Interaktiv systemstyrning av kemisk reaktor från en operatörspanel

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för elektroteknik och automation

Examensarbete:
Philip Saouan
Harmenjeet Sidhu

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

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

Tryckt i Sverige
Industriell elektroteknik och automation
Lunds universitet
Lund 2015

Sammanfattning

Idag finns många företag som använder PLC-styrning för att styra och manövrera olika maskiner och hårdvaror. AB FIA är ett av dessa företag. På deras begäran har en modul framställts med en Programmable Logic Controller (PLC) som styr dess tillkopplade slav-komponenter.

Denna rapport beskriver sambandet av hur styrsystem och komponenter interagerar för att göra kemiska reaktioner mer effektiva och noggranna. Med en kombination av en kraftfull PLC och ett Human Machine Interface (HMI) framställdes en styrmodul som kunde styra många typer av hårdvara och skapa olika typer av sekvenser. Dessa hårdvaror var utformade för att styras både manuellt och automatiskt. Det automatiserade programmet fungerar genom att användaren skickar en textfil till HMI:n som sedan översätter innehållet i filen till en sträng. Arbetet har mestadels bestått av att programmera styrsystemet och designa HMI:n.

Nyckelord: PLC, HMI, styrsystem, kemiska reaktioner, automatisera, interaktiv.

Abstract

Today there are many companies that use PLC-control to control and maneuver their machines and hardware. AB FIA is one of these companies. At their behalf, a module was produced with a Programmable Logic Controller (PLC) that controls its adjacent slave components.

This report describes the relationship how control systems and components interact to make chemical reactions more efficient and precise. With a combination of a powerful PLC and a Human Machine Interface (HMI) a control module was made that could control many types of hardware and create different types of sequences. This hardware was designed to be controlled both manually and automatically. The automated program works so that the user sends a text file through the HMI which then translates the contents of the file into a string. The project has mostly consisted of programming the control system and designing the HMI.

Keywords: PLC, HMI, control systems, chemical reactions, automation, interact.

Förord

Examensarbetet har framställts av Bo Wennergren, vd på AB FIA. Bo hade länge haft behov att automatisera de kemiska processer han utför i sitt laboratorium. Mycket av det manuella arbetet där man på plats måste vrida och knappa på instrumenten ansågs vara både ineffektivt och tidskrävande. Från denna grund lades arbetet upp för oss och från ett mindre projekt i höstas gick vi till ett mer fördjupat examensarbete.

Under projektets gång har vi fått mycket hjälp av Caj Gustafsson som agerade handledare på arbetsplatsen. Caj stod för mycket för den tekniska hjälp vi fick med projektet. Även Mats Lilja och Johan Björnstedt bistod med en hel del tips och hjälp med projektet.

Vi vill därmed passa på att tacka framförallt Bo Wennergren men även Marianne Wennergren och Caj Gustafsson från FIA som har varit väldigt hjälpsamma under hela processen. Även Mats och Johan ska ha ett stort tack. Vi vill även tacka för att vi fått vara delaktiga i FIA:s arbete och fått uppleva en oerhört trivsam miljö och härlig gemenskap som de har på arbetsplatsen.

Innehållsförteckning

Ordlista

PLC: ”Programmable Logic Controller”. Kärnan i systemet och är ett programmerbart styrsystem.

HMI: ”Human Machine Interface”. Touchpanel som är ett användargränssnitt mellan människan och maskinen.

I/O: Input/output. Kontaktdon som kunde ta emot och skicka data.

FTP: ”File Transfer Protocol”. Filöverföringsprotokoll över internet som skickar datafiler.

Oscar Wilde: ”Organic Synthetic Chemist Automated Research Workstation In Laboratory Development Environment”.

1 Inledning

Detta examensarbete ägde rum på ett företag som heter AB FIA och ligger i Södra Sandby strax utanför Lund. Under perioden januari till juni år 2015 har detta examensarbete utförts och ett litet ”förprojekt” la grunden för det under höstterminen 2014. Under förprojektet sattes själva PLC-modulen ihop och kunde då köra en relativt primitiv sekvens där modulen styrde en reaktor och några tillhörande instrument och maskiner. Denna sekvens var utformad för att testa en reaktion med en specifik substans där en statisk sekvens räckte. PLC-modulen visade sig ha mer potential vilket ledde till en vidareutveckling av projektet till ett examensarbete. Där skulle den inte bara köra en simpel sekvens utan den skulle även kunna styra oändliga kombinationer av sekvenser med fler komponenter och instrument i både manuell knappstyrning och automatiserad körning.

AB FIA började med att sälja diverse robottillbehör till AstraZenecas analysrobotar för turbuhaler. Sedan expanderade företaget och man började utveckla utrustning för frisättning av läkemedel från tuggummi. Man utvecklade även apparater med datorstyrd automatisk invägning av kemikalier. Företaget är dock mest känt för sina tuggapparater. Dessa tuggapparater har de både designat och konstruerat själva. År 2001 sålde AB FIA sin första egenkonstruerade tuggapparat till ett företag i Japan. Efter det har dessa apparater fortsatt att exporterats och de säljs fortfarande världen över. År 2012 startade AB FIA ett dotterbolag som hette Prototypverkstaden och ligger placerat i Lund. Prototypverkstaden arbetar hand i hand med Lunds universitet och sysslar till största del med att tillverka prototyper av olika slag. Detta gör så att bland annat AB FIA kan få egentillverkade delar och saker med egna mått för en relativt låg kostnad.

AB FIA driver även egen forskning. En del av den forskning de bedriver innefattar ett antal olika kemiska reaktioner där olika ämnen kan reagera i laboratoriemiljö. Tidigare har dessa reaktorer styrts manuellt av en övervakare och detta kan på lång sikt bli väldigt svårt, tidskrävande och inte ge precisa resultat. AB FIA har därför haft ett behov av ett automatiserat system som kan styra dessa olika processer och alla komponenter som är kopplade till dem. Syftet och ändamålet är då att få processen mindre tidskrävande och mer effektiv. Det är vad detta examensarbete gått ut på: att utveckla ett automatiserat system som kan styra reaktorer och processer genom att skriva

in kommandon i form av textsträngar som sedan behandlas av styrsystemet och utför det jobb operatören vill utfärda.

1.1 Bakgrund

Arbetssättet och bakgrunden bakom projektet byggde till viss del på ett koncept som kallas ”Oscar Wilde”. Det är en förkortning för “Organic Synthetic Chemist Automated Research Workstation In Laboratory Development Environment”. ”Oscar Wilde” utvecklades för att man ville övertyga cirka 70 kemister på AstraZeneca att börja använda organisk syntes på arbetsstationerna. Man ville automatisera detta för att få bättre resultat och mer exakt data på en avsevärt kortare tid. Detta gjorde att man blev mer produktiv och kunde utföra fler tester. Det kunde då implementeras i ett program som styrde hela systemet. Den teoretiska grunden för programmet utnyttjades för att definiera vissa kriterier som speglar ”Oscar Wilde”:s programuppbyggnad. Dessa kriterier kan man se nedan.

- Programmet skall vara enkelt och snyggt utformat.
- Skall vara enkelt att använda.
- Fel som uppstår av användaren skall minimeras.
- Skall öka produktiviteten.
- Skall vara enkelt att producera data.
- Skall inte vara några större buggar i programmet.
- Data skall kunna ses med excel.

För att förbättringen av kemiska processer skulle kunna utvecklas fanns ett stort behov av automatiserade system som med precision utförde processer där manuellt arbete blev för tidskrävande och ineffektivt. Företaget AB FIA arbetade med att förbättra och ta fram praktiska tillbehör och system för både stora och små företag. Det är ett företag med kemiprofil som arbetar hand i hand med utvecklare och forskare med framtagning av prototyper och diverse utrustning. Kunderna använder utrustning av en mängd olika material och fabrikat som antingen tas fram från företagets egen prototypverkstad eller från kunderna själva. I forskningens anda tar AB FIA emot många olika önskemål för forskningsprojekt inom många olika områden.

1.1.1 Syfte och målsättning

Syftet med projektet har varit att hjälpa Bo Wennergren, VD på AB FIA, med sina experiment och sin forskning. Att han ska kunna hålla bättre koll och ta mer precisa mätvärden på sina mätningar. Den senaste tiden har Bo forskat inom området biogastillverkning. Han har forskat kring ett specifikt ämne av plast som vid vissa temperaturer kan absorbera koldioxid ur gaser. Denna forskning ansågs kunna utvecklas i framtiden och behovet av att framställa ett mer effektivt styrsystem under sina experiment fanns. Men den framställda PLC-modulen som konstruerades under detta examensarbete skulle inte bara kunna styra detta lilla experiment. Syftet var att den skulle kunna styra ofattbart många experiment där olika instrument användes. Genom att ha en relativt liten och lättanvänd styrmodul kunde Bo enkelt byta miljö och visa sina resultat enklare på nya platser.

Målet med detta examensarbete är att försöka bygga och utveckla ett system som interaktivt kan styra hårdvaror och kringutrustning till en kemisk reaktor. Systemet ska vara utformat så att processen blir smidigare att utföra och mer effektiv. Systemet skall kunna styra all kringutrustning på egen hand utan hjälp av användare eller övervakare. Programmet skulle därmed fungera helt automatiskt efter att användaren hade skickat in en lista med kommandon som skulle köras. Systemet skulle även kunna köras i ett manuellt läge där användaren kunde styra all kringutrustning från operatörspanelen. Systemet skulle vara användarvänligt. Systemet skulle vara utrustat med nödstopp.

1.1.2 Problemformulering

När projektet startade och grunderna lades fram började problemställningar och frågor på hur projektet skulle se ut formas. Tanken var att styrsystemet skulle likna ett slags ”universalstyrsystem” där sekvenser skulle kunna definieras. Man skulle enkelt och smidigt kunna styra sina instrument och maskiner på ett användarvänligt sätt. En primär frågeställning uppstod inför projektet och största delen av programmeringen och framställningen av PLC-modulen kretsade kring en fråga:

Går det att interaktivt styra kringutrustning till en kemisk reaktor från en operatörspanel med ett användarvänligt grafiskt gränssnitt på ett enkelt och effektivt vis?

För att besvara denna fråga designades ett program utefter användarens, d.v.s. Bo Wennergren, tankar och krav. Genom att testa programmet grundligt själva och sen av användaren med olika hårdvaror och scenarion skulle användarvänlighet och enkelhet avgöras. För att testa effektiviteten av programmet skulle en sekvens köras utan PLC:n och sedan med PLC:n.

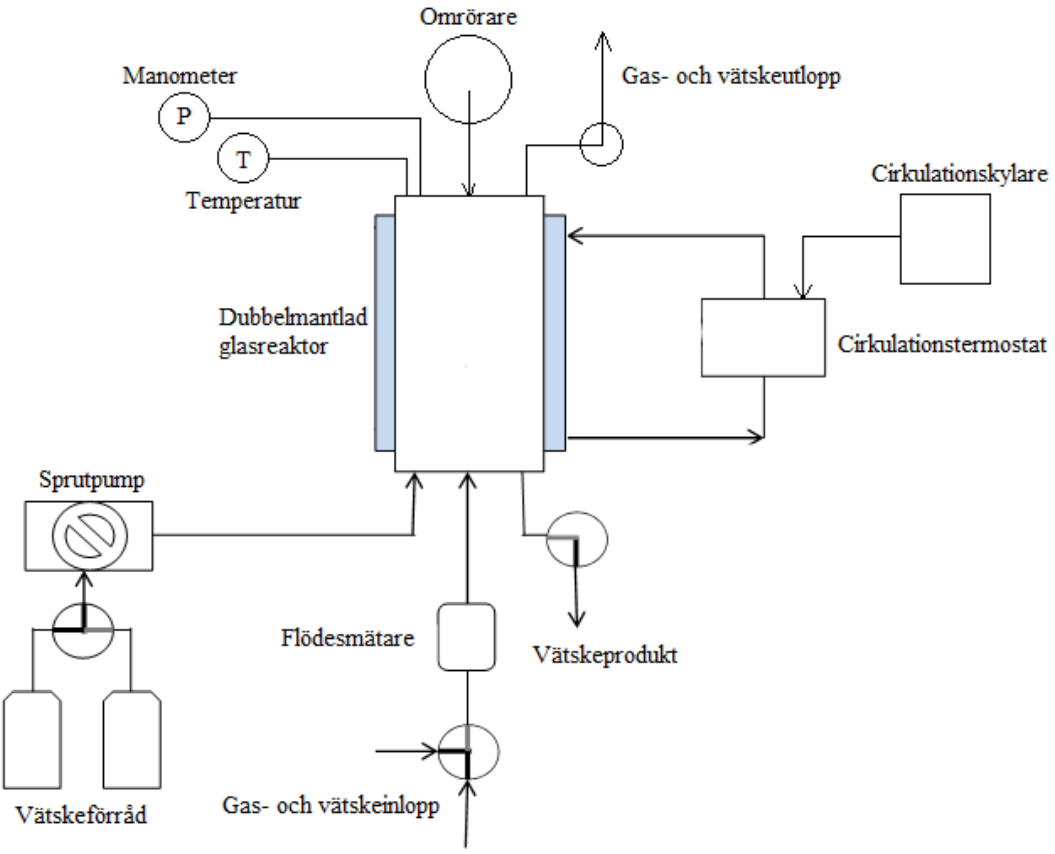
1.1.3 Avgränsningar

Vid ett tids- och kostnadsbegränsat projekt som detta där man byggt ett styrsystem som skulle kunna styra mängder av olika instrument och maskiner måste man sätta vissa begränsningar. Den största begräsningen med styrsystemet var att det endast fanns två portar för RS-232-kommunikation. Man kunde alltså bara jobba med omrörare, pump, termostat eller logger med två val körande samtidigt. Man kunde givetvis skifta mellan dem men man kunde aldrig köra dem allihop samtidigt. PLC-modulen hade kapacitet att ha fler portar men valet gjordes när lådan byggdes ihop och togs för att lådan inte skulle bli för stor och tung. En lätt och smidig portabel PLC-modul var tanken.

2 Teknisk bakgrund

2.1 Grafiskt systemupplägg

Tanken och upplägget kring systemet som PLC-modulen skulle styra kretsade kring upplägget på bilden nedanför (figur 2.1). Den dubbelmantlade glasreaktorn var i vårt fall kärnan i systemet. Där kunde gaser och vätskor blandas och reagera med varandra. Omröraren var delaktig i denna reaktion, ofta för att hålla ämnena i rörelse så man slapp eventuella klumpningar. Sprutpumpen skickade kontinuerligt in vätska och cirkulationstermostaten tillsammans med cirkulationskylaren hjälpte till att hålla rätt temperatur under reaktionens gång. Givare som tryck- flöde- och temperaturgivare gav loggdata.



Figur 2.1: Grafisk uppsättning av reaktorsystemet.

Under projektets gång användes olika hårdvaror och mjukvaror. Mjukvarorna och programmen var QViS, Codesys och RScom. Hårdvarorna var PLC med interface (HMI), omrörare, pump, cirkulationstermostat, cirkulationskylare, logger, tryckgivare, flödesgivare, temperaturgivare och ventiler. Dessa förklaras mer djupgående i kommande stycken.

2.2 Mjukvaror

Ett HMI (Human Machine Interface) programmerades i ett program som hette QViS. Detta var ett användarvänligt gränssnitt där man enkelt gjorde upp en grafisk bild över hur touchpanelen på modulen skulle se ut och användas. Kommunikationen med styrsystemet skickades via olika kommandon, textsträngar och variabler som lokaliserade vilka knappar som ska göra vad. QViS har utvecklats av ett företag som heter Hilscher. Programmet var utformat för att vara användarvänligt och enkelt för en användare att navigera i. Man kunde enkelt ladda upp sitt program till HMI:n genom att göra en sökning av hårdvaror i nätverket. Variabler kunde laddas upp och användas i programmet genom att hitta de sparade filerna med alla variabler som kodats i det programmeringsverktyg man har använt sig av. Precis som de flesta programmeringsverktyg kunde man kompilera det program som man utformat och kontrollera så att programmet inte hade några buggar. Genom att ladda upp nya bibliotek i programmet kunde man göra sitt program precis som man själv behagade.

För att skicka och ta emot kommandon direkt till/från respektive hårdvara användes ett program som heter RScom. För att använda detta program behövdes en RScom-kabel som kopplades direkt från den dator man använde till den hårdvara man ville skicka kommandon till. Denna kabel var tvungen att installeras i datorn så att den gick att använda. Genom att ställa in den port kabeln använde kunde man enkelt skicka och ta emot kommandon från den hårdvara man var uppkopplad till. Beroende på den hårdvara man var uppkopplad till kunde man behöva ställa in start- och slutbitar som läggs till automatiskt i den kommandosträng man som användare ville skicka. Detta program underlättade kommunikationen till hårdvaran då man testade hur hårdvaran fungerade och hur en kommandosträng som skickades skulle se ut.

PLC-modulens komponenter kom från företaget Turck och kodningen till programmet skrevs i Codesys version 3 med standarden IEC 61131. Detta

program samt bibliotek hämtades från deras hemsida. Strukturen valdes att programmera i strukturerad text (ST) då detta upplevdes som mer användarvänligt. Programmet och kodningen i Codesys kopplades i sin tur ihop med QViS och HMI:t.

2.3 Hårdvaror

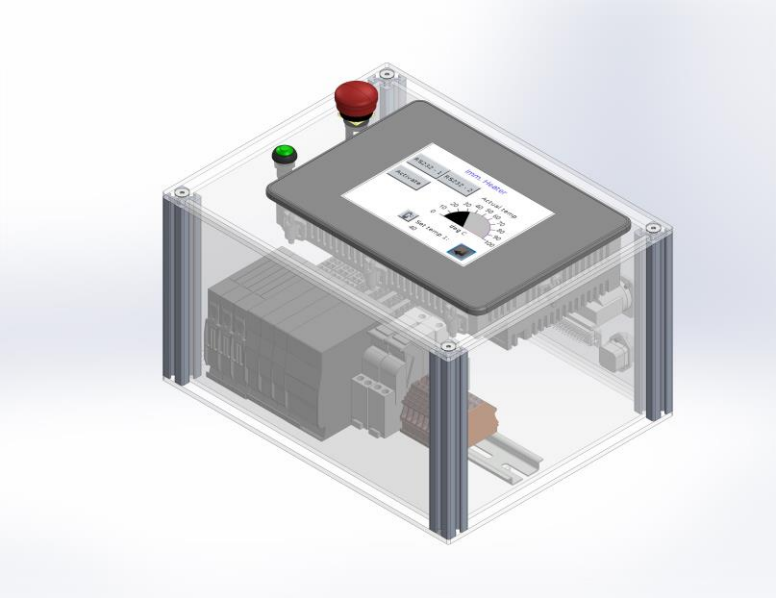
På ett tidigt stadium påbörjades projektet med att modulen sattes ihop. Detta styrsystem bestod av sju olika delar. Delarna som byggdes ihop var BL20-I/O-moduler från företaget Turck. En del var för busskommunikation av typen Profibus, två delar för RS-232-kommunikation med termostat, omrörade, logger och pump. Resterande delar bestod av digitala och analoga in- och utgångar för att ge och ta emot data. Termostaten var av typen Huber MPC-E och var en cirkulationstermostat som användes för att kontrollera temperaturen på ett stabilt sätt i vattenbad eller bad med andra vätskor. Termostaten innehöll en MPC microprocessor och en LED-skärm med knappar som man manövrerade inställningarna med. Här fanns även en cirkulationskylare (kylfinger) av typen Julabo FT200 som hjälpte till att kyla vätskan. Detta på grund av att cirkulationstermostat inte kylde vätskan tillräckligt. Omröraren var av typen CAT R20D_PC och var en elegant omrörare som kan nå varvtal mellan 50-2000 rpm med en kapacitet på 35 liter (vatten). En logger fanns även med som ett extraval vid lämpligt användningstillfälle. Loggern var en Almemo 8590. Pumpen som användes var en NE-1000 Syringe Pump med ett avancerat interface. Med nio ställbara knappar kunde man automatiserat eller manuellt ställa in pumpens olika funktioner och inställningar. Tryckgivaren som användes var en PT010R med ett tryckområde på 0-10 bar. Temperaturgivaren var en 4-tråds PT-100 från Turck.

HMI och PLC har konstruerats av företaget Turck och var av modellen VT-250. Som tidigare nämnt kopplades denna HMI till programmet QViS för att designa och utveckla den grafiska bilden på HMI:t, men även för att ställa in de knappar som används i programmet.

3 Metod

3.1 PLC-modulen

Modulen kopplades samman med HMI:n och senare även med reaktorn och de olika komponenter och instrument som önskades. En användarvänlig låda konstruerades där modulen och respektive sladdar monterades. Man slapp således de flesta sladdar som ofta tar plats och var i vägen och kunde enklare flytta på konstruktionen dit man behagade. Lådan konstruerades hos AB FIA:s ”dotterbolag” Prototypverkstaden på ett sätt så man enkelt kunde byta ut och lägga till olika komponenter inuti modulen om man så ville.



Figur 3.1.2. PLC-modul, BL20-moduler och touchpanel (HMI).

3.2 Reaktorn

Reaktorn som användes i detta fall bestod av en dubbelmantlad glasreaktor som innehöll olika substanser och vätskor. I det yttre höljet av reaktorn reglerades temperaturen i form av en vätska som cirkulationstermostaten värmde och cirkulationskylaren kylde. In och ut från reaktorerna pumpades gaser och vätskor via ventiler med pumpen som satt intill modulen. Inuti reaktorn blandades vätskan med hjälp av omröraren. Mätvärden från olika givare, som temperaturgivare, lagrades i olika variabler.

3.3 Koden

Koden byggdes upp på ett sätt som enkelt kunde hålla reda på vilka instrument som kördes. Blocksystem med case-satser var essentiellt i upplägget. Ett huvudprogram kördes där varje plats i en vektor innehöll ett instrument/maskin eller en specifik sorts körning. En del bestod av textsträngar som skickades in via FTP (file transfer protocol) och översattes i modulen innan behandling. Variablerna som var globala överfördes till HMI:n där de allokerades till knappar eller symboler. I QViS hanterade dessa variabler och layouten för programstrukturen definierades.

Koden bestod av flera olika block och centralt bland dem var vårt huvudprogram, ’’P0_MainProgram’’. Genom denna kunde man starta och stänga sina block precis som man behagade och den underlättade testningen av de olika programblocken och hur de fungerade tillsammans. Detta gjordes genom att alla knappar och kommandon för att starta olika block kopplades till huvudprogrammet. Man implementerade även en loop som översatte den inkommande RS-232 vektor-listan från bytes till strängar så att man enkelt kunde se vad denna vektor innehöll. Detta var ett block som var väldigt viktigt under examensarbetets gång. På bilden nedan (figur 3.3.1) kan man se hur detta block såg ut.

P0_MainProgram

```
(*                  Huvudprogram
                    ============
                    Programkontroll
                    ---------------                    *)

IF P[1]  THEN P1_Init();                ELSE    uiSekvensP[1]:= 0; END_IF;
IF P[2]  THEN P2_RunSequence();         ELSE    uiSekvensP[2]:= 0; END_IF;
IF P[3]  THEN P3_ConfigureSequence();   ELSE    uiSekvensP[3]:= 0; END_IF;
IF P[4]  THEN P4_ConfigureHardware();   ELSE    uiSekvensP[4]:= 0; END_IF;
IF P[5]  THEN RS_232();                 ELSE    uiSekvensP[5]:= 0; END_IF;
IF P[6]  THEN NE_1000();                ELSE    uiSekvensP[6]:= 0; END_IF;
IF P[7]  THEN ALMEMO();                 ELSE    uiSekvensP[7]:= 0; END_IF;
IF P[8]  THEN R20D_PC();                ELSE    uiSekvensP[8]:= 0; END_IF;
IF P[9]  THEN Logger();                 ELSE    uiSekvensP[9]:= 0; END_IF;
//IF P[10] THEN Logger_com();           ELSE    uiSekvensP[10]:= 0; END_IF;
IF P[11] THEN Valves();                 ELSE    uiSekvensP[11]:= 0; END_IF;
IF P[12] THEN ImmersionHeater();        ELSE    uiSekvensP[12]:= 0; END_IF;
IF P[13] THEN Com_List();               ELSE    uiSekvensP[13]:= 0; END_IF;
IF P[14] THEN P14_FileRead();           ELSE    uiSekvensP[14]:= 0; END_IF;
IF P[15] THEN P15_Flow();               ELSE    uiSekvensP[15]:= 0; END_IF;

k:='';
k2:='';

FOR i:= 0 TO 100 DO
        k:= CHR_TO_STRING(RS232IN[i]);
        k2 := CONCAT(k2,k);
END_FOR
```

Figur 3.3.1. Huvudprogrammet.

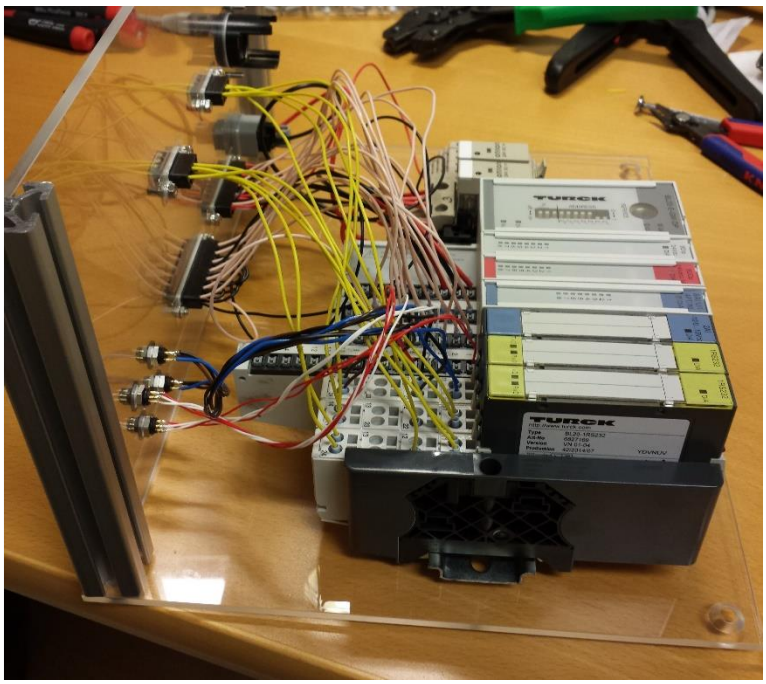
4 Utförande

4.1 Design och konstruktion

Projektet inleddes med flera möten där den byggda modulens funktion diskuterades. Under dessa möten diskuterades även de komponenter som behövdes för just detta projekt. Därefter skapades en lista med komponenter som behövdes tillsammans med varje komponents pris. Dessa komponenter beställdes från företaget Turck då man kom fram till att deras pris var rimligt och för att de kunde erbjuda alla de produkter som behövdes. Då man även diskuterade utbyggnad och utveckling av modulen var även här Turck ett bra val.

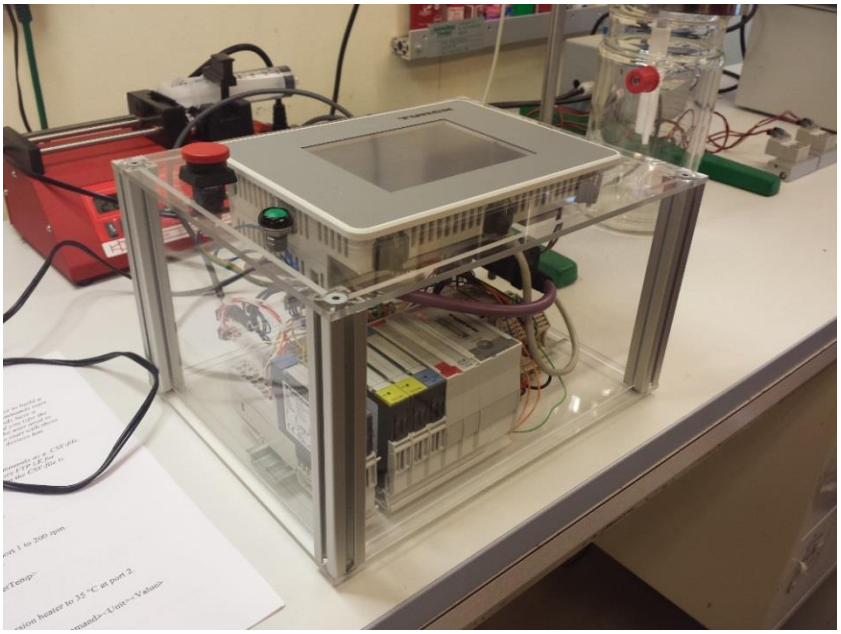
I väntan på ankomsten av beställda komponenter diskuterades designen av det hölje där alla dessa komponenter skulle monteras. När designen var färdig skickades beställningen till Prototypverkstaden där den tillverkades. Höljets material valdes att vara transparent. När de olika komponenterna från Turck anlände monterades dessa komponenter på specifika platser för att det inte skulle uppstå problem senare under projektets gång.

För att binda de olika komponenterna på kopplingsplinten till varandra drogs kablar mellan dem (se figur 4.1). Detta tog väldigt lång tid då man var tvungen att studera olika datablad för att undvika problem och kortslutning när modulen senare skulle köras igång. Sedan monterades även två reläer i höljet. Dessa kopplades till en nödstoppsknapp och en knapp som styrde strömmatning, det vill säga en på och av knapp.



Figur 4.1: BL20 I/O-modul vid konstruktion.

När insidan av höljet var klart började man jobba på utsidan genom att först montera HMI:n och PLC:n överst på höljet. Denna kopplades ihop med de andra komponenterna med hjälp av en Profibus. Komponenterna som styrde in- och utgångar kopplades till D-subbar som även dem monterades på utsidan av höljet. Detsamma gällde för RS-232 komponenterna som även dem kopplades till D-subbar på höljets utsida. Man monterade även 4 stycken kontaktdon på utsidan som sedan kopplades till de komponenter som styrde PT-100 givaren och tryckgivaren. Sedan var höljet färdigmonterat och arbetet på att programmera modulen påbörjades (se figur 4.2).



Figur 4.2: PLC-modulen färdig och komplett.

4.2 Programmets funktion

För att programmera själva PLC:n användes ett program som hette Codesys. I denna programvara kunde man programmera sina enheter i olika former. För detta projekt valde man att programmera i strukturerad text (ST) då detta upplevdes som lite enklare och mer effektivt. I vår version av Codesys användes en standard som hette IEC 61131-3. När programvaran hade studerats noga påbörjade man programmeringen enheten. Man började med att skapa en enda sekvens som skulle kunna köras på egen hand efter att nödvändiga värden valts in i programmet. I takt med att kunskaperna om programvaran ökade påbörjades arbetet på den manuella styrningen av olika hårdvaror. Man var tvungen till att programmera hårdvarorna var för sig då de använder sig av olika kommandon.

Koden för varje hårdvara skrevs in i separata block så att man enkelt kunde hålla reda på vilken del av koden som styrde vilken hårdvara. Det uppstod vissa komplikationer under denna process då vissa hårdvaror hade begränsningar i hur fort de behandlade kommandon. Men efter lite tid var den manuella styrningen färdig och programmet testades och vissa funktioner implementerades i programmet efter förfrågan av företagets vd, Bo. När den manuella styrningen var färdig började arbetet med den automatiska styrningen. Då man önskade att den automatiska styrningen skulle fungera genom att man skickade in ett Excel-blad (se ett exempel i bilagor, slutet av ”Manual for auto-control”) till modulen funderade man ut en enkel och

effektiv lösning för detta. Valet föll på att använda ett gemensamt nätverk för att skicka Excel-blad från en dator till modulen. Detta gjordes genom att man använde ett FTP-program där man kopplade upp sig till modulen genom nätverket och angav användarnamn och lösenord för att logga in på modulen.

Då koden för den manuella styrningen redan var färdigskriven kunde man återanvända samma kod vid automatisk styrning. Man fick dock skriva ett helt nytt block för att behandla den inkommande strängen som hämtades från Excel-bladet. Eftersom detta block blev centralt och mycket fokus lades här har detta block lagts som bilagor (se Com_List). Detta block skickade olika kommandon till de block som styrde respektive hårdvara. Blocket Com_List bestod huvudsakligen av en inkommande vektor (array) av olika strängar som bearbetades sträng för sträng. Dessa strängar bestod i sin tur av olika kommandon och variabler som skulle ställas in i varje hårdvara.

Som man kan se i bilagan bestod blocket av en case-sekvens där man i sekvens noll började man med att identifiera vilken hårdvara efterföljande kommandon hade. Då programmet hade en funktion där en användare kunde välja att repetera sina kommandon gjordes denna kontroll i sekvens noll. När detta var klart gick programmet vidare till den sekvens som hårdvaran behandlade skulle bearbetas. Efter att de olika värden hade sparats i de interna variablerna startades rätt block för att skicka de begärda kommandona till rätt hårdvara. Sedan väntade man in blocket Com_List på att det block som startades blev klart. Denna funktion lades till för att modulen man använde sig av hade två olika RS-232-portar. Då två olika hårdvaror om använde sig av RS-232-portar kunde användas samtidigt uppstod det komplikationer när man körde programmet utan denna funktion. Denna komplikation var att fel kommandon skickades till fel hårdvara då programmet inte hann ändra vilken RS-232-port som de olika hårdvarorna använde sig av. Därför beslutades att man endast körde en hårdvara som använde sig av en RS-232-port åt gången. Sedan stegade programmet igenom den inkommande vektorlistan tills ett slutkommando påträffades. När detta skedde stängde programmet ner blocket InComList då detta slutkommando indikerar att vektorlistan med alla kommandon hade körts igenom. Parallellt med detta började arbetet med den grafiska uppsättningen av HMI:n.

4.3 Operatörspanel

Designen av den grafiska uppsättningen för HMI:n gjordes i en programvara som hette QViS. Detta program valdes för att det var enkelt att använda och hade alla de funktioner man efterfrågade. Under projektets gång skapades en enkel version av den grafiska uppsättningen för att man skulle kunna använda sig av olika variabler som behövdes för att testa programmet. Det utformades med en startmeny där användaren välkomnades och ställdes inför ett antal knappval. Man valde mellan ”automatiserad drift”, ”manuell drift” eller ”sparade sekvenser”. Automatiserad drift-knappen tog användaren till en ny meny där man kunde se alla värden och alla data i realtid. Det vill säga man kunde se både satta värden och aktuella värden på sina parametrar. Den manuella driften tog användaren till en meny där alla instrument och tillkopplade maskiner fanns som knappar. Man kunde då trycka sig vidare in på varje instrument och detaljerat ändra och sätta värden som man själv önskade. Under knappen sparade sekvenser fanns det, i detta fall, endast en sekvens inlagd. Denna sekvens var den tidigare nämnda som skrevs i början av projektet. Givetvis fanns det möjligheter att i framtiden lägga in fler sparade sekvenser där. Efter hand som koden blev bättre och mer avancerad så utvecklades och förfinades även den grafiska designen av HMI:n. Denna gjordes mer användarvänlig och prydlig så att användaren enkelt kunde använda programmet. Variabler skickades från Codesys till HMI:n där de implementerades och allokerades. När designen var färdig laddades den upp till HMI:n och testades.

När man var klar med både koden till PLC:n och designen av HMI:n påbörjades flera olika tester av programmet för att se som programmet innehöll eventuella buggar. Vissa buggar upptäcktes och noterades för att sedan fixas. Programmet testades flera gånger om och om igen. När man var färdigt med detta fick huvudanvändaren, det vill säga företagets vd Bo, testa programmet. När det inte uppstod fler buggar i programmet ansågs man vara färdig med projektet och en utförlig manual skrevs för användning till det automatiserade läget av programmet (se bilagor ”Manual for auto-control”).

4.4 Testning

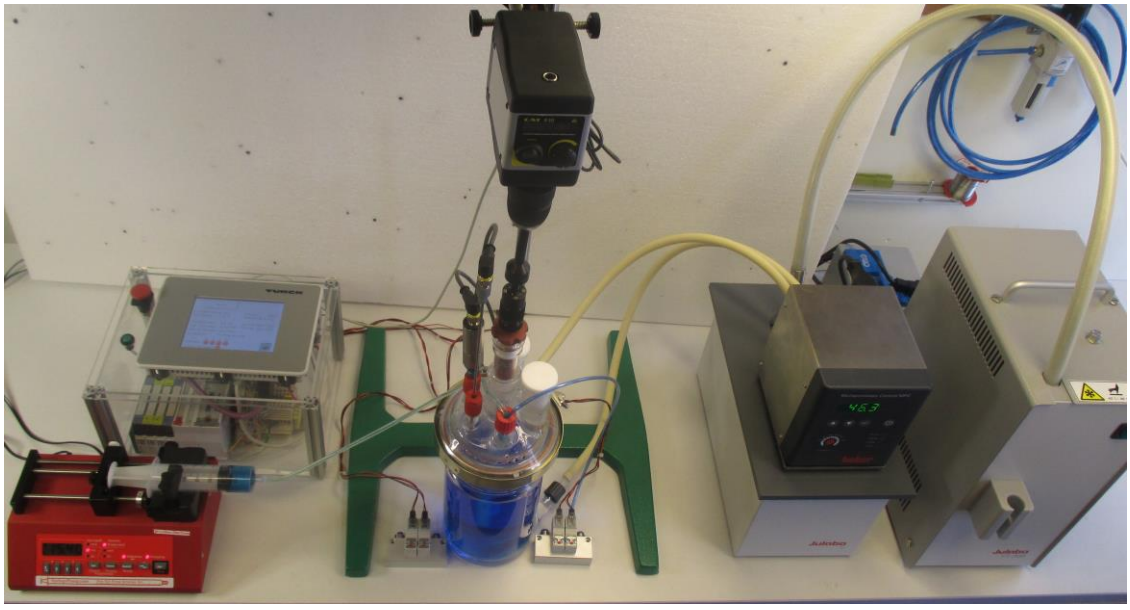
Alla komponenter och instrument testades kontinuerligt under tiden koden skrev och testades. För varje instrument som kopplades samman med modulen skrevs först kod för manuell körning. Detta gjorde det enklare att förstå hur maskinen och instrumenten fungerade tillsammans. När koden för manuell körning var gjort så testades varje instrument en efter en, grundligt och genomdrivande. Då alla hårdvaror testats och de kunde styras via modulen började arbetet med att få programmen att arbeta automatiserat. En test-sträng skrevs som skulle prova hur modulen arbetade i automatiserad drift. Resultatet från testningarna presenteras under resultat.

5 Resultat

Resultatet av examensarbetet blev en modul som enkelt kunde manövreras från en touchpanel. Genom att programmera PLC:n på ett sätt där en användare interaktivt kunde styra hårdvaror och konstruera en eller flera sekvenser som kunde köras automatiskt eller manuellt. Användaren kunde skapa ett Excel-dokument där varje funktion ens program skulle innehålla skrevs samman till ett program med hjälp av en manual för automatiserad styrning (se bilagor ”Manual for auto-control”). Denna egenskrivna sekvens behandlades sedan av PLC:n och översattes till körbar kod.

Syntaxen för programmet var uppbyggd så användaren först skrev in vilken del av sin utrustning som skulle köras. Ett kort kommando som t.ex. ”ST” följt av vilken port man ville använda och sist vilken hastighet man ville att, i detta fall, omröraren skulle arbeta i. Olika instrument hade olika uppbyggnader på hur syntaxen skulle se ut och vissa instrument som t.ex. pumpen behövde först ställas in med vissa inställningar som diameter, hastighet och volym. Ventilerna valde användaren antingen att köra med en specifik tid (timer) eller bara låta dem vara öppna tills kommandot ”stäng” uppträdde. Användaren kunde även välja att bara lägga in en timer som skulle hålla ens sekvens stilla på ett specifikt ställe i strängen. Att upprepa vissa repetitioner kunde också köra ens sekvenser mer användarvänliga då man slipper återupprepa sig i kommandosträngen varje gång man vill köra repetitioner. Man valde då först hur många gånger man ville köra de nästkommande sekvenserna följt av vilka som skulle köras.

Parallellt med koden som kördes arbetade en egenskriven logger som loggade värden med valt intervall. Loggern skrev över data i ett Excel-dokument med givna tider och klockslag som markerade varje händelse.
På bildern nedan (figur 5.1) kan man se reaktorn och de instrument som användes vid just denna reaktion. Upp till vänster ser man modulen som bestod av HMI och PLC Modulen kopplades upp med diverse hårdvaror – Syringepumpen syns ner till vänster, cirkulationstermostat och cirkulationskylaren tillhöger och omrörare högst upp i mitten. Centralt ser man reaktorn med en blå vätska och intilliggande fyra ventiler.



Figur 5.1. Reaktorn och alla komponenter.

Som tidigare nämnt implementerades ett manuellt läge i modulen. I detta läge kunde alla hårdvaror och utgångar styras och ställas in manuellt var för sig. På detta vis kunde användaren manuellt ändra saker och testa sina instrument utan att behöva skriva ihop en automatiserad lista. Man kunde även se aktuella värden och parametrar i realtid. Man kunde även använda detta läge för att säkerställa att all hårdvara fungerade som det skulle. När modulen kördes i manuellt läge kunde man se alla mätvärden direkt på HMI:n och även vilka hårdvaror som för tillfället användes.

6 Slutsats och diskussion

Resultatet av examensarbetet har varit mycket varierande, intressant och på många sätt givande. Somliga saker flöt på bra och lösningar fanns nära till hands och vissa bitar tog lite längre tid. De automatiserade sekvenserna som skrevs ihop av textsträngar gjorde att programmet blev oerhört mycket bättre och gav användaren en större variation. Att styra instrumenten som användes fungerade bra och koden gjorde det den var utsatt att göra. Första problemen kom när data skulle loggas. Det fanns en logger till hands som kunde logga värden mycket snabbt och effektivt. Dock behövde den en av de två RS-232-portar som PLC-styrmodulen hade. Detta skapade lite bekymmer eftersom man då bara har en port kvar till de större instrumenten. Därför valdes att inte fördjupas i denna logger utan istället försöka logga data på egen hand. Detta beslut togs på grund av att loggern skapade onödigt mycket data. Det räckte därför att logga data på egen hand med en bit kod istället. En annan felkälla var att cirkulationstermostaten inte kunde starta blandaren/uppvärmningen via kommandon. Det fanns en sträng som skulle skickas och utföra detta men programmet lyckades aldrig utföra detta. Därför lades det som ett känt fel och uppmanade istället användaren att först starta termostaten manuellt innan programmet kördes.

Kunskaperna under examensarbetet har ökat markant inom den tekniska biten, programmeringen och felsökningen. Även planering, projekthantering och grupparbetet har satts på prov och utvecklats. Förprojektet som gjordes i höstas gav ett litet försprång i hur PLC-modulen fungerade och hur den var uppbyggd. Detta gjorde så att energin kunde riktas i att fördjupas mer i programmet och hur systemet skulle fungera på ett så effektivt sätt som möjligt. Genom att skriva kod som kunde hantera textsträngar som användaren själv byggde ihop kunde programmet agera mer dynamiskt än statiskt, som det tidigare varit. Allt detta för att på bästa sätt kunna styra kemiska reaktorer och dess komponenter på ett så användarvänligt sätt som möjligt.

”Oscar Wilde”, som tidigare har nämnts, gick ut på att skapa ett automatiserat system för att styra en reaktor utan hjälp av en person. Med detta som grund utvecklades denna modul för att fungera helt automatiskt i det automatiserade läget. Samtidigt kunde man lagra mätvärden och se uppdaterad data under tiden som sekvensen kördes. Målet med examensarbetet var att effektivisera olika sorters kemiska reaktioner och göra laborationer mindre tidskrävande,

mer effektiva samt införa automatisk loggning av data. Om man återspeglar till ”Oscar Wilde”-upplägget så kan resultatet sägas uppfylla de flesta krav som var formulerade för examensarbetsprojektet. Då det fanns krav på att programmet skulle fungera enkelt och effektivt, la man mycket tid på detta och med tanke på några mindre komplikationer lyckades man relativt bra med detta. Detta avgjorde Bo Wennergren efter att ha testat programmet fullständigt på det vis som önskades. HMI:n kunde göras snyggare men valet att ha ett så enkelt och lättmanövrerat HMI gjorde att fokus lades på detta istället för hur snyggt upplägget var. HMI:n har ett simpelt upplägg med trevlig meny-miljö. Det är också enkelt att använda med ett litet undantag vid den manuella styrningen. Där var det några variabler som var svåra att implementera i enkel styrning. Detta kunde upplevas som svårt för en ovan användare. Minimera felen från användaren har implementerats så gott som möjligt. Det finns alltid en chans att man missat något när man inte testat programmet tillräckligt många gånger. Programmet hade behövt användas på många olika sätt för att i efterhand upptäcka eventuella buggar. Dock var detta ett steg fram i framtiden. Att öka produktiviteten var en essentiell del i arbetet. Produktiviteten ökar jämsides med att effektivisering införs. Att inte lägga onödig tid på att hålla koll på rätt värden och istället kunna använda sin dyrbara tid till andra saker låg i stort intresse. Det är där målen möts med Oscar Wilde-teorin. Med textsträngar som sänds in via FTP och tillsammans med manualen (”Manual for auto-control”, se bilagor) som framställdes för användaren kunde man bygga sin egen sekvens precis som man ville ha den. Därmed effektiviserades processen avsevärt.

6.1 Felkällor

En nackdel med programmet var när man körde automatiserat läge och skulle använda kommandot ”RP” (Repetitions, se bilagor), så implementerades aldrig kommandot att köra repetitioner inuti andra repetitioner. Alltså man kunde bara köra ett ”RP”-kommando i taget, när det första var färdigt så kunde man skriva ett nytt ”RP”-kommando.

7 Framtida utvecklingsmöjligheter

Största framtida möjligheten kommer definitivt vara en utveckling av lådan som PLC-modulen lades i. Att lägga till fler I/O-komponenter som kan få modulen att styra fler instrument parallellt. Även att styra tyngre och maskiner som kräver mer kraft kan vara en framtidsvision. Detta kan man som användare utveckla hur mycket som helst efter sin egen syn. Möjligheter finns att implementera koden i en starkare PLC som kan styra stora maskiner och realisera de ”små” experimenten som görs på AB FIA till fullskaliga experiment i större miljöer.

8 Referenser

[1] Datablad VT-250
http://www.download-turck.se/root_files/VT250_manual.pdf
Januari 2015

[2] Hemsida till BL20
http://www.turck.us/Products/Networks/BL20/
Januari 2015

[3] Cirkulationstermostat, Huber MPC-E
http://www.huber-online.com/en/product_datasheet.aspx?no=2035.0005.99
Mars 2015

[4] Omrörare, CAT R20D_PC
http://www.labteamet.com/produkter?catP=70/
Mars 2015

[5] Logger, Almemo 8590
http://www.inds.co.uk/almemo/almemo_8590.htm
Mars 2015

[6] NE-1000 Syringe Pump
http://www.alascience.com/products/pdf/NE-1000.pdf
http://www.syringepump.com/
Senast besökt: maj 2015

[7] Kylare, Julabo FT200
http://www.julabo.com/en/products/additional-products/immersion-coolers/ft200-immersion-cooler
April 2015

[8] Tryckgivare, PT010R
http://pdb2.turck.de/en/DE/products/0000000100011674000 4003a
Februari 2015

[9] Temperaturgivare, PT100
http://pdb.turck.de/media/_en/Anlagen/Datei_EDB/edb_9910474_gbr_en.pdf
Februari 2015

[10] Turck
http://www.turck.se/
Senast besökt maj 2015

[11] AB FIA
http://www.fia.se/
Senast besökt: maj 2015

9 Bilagor

9.1 Manual för automatiserat läge

Manual for auto-control

"How to build your command-string"

Command syntax:
To write your string you will set up a number of commands together to build a string for your PLC through FTP (File Transfer Protocol). All commands start with the command type; stirrer, pump, heater etc. These commands have a specific abbreviation (ST, SP, IH and so on). After the command you type the number of the port you want to send the commands through. The user need to know which port number the hardware is set to. All commands start with these two sets of syntaxes (Command and Port) and after this your devices has different kinds of follow up commands for settings.

How to upload your excel-file through FTP:
First you need to save your excel-document with your commands as a .CSV-file. Download a program that can handle FTP-transfers (Core FTP LE for example) and upload your document to the VT-250. When the CSV-file is uploaded you can start your program.

Stirrer
Command syntax: <Command><Port><SetSpeed>
Command: ST
Port: 1 / 2
Set speed 0 – 2000 rpm
Command example:
"ST 1 200" – This will set the stirrer at port 1 to 200 rpm.
NOTE: To stop the stirrer type 0 at SetSpeed.

Immersion Heater
Command syntax: <Command><Port><SetTemp>
Command: IH
Port: 1 / 2
Set Temperature: 0 – 150 °C
Command example:
"IH 2 35" – This will set the immersion heater to 35 °C at port 2.

Syringe Pump

Command syntax: <Command><Port><StringCommand><Unit><Value>
Command: SP
Port: 1 / 2
String command:

INF – Sets the pump to INFUSE.
Unit: 0.
Value 0.

WDR – Sets the pump to WITHDRAW.
Unit: 0.
Value 0.

DIA – Sets the pump diameter.
Unit: 0.
Value 0 – X mm.

RATC – Sets the pumping rate.
Unit: UM (µL/min)
MM (mL/min)
UH (µL/hour)
MH (mL/hour)
Value 0 – X.

VOL* – Sets the volume to be dispensed.
Unit: ML (ml)
UL (µl)
Value: 0 – X
**You can only set one of unit/value each time you type VOL.*
*For example you type **SP 1 VOL ML 0** or **SP 1 VOL 0 20**.*

CLD – Clears the volume dispensed.
Unit: INF (volume infused)
WDR (volume withdrew)
Value: 0.

Command example:

"SP 1 INF 0 0" – This will set the pump to infuse at port 1.
"SP 1 RATC MM 20" – This will set the pump rate to 20 mL/min at port 1.
"SP 1 VOL 0 28" – This will set the volume to be dispensed to 28.
"SP 1 CLD INF 0" – This will clear the volume infused at port 1.

Valves

Command syntax: <Command><ValveNumber><SetTime>
Command: VA
Valve Number: 1 – 7
Set Time*: 0 – X sec

**If SetTime is 0 the valve will activate with no time limit and is closed by the same command.*

Command example:

"VA 13 10" – This will activate valves 1 and 3 for 10 seconds.

"VA 1234 0" – This will activate valves 1, 2, 3 and 4 until you type in the same command to close them.

Repetitions

Command syntax: <Command><Times><WhichBlocks>
Command: RP
Times: 1 – N
Which Blocks: 1 – N

Command example:

"RP 3 4" – This will set the following four commands to repeat themselves in a sequence three times.

NOTE: You cannot do a set of repetitions inside another repetition.

Timer

Command syntax: <Command><SetTime>
Command: TI
Set Time: 0 – X sec

Command example:

"TI 20" – This will hold/delay your string sequence at a certain point until the 20 seconds are counted.

9.1.1 Excel-example:

Sequence 1				
<Date>				
ST	2	600		
SP	1	DIA	0	27
SP	1	RATC	MM	22
SP	1	WDR	0	0
SP	1	VOL	0	10
SP	1	RATC	MM	22
SP	1	INF	0	0
SP	1	VOL	0	10
ST	2	400		
RP	4	3		
VA	12	0		
VA	34	30		
ST	2	1000		
ST	2	400		
TI	15			
ST	2	0		

10.7 Syringe Diameters and Rate Limits

Syringe Manufacturer (all names ™)	Size (mL)	Inside Diameter (mm)	Maximum Rate (mL/hr)	Minimum Rate (µL/hr)	Maximum Rate (mL/min)
B-D	1	4.699	53.07	0.73	0.884
	3	8.585	177.1	2.434	2.952
	5	11.99	345.5	4.748	5.758
	10	14.43	500.4	6.876	8.341
	20	19.05	872.2	11.99	14.53
	30	21.59	1120	15.4	18.67
	60	26.59	1699	23.35	28.32
HSW Norm-Ject	1	4.69	52.86	0.727	0.881
	3	9.65	223.8	3.076	3.73
	5	12.45	372.5	5.119	6.209
	10	15.9	607.6	8.349	10.12
	20	20.05	966.2	13.28	16.1
	30	22.9	1260	17.32	21
	50	29.2	2049	28.16	34.15
Monoject	1	5.74	79.18	1.088	1.319
	3	8.941	192.1	2.64	3.202
	6	12.7	387.6	5.326	6.46
	12	15.72	593.9	8.161	9.899
	20	20.12	972.9	13.37	16.21
	35	23.52	1329	18.27	22.15
	60	26.64	1705	23.44	28.42
	140	38	3470	47.69	57.84
Terumo	1	4.7	53.09	0.73	0.884
	3	8.95	192.5	2.646	3.208
	5	13	406.1	5.581	6.769
	10	15.8	600	8.244	10
	20	20.15	975.8	13.41	16.26
	30	23.1	1282	17.63	21.37
	60	29.7	2120	29.13	35.33
Poulten & Graf (Glass)	1	6.7	107.8	1.483	1.798
	2	8.91	190.8	2.622	3.18
	3	9.06	197.2	2.711	3.288
	5	11.75	331.8	4.559	5.53
	10	14.67	517.2	7.107	8.62
	20	19.62	925.2	12.72	15.42
	30	22.69	1237	17.01	20.62
	50	26.96	1746	24.01	29.11
Steel Syringes	1	9.538	218.6	3.005	3.644
	3	9.538	218.6	3.005	3.644
	5	12.7	387.6	5.326	6.46
	8	9.538	218.6	3.005	3.644
	20	19.13	879.5	12.09	14.65
	50	28.6	1965	27.01	32.76

	Size (µL)	Inside Diameter (mm)	Maximum Rate (µL/hr)	Minimum Rate (µL/hr)	size (mL)	Inside Diameter (mm)	Maximum Rate (mL/hr)	Minimum Rate (µL/hr)
SGE (gas tight)	5	0.343	282.7	0.004	0.25	2.303	12.74	0.176
	10	0.485	565.3	0.008	0.5	3.257	25.49	0.351
	25	0.728	1273	0.018	1	4.606	50.99	0.701
	50	1.03	2549	0.036	2.5	7.284	127.5	1.752
	100	1.457	5102	0.071	5	10.3	254.9	3.504
Hamilton Microliter	0.5	0.103	25.49	0.001	10	14.57	510.2	7.01
	1	0.146	51.23	0.001	25	23.03	1274	17.52
	2	0.206	101.9	0.002	50	27.5	1817	24.98
	5	0.326	255.4	0.004	100	34.99	2942	40.43

Syringepumpens diameter- och hastighetsbegränsningar.
Referens: http://www.syringepump.com/download/NE-500%20OEM%20Syringe%20Pump%20User%20Manual%20V3.9.pdf

9.2 Com_List

```
P[9] := TRUE;

CASE uiSekvensP[13] OF

0:  // Kollar och jämför strängarna
InComString := InComArray[InComInt];

IF  InComString = 'ST' THEN                                     // Stirrer
                IF tmp > 0 AND Block_qntr > 0 THEN      // Kontrollerar om repetitioner skall
utföras
                                IF (Block_w > 0) THEN
                                Block_w := Block_w-1;
                                Block_qntr := Block_qntr-1;
                                uiSekvensP[13] := 10;           // Sätter igång stirrern om repetioner
skall utföras
                                ELSE
                                InComInt := InComInt-(counter)
                                Block_w := tmp;
                                counter := 0;
                                uiSekvensP[13] := 0;            //Om ifsatsen inte stämmer blir nästa
chase 0
                                END_IF
                ELSE
                counter := 0;
                uiSekvensP[13] := 10;           //Sätter igång stirrern
END_IF

ELSIF InComString = 'IH' THEN               // Immersionheater

                IF tmp > 0 AND Block_qntr > 0 THEN
                                IF (Block_w > 0) THEN
                                Block_w := Block_w-1;
                                Block_qntr := Block_qntr-1;
                                uiSekvensP[13] := 20;
                                ELSE
                                InComInt := InComInt-(counter);
                                Block_w := tmp;
                                counter := 0;
                                uiSekvensP[13] := 0;
                                END_IF
                ELSE
                counter := 0;
                uiSekvensP[13] := 20;
                END_IF

ELSIF InComString = 'SP' THEN               // SyringePump

                IF tmp > 0 AND Block_qntr > 0 THEN
                                IF (Block_w > 0) THEN
                                Block_w := Block_w-1;
                                Block_qntr := Block_qntr-1;
                                uiSekvensP[13] := 30;
```

```
                        ELSE
                        InComInt := InComInt-(counter);
                        Block_w := tmp;
                        counter := 0;
                        uiSekvensP[13] := 0;
                        END_IF
                ELSE
                counter := 0;
                uiSekvensP[13] := 30;
                END_IF

ELSIF InComString = 'LG' THEN           // Logger

                IF tmp > 0 AND Block_qntr > 0 THE
                        IF (Block_w > 0) THEN
                        Block_w := Block_w-1;
                        Block_qntr := Block_qntr-1;
                        uiSekvensP[13] := 40;
                        ELSE
                        InComInt := InComInt-(counter);
                        Block_w := tmp;
                        counter := 0;
                        uiSekvensP[13] := 0;
                        END_IF
                ELSE
                counter := 0;
                uiSekvensP[13] := 40;
                END_IF

ELSIF InComString = 'VA' THEN           // Valves
                IF tmp > 0 AND Block_qntr > 0 THEN
                        IF (Block_w > 0) THEN
                        Block_w := Block_w-1;
                        Block_qntr := Block_qntr-1;
                        uiSekvensP[13] := 50;
                        ELSE
                        InComInt := InComInt-(counter);
                        Block_w := tmp;
                        counter := 0;
                        uiSekvensP[13] := 0;
                        END_IF
                ELSE
                counter := 0;
                uiSekvensP[13] := 50;
                END_IF

ELSIF InComString = 'RP' THEN           // Repetitions
                IF tmp > 0 AND Block_qntr > 0 THEN
                        IF (Block_w > 0) THEN
                        Block_w := Block_w-1;
                        Block_qntr := Block_qntr-1;
                        uiSekvensP[13] := 60;
                        ELSE
                        InComInt := InComInt-(counter);
                        Block_w := tmp;
```

```
                    counter := 0;
                    uiSekvensP[13] := 0;
                    END_IF
            ELSE
            counter := 0;
            uiSekvensP[13] := 60;
            END_IF

ELSIF InComString = 'TI' THEN           // Timer
            IF tmp > 0 AND Block_qntr > 0 THEN
                    IF (Block_w > 0) THEN
                    Block_w := Block_w-1;
                    Block_qntr := Block_qntr-1;
                    uiSekvensP[13] := 70;
                    ELSE
                    InComInt := InComInt-(counter);
                    Block_w := tmp;
                    counter := 0;
                    uiSekvensP[13] := 0;
                    END_IF
            ELSE
            counter := 0;
            uiSekvensP[13] := 70;
            END_IF

ELSIF InComArray[InComInt] = '%' THEN// Stopp tecken, stänger av blocket när detta tecken
dyker upp om inte repetioner skall utföras
            IF (Block_qntr = 0) THEN
            P[13] := FALSE;
            END_IF
                    IF tmp > 0 AND Block_qntr > 0 THEN
                            IF (Block_w > 0) THEN
                            Block_w := Block_w-1;
                            Block_qntr := Block_qntr-1;
                            ELSE
                            InComInt := InComInt-(counter);
                            Block_w := tmp;
                            counter := 0;
                            END_IF
                    END_IF
            END_IF



10:  //Omrörare ST
            InComInt := InComInt + 1; // Räknar igenom array listan
            RS232_Port := STRING_TO_INT(InComArray[InComInt]); //Ställer in RS232
Porten
            InComInt := InComInt + 1;
            Stirring_SPEED := STRING_TO_INT(InComArray[InComInt]); // Ställer in
hastigheten för stirrern
            InComInt := InComInt + 1;
            counter := counter + 3;
            P[8] := TRUE;
            uiSekvensP[13] := 11;
```

```
11: // Väntar på att Stirrern ska bli false
            IF P[8] = FALSE THEN
                        uiSekvensP[13] := 0;
            END_IF

20:  //Termostat IH
            InComInt := InComInt + 1;
            RS232_Port := STRING_TO_INT(InComArray[InComInt]);
            InComInt := InComInt + 1;
            SetTemperature := STRING_TO_INT(InComArray[InComInt]); // Ställer in
temperaturen
            InComInt := InComInt + 1;
            counter := counter + 3;
            P[12] := TRUE;
            uiSekvensP[13] := 21;

21:  // Väntar på att termostaten ska bli False
            IF P[12] = FALSE THEN
                        uiSekvensP[13] := 0;
            END_IF

30:         //Pump
            InComInt := InComInt + 1;
            RS232_Port := STRING_TO_INT(InComArray[InComInt]);
            InComInt := InComInt + 1;

            // Kontrollerar vilket kommando som skall utföras
            IF InComArray[InComInt] = 'RUN' THEN
                        Syringe_COM := 1;
                        InComInt := InComInt + 3;

            ELSIF InComArray[InComInt] = 'STP' THEN
                        Syringe_COM := 2;
                        InComInt := InComInt + 3;

            ELSIF InComArray[InComInt] = 'INF' THEN
                        Syringe_COM := 4;
                        InComInt := InComInt + 3;

            ELSIF InComArray[InComInt] = 'WDR' THEN
                        Syringe_COM := 5;
                        InComInt := InComInt + 3;

            ELSIF InComArray[InComInt] = 'DIA' THEN
                        Syringe_COM := 6;
                        InComInt := InComInt + 2;
                        Syringe_DIA := STRING_TO_REAL(InComArray[InComInt]);
                        InComInt := InComInt + 1;

            ELSIF InComArray[InComInt] = 'RATC' THEN
                        Syringe_COM := 7;
                        InComInt := InComInt + 1;

                        IF InComArray[InComInt] = 'UM' THEN
```

```
                        Syringe_RATE := 1;
                ELSIF InComArray[InComInt] = 'MM' THEN
                        Syringe_RATE := 2;
                ELSIF InComArray[InComInt] = 'UH' THEN
                        Syringe_RATE := 3;
                ELSIF InComArray[InComInt] = 'MH' THEN
                        Syringe_RATE := 4;
                END_IF

                InComInt := InComInt + 1;
                Syringe_FLOW := STRING_TO_REAL(InComArray[InComInt]);
                InComInt := InComInt + 1;

        ELSIF InComArray[InComInt] = 'VOL' THEN
                Syringe_COM := 8;
                InComInt := InComInt + 1;

                IF InComArray[InComInt] = 'ML' THEN
                        Syringe_VOL_UNIT := 1;
                ELSIF InComArray[InComInt] = 'UL' THEN
                        Syringe_VOL_UNIT := 2;
                END_IF

                InComInt := InComInt + 1;
                Syringe_VOL := STRING_TO_REAL(InComArray[InComInt]);
                InComInt := InComInt + 1;

                IF Syringe_INWI = 'Infusing' THEN
                        Syring_Vol_Auto := Syring_Vol_Auto - Syringe_Vol;
                ELSIF Syringe_INWI = 'Withdrawing' THEN
                        Syring_Vol_Auto := Syring_Vol_Auto + Syringe_Vol;
                END_IF

        ELSIF InComArray[InComInt] = 'CLD' THEN
                Syringe_COM := 9;
                InComInt := InComInt + 1;

                IF InComArray[InComInt] = 'INF' THEN
                        Syringe_CLEAR_DIS := 1;
                ELSIF InComArray[InComInt] = 'WDR' THEN
                        Syringe_CLEAR_DIS := 2;
                END_IF

                InComInt := InComInt + 2;

        END_IF

        counter := counter + 5;
        P[6] := TRUE;
        uiSekvensP[13] := 31;

31:  // Väntar på att pumpen ska bli False
        IF P[6] = FALSE THEN
                uiSekvensP[13] := 0;
        END_IF
```

```
40:  // Logger
		InComInt := InComInt + 1;
		RS232_Port := STRING_TO_INT(InComArray[InComInt]);
		InComInt := InComInt + 1;
		counter := counter + 2;
		uiSekvensP[13] := 41;

41:  // Väntar på att loggern ska bli False
		IF P[12] = FALSE THEN
			uiSekvensP[13] := 0;
		END_IF

50: //Ventiler
		InComInt := InComInt + 1;
		// Kontrollerar vilka ventiler som skall öppnas/stängas
		FOR i := 1 TO LEN(InComArray[InComInt]) DO
			tempString :=
BYTE_TO_STRING(CODE(InComArray[InComInt],i));
			IF tempString = '49' THEN
				Valve_1 := 1;
			END_IF

			IF tempString = '50' THEN
				Valve_2 := 1;
			END_IF

			IF tempString = '51' THEN
				Valve_3 := 1;
			END_IF

			IF tempString = '52' THEN
				Valve_4 := 1;
			END_IF

			IF tempString = '53' THEN
				Valve_5 := 1;
			END_IF

			IF tempString = '54' THEN
				Valve_6 := 1;
			END_IF

			IF tempString = '55' THEN
				Valve_7 := 1;
			END_IF

			tempString := '';
		END_FOR;

		InComInt := InComInt + 1;
		Valve_timer := STRING_TO_INT(InComArray[InComInt]);		//Ställer in
timern, om 0 triggas ventilen
		InComInt := InComInt + 1;
		counter := counter + 3;
```

```
                P[11] := TRUE;
                uiSekvensP[13] := 51;
51:
                 IF P[11] = FALSE THEN
                                uiSekvensP[13] := 0;
                 END_IF


60: // Repetitions, kontrollerar hur många repetioner som skall utföras
                InComInt := InComInt + 1;
                Block_qntr := STRING_TO_INT(InComArray[InComInt]);
                InComInt := InComInt + 1;
                Block_qntr := Block_qntr*STRING_TO_INT(InComArray[InComInt]);
                Block_w := STRING_TO_INT(InComArray[InComInt]);
                tmp := Block_w;
                InComInt := InComInt + 1;
                uiSekvensP[13] := 0;

70: // Timer, ställer in en timer som en delay i programmet
                InComInt := InComInt + 1;
                timerInt := STRING_TO_INT(InComArray[InComInt]);
                InComInt := InComInt +1;
                counter := counter + 2;
                uiSekvensP[13] := 71;

71: // Inväntar timern
                timer(IN := (NOT timer.Q), PT:= SECOND_TO_TIME(INT_TO_REAL(timerInt)));
                IF timer.Q = TRUE THEN
                                uiSekvensP[13] := 0;
                END_IF

END_CASE
```