuleToMain.q_xNormalStop := FALSE;
540 iq_stModuleToMain.q_xReset := FALSE;
541 iq_stModuleToMain.q_xStart := FALSE;
542 
543 //FB_SEAxis refresh variables
544 FOR iAxis := 1 TO 3 DO
545     afbAxis[iAxis](
546     i_ifAxis:= aifAxis[iAxis],
547     i_xPowerEnable:= xPowerEnableAll,
548     i_xReset:= xResetAll,
549     q_xStatus=> ,
550     q_xResetBusy=> ,
```

```
551     q_xResetDone=> );
552 END_FOR
553
554 //FB_SetPos refresh variables while in prepare mode
555 IF iq_stMainToModule.q_xPrepare THEN
556         FOR iAxis := 1 TO 3 DO
557         afbSetPos[iAxis](   ifAxis:= aifAxisDevice[iAxis],
558                             i_xExecute:= ,
559                             i_rPosition:= ,
560                             ifSercosMaster:= SercosMaster,
561                             q_xDone=> ,
562                             q_xBusy=> ,
563                             q_tiTimeToReset=> ,
564                             q_xError=> ,
565                             q_sResultText=> ,
566                             q_xSercosCommOK=> ,
567                             q_lrActPosition=> );
568     END_FOR
569 END_IF
570
571 //Set and store auto sequence
572 IF NOT fbRobot.ifFeedback.xInMotion THEN
573     IF G_stHMI.xAutoSeq1 OR SR_RFID.xPnP THEN
574         xAutoSeq1 := TRUE;
575         xAutoSeq2 := FALSE;
576         xAutoSeq3 := FALSE;
577     ELSIF G_stHMI.xAutoSeq2 OR SR_RFID.xInfinity THEN
578         xAutoSeq1 := FALSE;
579         xAutoSeq2 := TRUE;
580         xAutoSeq3 := FALSE;
581     ELSIF G_stHMI.xAutoSeq3 OR SR_RFID.xPnPSpline THEN
582         xAutoSeq1 := FALSE;
583         xAutoSeq2 := FALSE;
584         xAutoSeq3 := TRUE;
585     END_IF
586 END_IF
587
588 //Check current axis positions
589 lrRefAxisX := aifAxis[1].Axis.lrPosition;
590 lrRefAxisY := aifAxis[2].Axis.lrPosition;
591 lrRefAxisZ := aifAxis[3].Axis.lrPosition;
592
593 fbRobot.ifMotion.lrVelOverride := G_stHMI.lrOverride;
```

# 9.2 HMI

## 9.2.1 ST_HMI

```
1  TYPE ST_HMI :
2  STRUCT
3
4   xManualStart            : BOOL;
5   xManualActive           : BOOL;
6   xManualForwardX         : BOOL;
7   xManualBackX            : BOOL;
8   xManualForwardY         : BOOL;
9   xManualBackY            : BOOL;
10  xManualUpZ              : BOOL;
11  xManualDownZ            : BOOL;
12
13  xAutoStart              : BOOL;
14  xAutoActive             : BOOL;
15  xAutoSeq1               : BOOL;
16  xAutoSeq2               : BOOL;
17  xAutoSeq3               : BOOL;
18  xAutoSeq1Active         : BOOL;
19  xAutoSeq2Active         : BOOL;
20  xAutoSeq3Active         : BOOL;
21
22  xPrepareStart           : BOOL;
23  xPrepareActive          : BOOL;
24  xPrepareDone            : BOOL;
25
26  xReset                  : BOOL;
27  xStart                  : BOOL;
28  xNormalStop             : BOOL;
29  xStopiding              : BOOL;
30
31  xLightsOn               : BOOL;
32
33  rCurrent1               : REAL;
34  rVoltageL1LN            : REAL;
35  rActivePower1           : REAL;
36  rActivePowerTot         : REAL;
37  rPowerFactor            : REAL;
38  rFrequency              : REAL;
39  rEnergy                 : REAL;
40
```

```
41  axAlarmMain             : ARRAY[1..8] OF BOOL;
42  axAlarmRobot            : ARRAY[1..8] OF BOOL;
43  axAlarmSafety           : ARRAY[1..8] OF BOOL;
44  axAlarmHarmony          : ARRAY[1..8] OF BOOL;
45
46  xResetNeeded            : BOOL;
47
48  lrRefAxisX              : LREAL;
49  lrRefAxisY              : LREAL;
50  lrRefAxisZ              : LREAL;
51
52  lrOverride              : LREAL := 100;
53 END_STRUCT
54 END_TYPE
```

### 9.2.2 SR_HMI

```
1  //Main to Module 5 variables
2  iq_stModuleToMain.q_xFastStop := FALSE;
3  iq_stModuleToMain.q_xNormalStop := xStop;
4  iq_stModuleToMain.q_xReset := xReset;
5  iq_stModuleToMain.q_xStart := G_stHMI.xAutoStart;
6  iq_stModuleToMain.q_xFastStopDone :=
iq_stMainToModule.q_xFastStop;
7  iq_stModuleToMain.q_xNormalStopDone :=
iq_stMainToModule.q_xNormalStop;
8
9  IF G_stHMI.lrOverride > 99 THEN
10  G_stHMI.lrOverride := 100;
11 ELSIF G_stHMI.lrOverride < 1 THEN
12  G_stHMI.lrOverride := 0;
13 END_IF
14
15 //Check active mode
16 G_stHMI.xAutoActive := SR_MAIN.xActAuto;
17 G_stHMI.xManualActive := SR_MAIN.xActManual;
18 G_stHMI.xPrepareActive := SR_MAIN.xActPrepare;
19
20 //Only accept mode switch request while in ModeSelect
21 IF NOT iq_stMainToModule.q_xIdle THEN
22  G_stHMI.xAutoStart := FALSE;
23  G_stHMI.xManualStart := FALSE;
24  G_stHMI.xPrepareStart := FALSE;
25 END_IF
26
27 //Stop button requested from HMI
28 IF G_stHMI.xNormalStop THEN
```

```
 xStop := TRUE;
END_IF

//Reset stop button when in idle
IF iq_stMainToModule.q_xIdle THEN
 xStop := FALSE;
END_IF

//Reset stop on reset request
IF iq_stMainToModule.q_xReset THEN
 xStop := FALSE;
END_IF

G_stHMI.lrRefAxisX := SR_RobotNGSFC.lrRefAxisX;
G_stHMI.lrRefAxisY := SR_RobotNGSFC.lrRefAxisY;
G_stHMI.lrRefAxisZ := SR_RobotNGSFC.lrRefAxisZ;

//Connect alarms to HMI
G_stHMI.axAlarmMain := SR_Main.axAlarm;
G_stHMI.axAlarmRobot := SR_RobotNGSFC.axAlarm;
G_stHMI.axAlarmSafety := SR_Preventa.axAlarm;
G_stHMI.axAlarmHarmony := SR_Harmony.axAlarm;

// Check which Auto run sequence is selected
G_stHMI.xAutoSeq1Active := SR_RobotNGSFC.xAutoSeq1;
G_stHMI.xAutoSeq2Active := SR_RobotNGSFC.xAutoSeq2;
G_stHMI.xAutoSeq3Active := SR_RobotNGSFC.xAutoSeq3;

// Check if stop is in action
G_stHMI.xStopiding := SR_Main.xStopiding;

//Check if reset is needed
G_stHMI.xResetNeeded := FALSE;
FOR iAlarmCheck := 1 TO 8 DO
 IF G_stHMI.axAlarmMain[iAlarmCheck] OR
G_stHMI.axAlarmRobot[iAlarmCheck] OR
G_stHMI.axAlarmSafety[iAlarmCheck] OR G_stHMI.axAlarmHarmony[2]
THEN
     G_stHMI.xResetNeeded := TRUE;
 END_IF
END_FOR


//Keep reset high 500MS to ensure complete reset
tofReset(IN:= G_stHMI.xReset, PT:= T#500MS);
xReset := tofReset.Q;

```

```
73
74
75 //Connect power readings to HMI
76 G_stHMI.rCurrent1 := SR_PowerMeter.rCurrent1;
77 G_stHMI.rVoltageL1LN := SR_PowerMeter.rVoltageL1LN;
78 G_stHMI.rActivePower1 := SR_PowerMeter.rActivePower1;
79 G_stHMI.rActivePowerTot  := SR_PowerMeter.rActivePowerTot;
80 G_stHMI.rPowerFactor := SR_PowerMeter.rPowerFactor;
81 G_stHMI.rFrequency := SR_PowerMeter.rFrequency;
82 G_stHMI.rEnergy  := SR_PowerMeter.rEnergy;
```

Printed by Tryckeriet i E-huset, Lund 2021

LUND
UNIVERSITY

Series of Bachelor's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2021-816
http://www.eit.lth.se