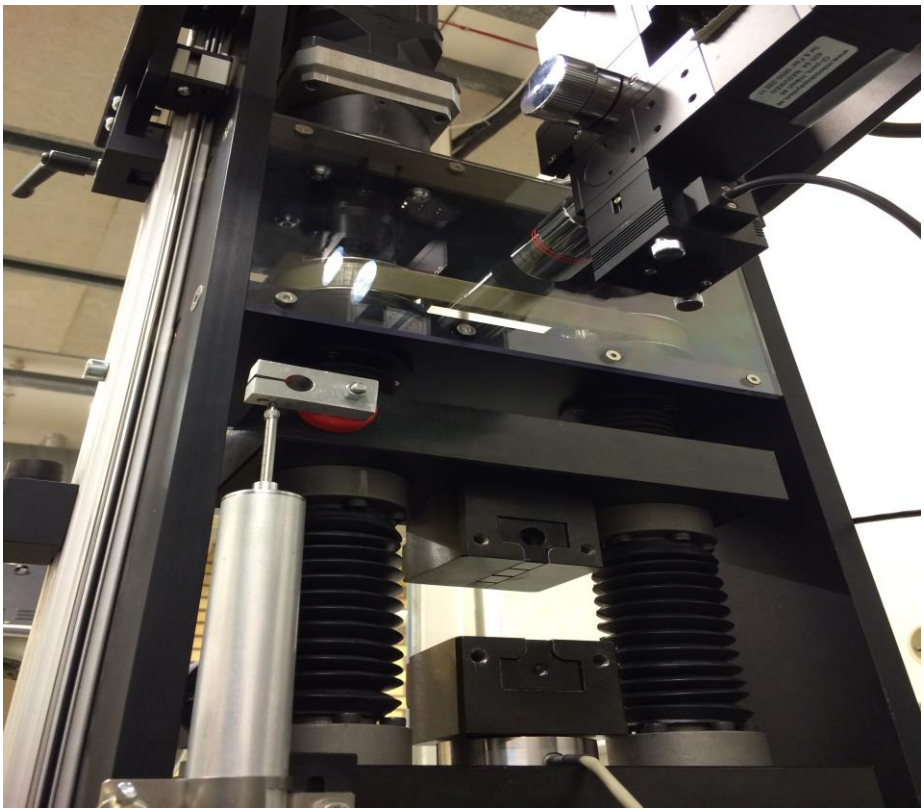
CHALMERS



Utveckling av ett LabVIEW-program för styrning av en servomotor vid mekanisk provning

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

ANDREAS SALOMONSSON

CALLE ARKEVALL

Institutionen för Signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Förord

Projektet som beskrivs i denna rapport har utförts som ett examensarbete i slutet av Mekatronikprogrammet, programmet omfattar 180 högskolepoäng. Examensarbetet utfördes på Institutionen för Signaler och System på Chalmers Tekniska Högskola vid Lindholmen. Arbetet utfördes på avdelningen Bygg och Mekanik på SP Sveriges Tekniska Forskningsinstitut i Borås.

Vi vill tacka för möjligheten till att få göra vårt examensarbete på SP. Vi vill framförallt tacka Torsten Sjögren som var vår handledare på SP och David Samuelsson som hjälpt oss under arbetets gång. Vi vill även tacka alla övrig personal på SP som hjälpt oss. Vi vill tacka Manne Stenberg på Chalmers för den hjälp vi fått med rapporten.

Projektet lämpade sig väl mot vår utbildning, där många kunskaper kom till användning. Med ett nytt program att arbeta med har vi lärt oss väldigt mycket vilket kommer vara till god användning i arbetslivet.

Andreas Salomonsson
Calle Arkevall

Göteborg 2014-05-27

Sammanfattning

SP Sveriges Tekniska Forskningsinstitut (SP) i Borås behövde ett program för styrning av en servomotor, då det existerande programmet inte var användarvänligt och den funktionalitet som behövdes inte existerade. Motorn användes till mekanisk provning, där den åstadkom en linjär förskjutning i ett så kallat dragsteg för materialprovning. Arbetet utfördes på avdelningen för Bygg och Mekanik på SP i Borås under en period av 10 veckor. Ett program utvecklades som styr motorn vid mekanisk provning. Programmets struktur är uppbyggd på principen att ingen onödig information skall visas samt möjlighet till en lätt vidareutveckling av programmet. Ny funktionalitet i programmet lades till där information från positionsgivare och lastcell tas in och behandlas. Resultatet är ett program som är mer användarvänligt med utökad funktionalitet och möjlighet till vidareutveckling.

Abstract

SP Technical Research Institute of Sweden in Borås needed a program that could operate a servomotor, since the existing program was not user friendly and did not have the functionality that was needed. The motor was used for mechanical testing, where it produced a linear displacement in a so called tensile stage for mechanical testing. The work was performed at the Department of Structural and Solid Mechanics at SP in Borås for a period of 10 weeks. A program was developed that controls the motor used at mechanical testing. The program structure is based on the principle that no unnecessary information shall be shown and there should be a possibility to develop the program further. Additional functionality was added to the program which collected data from position sensors and a load cell. The result is a program that is more user friendly with increased functionality and has a potential for further development.

Förkortningar

GUI = Graphical user interface

rpm= Rotations per minute (varv per minut).

RS-232 = Recommended standard 232.

RS-485 = Recommended standard 485.

SP = SP Sveriges Tekniska Forskningsinstitut

PM = permanent magnet

VI = Virtuella Instrument

Innehållsförteckning

1 Inledning

1.1 Bakgrund

SP i Borås har en äldre version av ett styrprogram som styr en stegmotor vid mekanisk provning. Man vill utveckla programmet så att det skrivs i aktuell LabVIEW-version, då det befintliga programmet är skrivet i en gammal version och källkoden ej går att få tag på, samt implementera fler funktioner för att möjliggöra t.ex. provning i last- eller töjningskontroll.

1.2 Syfte

Arbetet går ut på att skriva ett nytt styrprogram (i aktuell LabVIEW-version) som styr en stegmotor vid mekanisk provning samt lägga till ökad funktionalitet vid provning, såsom styrning mot återkopplad last eller töjning. Resultatet ska vara ett fungerande program som kan styra mot en återkopplad last eller töjning. En likadan stegmotor används även vid provning av lås. I arbete skall även styrprogram skrivas för denna typ av provning.

1.3 Avgränsningar

Programmet skall styra motorn vid mekanisk provning med en konstant deformationshastighet och eventuellt laststyrt/förskjutningsstyrt. Till dragprovningsutrustningen finns en kamera som kan ta bilder från vilka töjningsfält kan beräknas. Insamling av bilder och behandling av bilderna utförs inte på grund av den begränsade tiden.

1.4 Precisering av frågeställning

Följande frågeställningar har hanterats:

1. Hur styrs motorn och vilken funktionalitet har dagens program?
    - Vilka parametrar ska kunna ändras på?
    - Vilka funktioner är viktiga?
    - Hur ska ett användarvänligt GUI vara uppbyggt?

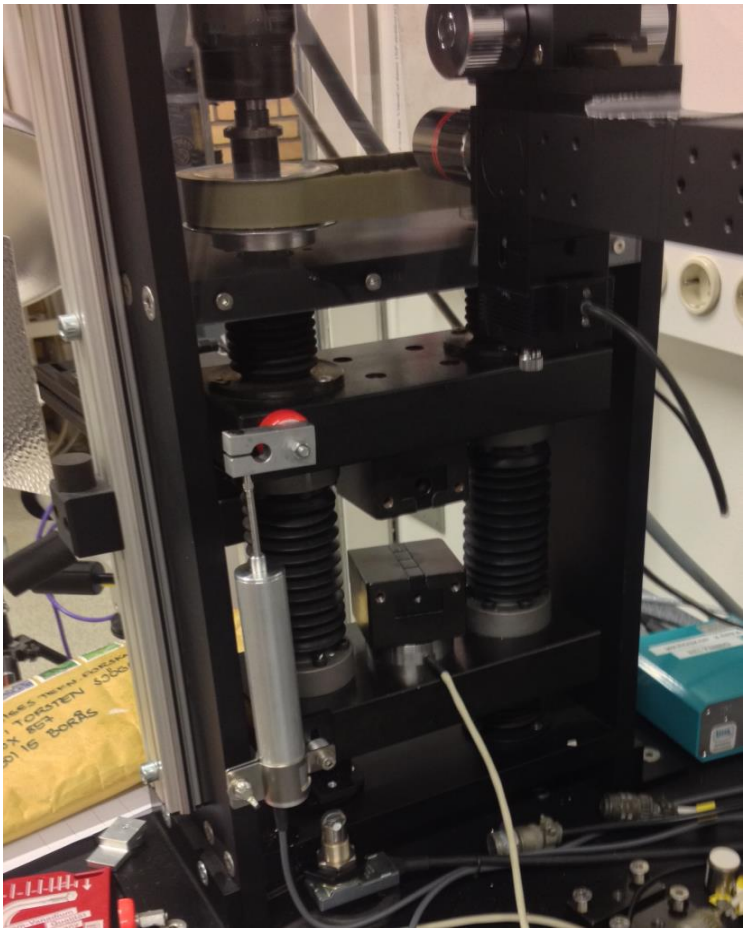
2. Hur åstadkoms last- och töjningsstyrd mekanisk provning?
    - Hur tas signaler om last och förskjutningar in i programmet?
    - Hur fungerar last och töjningsstyrd provning i teorin?
    - Hur ska ett användarvänligt GUI vara uppbyggt?

2 Teknisk och Teoretisk bakgrund

Följande kapitel ger en beskrivning av den utrustning och det program som används vid framtagning av styrprogrammet för dragsteget samt styrprogrammet för provning av lås.

2.1 Utrustning för mekanisk provning

Utrustningen för den mekaniska provningen består av en motor som roterar två skruvar vilka i sin tur förflyttar två plattor. Detta gör det möjligt att mekaniskt prova material under både drag- eller tryckbelastning.



Figur 1. Utrustning för mekanisk provning.

Stegmotorn sitter direktmonterad mot en av skruvarna som kan ses i Fig.1, rotationen förs även över till den andra skruven med hjälp av en rem. Skruvarna är höger- och vänstergängade där den ena halvan av skruven är högergängad och den andra delen är vänstergängad. Båda skruvarna är monterade i samma riktning vilket gör att de två plattorna som sitter fast i skruvarna antingen flyttar sig mot varandra eller ifrån varandra beroende på vilken riktning motorn roterar med. De två skruvarna har en gängstigning på 1 mm/varv vilket leder till att förskjutningen mellan de två plattorna ökar med två 2 mm/varv [1]. Förskjutningshastigheten blir således dubbelt så stor som rotationshastigheten på motorn.

Mellan de två plattorna sitter det två mindre block som är till för att hålla fast provbiten under provning. Under det nedre blocket sitter en lastcell för registrering av den axiella last som provet känner av (se Fig. 1). I Fig. 1 ser man även den större av positionsgivarna till vänster på ramen. Positionsgivaren registrerar den förskjutning som sker mellan de båda plattorna som provet är monterat mellan under provning. Vid dragprovet finns det en inbyggd säkerhetsbrytare som stoppar motorn då den nedre plattan slår emot denna. Brytaren syns under den vänstra positionsgivaren i Fig. 1. Brytaren skickar en signal till motorn som stoppar dess rotationsrörelse. Brytaren är endast verksam vid det yttre ändläget dvs. bryter provet när man provar under dragbelastning. I tryckbelastning finns det inte någon hårdvarusäkerhet.

2.1.1 Servomotor och BIVECTOR 500

Motorn som används vid dragprovning är en AC permanentmagnet (PM) synkron servomotor. På motorns rotor sitter ett antal permanentmagneter och runt om i höljet sitter ett antal lindade poler. Spolarna spänningsätts efter varandra i en viss riktning, detta skapar ett magnetfält som roterar med eller moturs [2]. Permanentmagneterna drar sig då efter magnetfältet och skapar rotation på rotorn. Hastigheten som motorn roterar i är den samma som magnetfältets.

Motorer som använder sig av PM behöver en styrenhet som är tillverkad och anpassad för just den aktuella motorn. Styrenheten tar hand om den ström- och spänningsmatning som går till motorn. Den styrenhet som hör till motorn som används i detta arbete heter BIVECTOR 500. Med hjälp av en styrenhet kan ett antal parametrar ändras på, vilka parametrar som kan ändras varierar mellan olika styrenheter. Några exempel på parametrar som kan ändras är vridmoment, hastighet och max hastighet. Alla parametrar sparas i tabeller som finns i BIVECTOR, det finns totalt 64 stycken tabeller där inställningar av parametrar kan sparas [3].

I vanliga fall beräknas motorns position genom av att mäta den ström och spänning som skickas till motorn. I det aktuella fallet för detta arbete används återkoppling vilket gör de möjligt att läsa av motorns exakta position och hastighet.

Till BIVECTOR skickas kommandon med instruktioner om vilken parameter som skall ändras eller om motorn ska stanna/starta. Detta sker med hjälp av seriell kommunikation från en extern enhet, vanligtvis en dator.

2.2 Utrustning för provning av lås

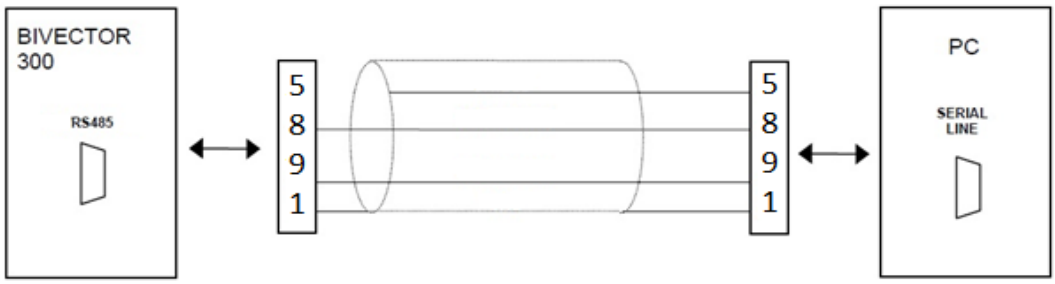
Utrustningen för provning av lås är en stålplatta med infästning av motorn samt låsmekanismen. Stålplattan används för att säkerställa att motorns vridaxel centreras mot låsmekanismens låsaxel. Då motorn roterar vrids låsaxeln och låsmekanismen prövas.

2.3 Kommunikation

Kommunikation med BIVECTORN sker via den seriella porten och den kan endast hantera två stycken protokoll, RS-232 och RS-485. Följande kapitel förklarar hur olika kommandon skickas med de två standarderna.

2.3.1 RS-232 & RS-485

Recommended standard 232 (RS-232) och Recommended standard 485 (RS-485), även kallad TIA/ANSI/ESI-485, är två standarder vilka definierar de elektriska egenskaperna hos sändaren och mottagaren vid seriell kommunikation [4]. Kommunikationen sker via en comportskabel med 9st ledare. I det aktuella fallet används endast fyra ledare. Bilden nedan (Fig. 2) illustrerar hur kopplingen mellan styrenheten och datorn fungerar då RS-485 används.



Figur 2. RS-485 kommunikation mellan BIVECTOR 500 och PC.

Tabell 1 visar vilka ledare som används av de olika protokollen och vilken uppgift varje ledare har.

Tabell 1. Användning av ledare för protokoll RS-232 och RS-485.

Ledar nummer	**Uppgift**	**Beteckning**
RS-232 & 485		
1	Skärm	Shield.
5	Gemensam jord.	GND.
RS-485		
8	Skicka data/Ta emot data	Tx/Rx +
9	Skicka data/Ta emot data	Tx/Rx -
RS-232		
2	Skicka data	Tx
3	Ta emot data	Rx

Då RS-232 eller RS-485 används för att kommunicera med en BIVECTOR så utformas de kommandon som skickas enligt följande [5]:

&<DD><II><data field>/<CC>

Eller

&<DD><II><data field> =

Där:

&= Det första tecknet betecknar att en kommandosträng kommer samt identifierar om det är RS-232 eller RS-485 kommunikation, ”*” står för RS-232och ”&” står för RS-485.

DD= Står för vilken/vilka drivers som kommandot är menat åt. De möjliga adresserna är 00-FF. Då FF används skickas kommandot till alla drivers oavsett vilken adress den är på.

II= Operationskod

data field= Vissa operationskoder kräver att en adress eller ett värde skickas med och data field är det värdet/adressen.

”/” eller ” =” är de tecken som avslutar kommandosträngen, skillnaden mellan tecknen är att ”/” säger att det kommer en checksumma efteråt medan ”=” säger att det inte kommer en checksumma.

CC= Checksumma.

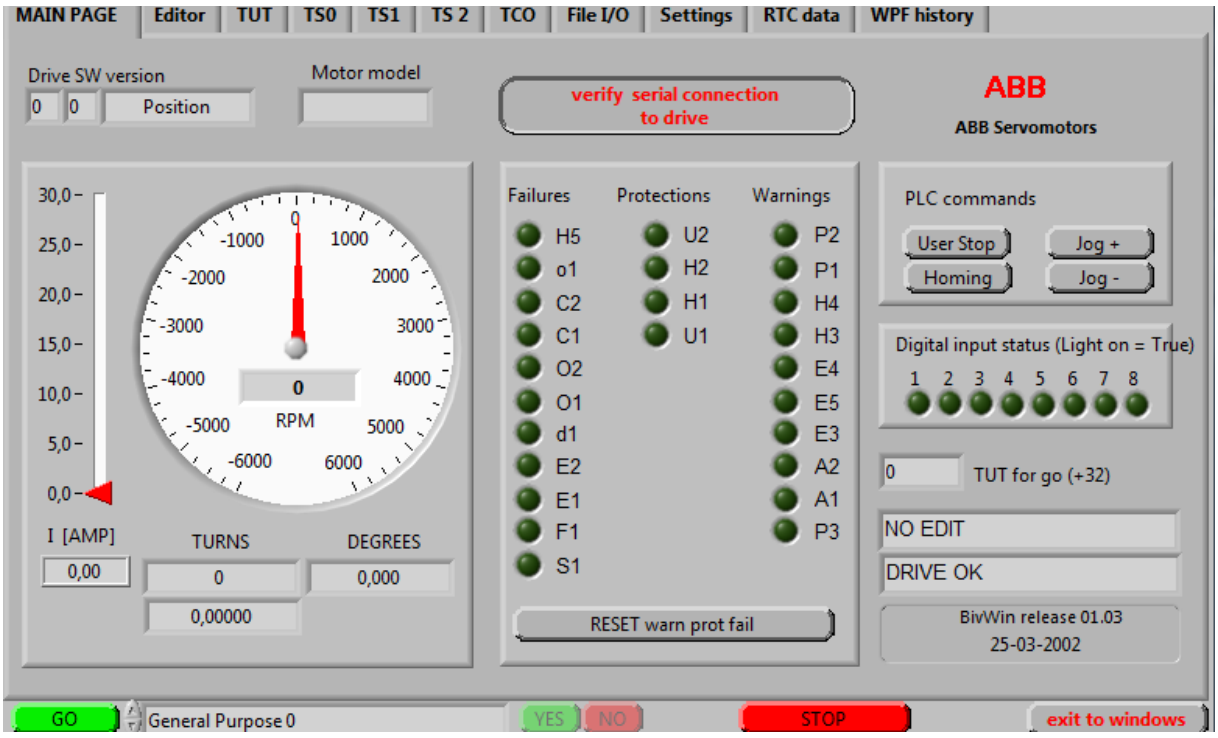
Exempel på kommandosträngar ges i Bilaga C.

2.4 Program

Ett program som tidigare användes för att skicka kommandon till BIVECTOR var BivWin, ett program utvecklat i LabVIEW. Även det nya programmet är utvecklat i LabVIEW.

2.4.1 BivWin

BivWin är ett program som är utvecklat av ABB Italien och är gjort för att styra en motor genom att skicka kommandon till BIVECTOR. BivWin är utvecklat i en äldre version av LabVIEW och är från år 2002.

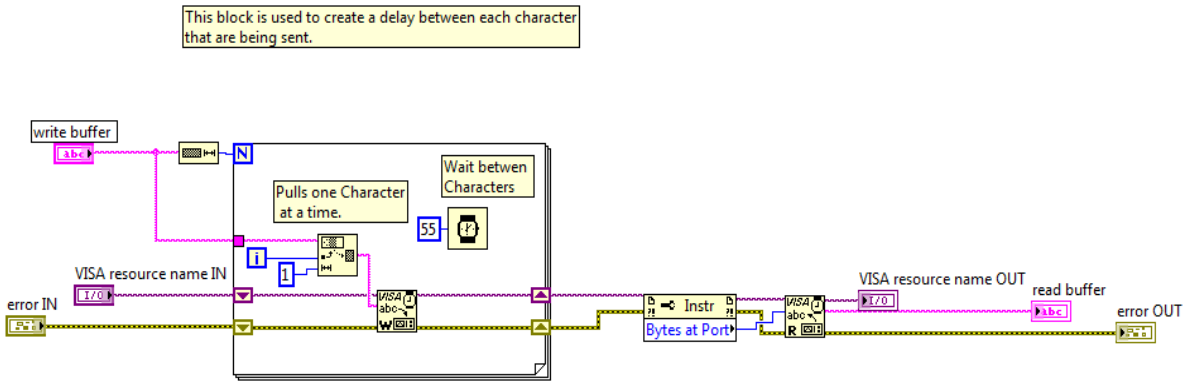


Figur 3. BivWin-layout.

Programmet är uppbyggt i ett antal olika flikar, som kan ses i Fig.3, där olika inställningar kan göras. Alla parametrar i BIVECTOR kan ändras från detta program. De värden som skrivs in på de olika parametrarna stämmer inte överens med de verkliga värdena. Det finns omskalningar mellan värdet som skickas in till BIVECTOR och det riktiga värdet för att rätt spänning ska mata motorn.

2.4.2 LabVIEW

LabVIEW är ett populärt grafiskt programmeringsverktyg som används för testutrustningar och datainsamling då det fungerar med de flesta gränssnitt [6]. Program som utvecklas i LabVIEW byggs upp med hjälp av olika block kallade ”Virtuella Instrument” (VI), där varje block har en viss funktion med ett antal ingångar och utgångar. De olika blocken som anropas binds ihop med varandra med hjälp av ledningar. Så fort ett block har fått all information det behöver för att utföra sin funktion utförs den, alltså kan många blocks funktioner utföras samtidigt. Vad som ska utföras och när kan begränsas med hjälp av olika ramar. Några av de ramar som finns tillgängliga vid programmering är ”while”, ”for” samt ”sequence”.



Figur 4. Exempel på VI.

I Fig.4 visas ett exempel på hur ett VI kan se ut. Detta är ett VI vilket är uppbyggt av grundblock som finns med i LabVIEW. Detta VI kan nu omvandlas till ett block med in och ut signaler vilket sen kan anropas likadant som ett grundblock. En teknik som rekommenderas när program byggs i LabVIEW är att de byggs upp i flera nivåer för att göra det lättförståeligt och lätta att bygga vidare på. Det som kan ses i Fig.4 är den lägsta nivå som kan ses i programmet då det endast är uppbyggt av block som finns i LabVIEW. I den högsta nivån ska det till största del bestå av färdiga funktioner där ett exempel är ”kör motorn med X varv/s” där det endast finns X som insignal och funktionen sköter sedan resten.

Det som beskrivits hittills är ”programmeringsdelen” av programmet. Varje VI består av två olika fönster, diagramfönstret och panelfönstret. I diagramfönstret finns programkoden i form av block och ledningar och i panelfönstret läggs allt som en användare ska se vid körning. När ett program körs ser andvändaren ett grafiskt gränssnitt uppbyggt av textboxar, grafer och allt som kan tänkas vara väsentligt att se. Alla textboxar och liknande som skapas agerar som in- och utsignaler till programmeringsfönstret.

3 Metod

Arbetet inleds med granskning av befintlig dokumentation av stegmotor, styrning och program för att få en förståelse för motorstyrningen samt hur kommunikationen mellan motorn och datorn fungerar. Efter det att denna information har tagits fram påbörjas programmering av motorstyrningen där motorn, i ett första skede, endast ska rotera i en bestämd hastighet i vald riktning.

Även en kartläggning av eventuell existerande LabVIEW-kod för styrning av den aktuella motorn ska genomföras där andra motortillverkare och annan kod tillgänglig på Internet undersöks.

Efter att enkel styrning av motorn har åstadkommits påbörjas arbetet med återkoppling och hur last- och töjningssignaler ska tolkas. När de återkoppade signalerna kan tolkas börjar arbetet med att programmera extrafunktionerna enligt den kravspecifikation som tagits fram i samråd med SP.

4 Kravspecifikation

Följande kravspecifikation har tagits fram som underlag till det arbete som ska utföras. I specifikationen anger (K) att det är ett krav och (Ö) att det är ett önskemål.

- **Det skall gå att kommunicera med BIVECTOR. (K)**
  Från det program som utvecklas skall det vara möjligt att skicka kommandon till BIVECTOR som ändrar på parametrar samt startar och stoppar motorn.
- **Enkel styrning av motorn. (K)**
  Det skall gå att styra motorn mot en given hastighet eller till en viss position.
- **Insamling av givardata. (K)**
  I programmet skall det samlas in data från tre stycken olika givare (en last- och två positionsgivare), informationen skall behandlas och visas i form av grafer och värden i GUI.
- **Reglerad körning. (Ö)**
  Köra motorn efter en konstant last- eller deformationshastighet.
- **Användarvänligt program. (K)**
  Programmet skall vara lätt att förstå och inga onödiga inställningar skall finnas att göra.
- **Bra struktur på programmet, ska vara lätt att bygga vidare på. (K)**
  Programmet skall vara uppbyggt på ett sätt så att det är lätt att bygga vidare på, samt att det ska vara lätt att hitta i programmet om det är något som behöver ändras.

5 Genomförande

Nedan följer ett antal avsnitt vilka förklarar de olika stegen som genomförts för att komma fram till det resultat som åstadkommits.

5.1 Granskning av befintliga program och dokumentation

Arbetet inleddes med att granska det befintliga programmet och den dokumentation som fanns tillgänglig. Det befintliga programmet var ett äldre program gjort av ABB Italien som medföljde vid köpet av servo motorn. Dokumentation och program gav följande information:

- Det finns två olika sätt att kommunicera med BIVECTOR.
- Alla kommandon som skickas är i formen av en hexadecimal kod.

5.2 Upprätta kommunikation

Den befintliga dokumentationen visade att det fanns två sätt att kommunicera med BIVECTOR, RS-232 och RS-485. De två metoderna testades och utvärderades. De faktorer som var intressanta i utvärderingen var hur säker och snabb kommunikationen var.

Testet gick ut på att ett kommando skickades till BIVECTOR som returnerade ett svar vilket granskades för att se om det stämde överens med vad som angetts i kommunikationsprotokollet, samt mätning av responstiden.

Ett problem som uppstod när testet utfördes var att kommunikationen inte fungerade som den skulle, responsen som gavs av BIVECTOR varierade mellan varje gång ett kommando skickades. De felmöjligheterna som testades var följande:

- Att programmet var uppbyggt på fel sätt samt att de ”block” som användes inte användes korrekt.
- Fel på den adapter som omvandlade COM => USB-port.
- Fel format på den kommandosträng som skickades eller att fel RS-standard användes.
- Att det behövdes en fördröjning mellan varje tecken i kommandot som skickades.

Det visade sig vara en kombination med att USB-adapterns inställningar inte var korrekta samt att det behövdes en fördröjning mellan varje tecken. Anledningen till att det behövs en fördröjning mellan varje tecken är att den styrenhet som används är gammal och äldre system klarar inte alltid av att ta emot en hel kommandosträng på en gång. Detta upptäcktes genom att avlyssna den befintliga kommunikationen som skedde mellan BivWin och BIVECTOR.

Efter att en fördröjning lagts in mellan varje tecken startades testet igen. Testet visade att RS-232 och RS-485 var lika snabba på att skicka och ta emot medelanden. RS-232 returnerade inte alltid rätt svar medan RS-485 alltid returnerade rätt, det bestämdes att RS-485 skulle användas då det var mer stabilt.

Efter att kommunikationen hade börjat fungera uppkom ett nytt problem. Då ett kommando hade skickats från testprogrammet kunde inte BivWin få kontakt med BIVECTOR. För att återfå kommunikationen mellan BivWin och BIVECTOR behövdes BIVECTOR startas om. Efter det att BivWin startats om kunde testprogrammet användas men efter testprogrammet

använts så fungerade inte kommunikationen mellan BivWin och BIVECTOR. De felmöjligheter som utvärderades var följande:

- Kommunikationen avslutades inte på ett riktigt sätt i testprogrammet.
- BIVECTOR behövde ett speciellt kommando för att avsluta kommunikationen.

Efter att ABB Italien, som utvecklat BivWin, och National Instruments kontaktats fastslogs det att det inte var något fel med testprogrammet samt att det inte behövdes något speciellt kommando för att avsluta kommunikationen med BIVECTOR.

Ett test genomfördes genom att ett kommando skickades med RS-232 typen i stället för RS-485 och då fungerade kommunikationen mellan BIVECTOR och BivWin även om testprogrammet hade använts. Efter avlyssning av kommunikationen och ett antal testkommandon som skickades med både RS-232 och RS-485 visade det sig att BIVECTOR ställer om sig till det protokoll som det senaste kommandot skickades med. Om ett kommando med RS-232 skickades senast sätts BIVECTOR i RS-232 läge. BivWin testar inte om kommunikationen fungerar genom att skicka ett kommando vilket ledde till att BIVECTOR inte bytte till RS-232 läge. Problemet löstes genom att ett kommando av typen RS-232 alltid skickas när det nyutvecklade programmet avslutades.

5.3 Programmering

Programmeringen startades genom att fortsätta bygga på de testprogram som hade använts för att testa om kommunikationen fungerade mellan datorn och BIVECTOR. Det första målet var att utveckla ett program som kunde styra motorns hastighet.

Då programmet kunde styra motorn efter en angiven hastighet utvärderades omvandlingsfaktorn mellan den verkliga hastigheten och det värde som skickas till BIVECTOR. Det fanns inte någon information om hur omvandlingen skedde. Ett test utfördes där ett värde skickades in och det antal varv motorn roterade på en minut räknades. Omvandlingen $\left(\frac{värde\ till\ BIVECTOR}{verkliga\ hastighetem}\right)$ beräknades till 35,2 vilket inte var ett exakt värde men en god början.

När ett enkelt program som kunde styra hastigheten hos motorn fanns, började arbetet med att utveckla ett program som skulle styra motorn mellan två positioner. Detta gjordes för att få en större förståelse för hur motorn fungerade och vilka funktioner som kunde anropas.

När det fanns två väldigt enkla program, ett som kunde styra motorns hastighet och ett som kunde styra motorn mellan två positioner fortsatte arbetet med att strukturera upp hur de slutliga programmen skulle se ut och vara uppbyggda. För att programmen skulle vara lätta att använda och inte visa för mycket information på en och samma gång bestämdes det att programmen skulle delas in i olika flikar. För att göra arbetet effektivt och programmet lätt att hitta i gjordes alla funktioner om till egna block som kunde anropas, detta gjorde arbetet mycket mer effektivt eftersom alla färdiga funktioner kunde återanvändas.

Ett problem som uppstod när programmet som skulle användas vid provning av lås utvecklades var att kommandot för att läsa av motorns nuvarande position inte fanns med i någon av manualerna. ABB Italien kontaktades men de hade inte arbetat med BIVECTOR på över 10 år och hade inte kvar några andra manualer än de som fanns tillgängliga på SP. Arbetet fortsatte men utan denna funktion. Programmet hade börjat närma sig ett stadie där

det kunde utföra vad som efterfrågades och man kunde ange de parametrar och inställningar som önskades. Det som nu saknades var inhämtning och behandling av givardata.

På SP fanns det ett färdigt program som tog in alla värden från givarna. Denna programkod kopierades och lades in i programmet för dragsteget. Extra säkerhetsfunktioner lades in som skulle stoppa motorn då en givare närmade sig sin angivna min- eller maxgräns.

Ett program för att styra motorn laststyrt började utvecklas till ett stadie där det var nära att kunna testas men med tanke på den tid som fanns kvar beslutades att göra de fungerande programmen så bra som möjligt istället.

Arbetet med den visuella sidan av programmet påbörjades nu där layouten diskuterades med Mathias Flansbjer, Torsten Sjögren och David Samuelsson på SP.

Efter att layouten hade ändrats och programmet snyggats till mottogs ny information från ABB Italien där de skickade med den gamla källkoden för BivWin för att möjligen lista ut vilket kommandot för motorns nuvarande position var. Programmet bestod av 240 filer där all information var på Italienska, efter sökning och testning av ett antal kommandon hittades kommandot för ta reda på motorns nuvarande position. Information om hur olika omvandlingar mellan BIVECTOR och verkliga värden kunde även letas upp i programkoden.

De funktioner som inte var möjliga att utveckla tidigare var möjliga kunde nu läggas till. Layouten ändrades en del vid samma tillfälle och ett nytt möte med Torsten Sjögren, Mathias Flansbjer och David Samuelsson på SP angående en slutgiltig layout hölls.

6 Programmets uppbyggnad

Programmet som gjordes delades in i två program, ett som styr servomotorn vid dragprovning och det andra programmet som används för att styra en annan servomotor vid provning av lås. I BIVECTOR finns ett flertal tabeller där inställningar kan sparas. De två programmen är uppbyggda av ett antal flikar där varje flik använder sig av en tabell. Det enda som kan ändras i programmen ska vara de inställningar som skiljer sig mellan varje körning. De olika flikarna i programmen är följande:

1. **Styrning av dragprovning**
   - Settings
   - Configure
   - Test mode
   - Visning av givar data(statisk)

   För en mer detaljerad beskrivning av programmets uppbyggnad samt källkod se Bilaga A.

2. **Test av lås**
   - Nyckelvridning
   - Position mode

   För en mer detaljerad beskrivning av programmets uppbyggnad samt källkod se Bilaga B.

6.1 Styrning av dragprovning

Programmet för styrning av servomotor vid dragprovning är gjort för att vara enkelt att använda. Endast de parametrar och inställningar som är väsentliga för ett dragprov kan ändras. Programmet är uppbyggt i flikar och är gjort för att det enkelt ska kunna byggas vidare på. Följande kapitel beskriver vilka funktioner och inställningar som varje flik har.

6.1.1 Settings

Settings-fliken består av funktioner som ändrar inställningar för alla flikar samt några extrafunktioner som inte finns i någon av de andra flikarna. De funktioner som settings-fliken har är följande:

- Ställa in vilken COM-port som skall användas.
- Sätta max- och mingräns för vridmoment.
- En ”jog”-funktion som roterar motorn ett visst antal varv.

De block som används för att hantera seriell kommunikation finns färdiga i LabVIEW. Blocken känner av vilka COM-portar som är aktiva på datorn. För att välja vilken COM-port som skall användas är det bara att välja i en lista av de COM-portar som är aktiva.

Funktionen att ställa in min- och maxgräns av vridmoment för alla flikar utförs enligt följande steg:

1. BIVECTOR görs redo för att ta emot kommandon.
2. Första flikens tabell öppnas i BIVECTOR.
3. Ändra parametrarna ”max positive torque” och ”max negative torque”.
4. Andra flikens tabell öppnas osv..

Steg två och tre upprepas för varje flik som ändringarna skall ske i. Efter att ändringen skett i alla flikar läses vridmomentet av i den senast öppnade tabellen. Om vridmomentet har ändrats i den sista tabellen kan det antas att värdet har ändrats i alla flikar för att den har gått genom alla steg.

Jog-funktionen är en funktion som ändrar motorns position positivt eller negativt och olika mycket beroende på vilken variant som används. Det finns fyra varianter där två ökar positionen positivt kallade ”Together” och två som minskar positionen kallade ”Apart” (benämningen syftar på hur rotationen påverkar de plattor som dragprovstaven är infästade i). Alla variationer av Jog-funktionen fungerar på följande sätt:

1. Ladda den tabell som används till positions mode.
2. Läs av motorns position.
3. Öka/minska motorn position beroende på vilken Jog-funktion som används.
4. Skriva det nya värdet till BIVECTOR
5. Starta motorn med den tabellen som Jog-funktionen använder

6.1.2 Configure

Till programmet för dragprovning finns det ett färdigt program som läser in värden från de tre givarna som är kopplade till dragprovsutrustningen. Configure-fliken är huvudsakligen till för inställningar till datainsamlingen samt sätta last- och förskjutningsgränser.

Till dragprovet finns det tre stycken givare som tidigare nämnts, två stycken är positionsgivare och den tredje är en lastcell. De två positionsgivarna har mätområden på 5 mm respektive 50 mm och används för att mäta förskjutningen som skett vid belastning av provet. Lastcellen är till för att läsa av kraften som provet utsätts för. Lastcellen klarar maximalt en last på 10 kN.

Det sätts last- och förskjutningsgränser och när en av de gränserna nås stoppas motorn. Det finns endast förskjutningsgräns satt för den större positionsgivaren då den mindre positionsgivaren endast används till att åstadkomma en större noggrannhet i mätningen. Då någon av gränserna nås stoppas motorn, detta körs parallellt med programmet. Det spelar inte någon roll vilken flik programmet är i, om en gräns nås stoppas motorn. Stoppet sker i tre steg enligt följande:

1. Kommando för att sätta hastigheten till 0 skickas till BIVECTOR.
2. En fördröjning på 150 ms för att säkergöra att motorn har hunnit sätta hastigheten till 0.
3. Kommando för att stoppa motorn skickas.

Anledningen till att det sker på detta sätt är att om hastigheten inte är noll då kommandot för stoppa motorn skickas bromsar motorn onödigt hårt. Detta leder till onödigt slitage, denna typ av stopp kallas ”soft stop”.

I Configure-fliken finns det även möjlighet att ändra samplingshastigheten som programmet använder då den läser av insignalerna från givarna samt att plotta en av insignalerna i en graf.

6.1.3 Test mode

I test mode fliken sker de sista inställningarna innan ett test kan startas. Det som kan utföras i fliken är att ställa in den hastighet som motorn ska köra i, spara de inställningar som är gjorda i de tidigare flikarna, samt i aktuell flik, och läsa av vilken position motorn befinner sig i.

Som tidigare nämnts i kapitel 2.1 är deformationshastigheten dubbelt så stor som $\mathrm{varv/min}$ är. Det värde som skickas till BIVECTOR är dock inte detsamma. Den omvandling som sker görs enligt följande:

$$\frac{\text{maxvärde för en 16bitars "signed integer"}}{\text{max motorhastighet}} = \frac{2^{15}-1}{7324{,}2} = \frac{32767}{7324{,}2} = 4{,}47 = \quad (1)$$
$$= \text{Värdet som ska skickas för att rotera motorn i 1 rpm.}$$

Siffran 32767 står för maxvärdet på en ”signed integer” och 7324,2 är den maxhastighet som motorn kan köra i. För att maxvärdet på ”integern” skall motsvara maxhastigheten på motorn är det gjort på detta sätt i BIVECTOR. BIVECTOR tar endast emot heltalsvärden vilket gör att motorn inte kan köras i exakt $1\ \mathrm{rpm}$, den minsta hastighet som motorn kan köras i är $0{,}22\ \mathrm{rpm}$.

På grund av att motorn är kopplad till en växel med utväxlingen 8:1 så kan motorn köras mycket långsammare, den nya beräkningen för vad som ska skickas till BIVECTOR är enligt följande:

$$\frac{32767}{7324{,}2} * 8 = 35{,}8 = \textit{Värde som ska skickas för köra i } 1\ \mathrm{rpm} \quad (2)$$

Detta gör att den nya minsta hastighet som motorn kan köras i är $0{,}028\ \mathrm{rpm}$ och den lägsta deformationshastigheten $0{,}056\ \mathrm{mm/min}$. Motorns nya maxhastighet blir $915{,}5\ \mathrm{rpm}$ men då dragprovning sker vid en mycket låg deformationshastighet är detta ingen nackdel.

Alla parametrarar som har ändrats kan sparas till BIVECTOR så att den har kvar dessa inställningar nästa gång som den startar upp. De enda parametrar som kan ha blivit ändrade i då motorn varit avstängd är vridmomentet som ändrades i settings-fliken samt hastigheten som ändrats i speed mode fliken. Det som händer är att BIVECTOR flyttar all data den har i sitt RAM-minne in till sin hårddisk. På samma sätt när en tabell laddas öppnar BIVECTOR aktuell tabell från sin flashminne och flyttar den till sitt RAM-minne.

Test mode fliken har även funktionen för att starta motorn. Funktionen fungerar på det sätt att den startar med den tabellen som är angiven, i detta fall är tabell 25 angiven, BIVECTOR har alla parametrar redan sparade i sitt internminne så det är inte någon väntetid för motorn att starta.

6.1.4 Visning av givardata

Vid sidan av de olika flikarna finns ett statiskt visningsfönster. I detta visningsfönster tas alla värden från givarna in och plottas upp i två grafer. Den första grafen visar ”Load – Position 50mm” och den andra grafen visar ”Load – Position 5mm”.

De värden som tas in från givarna är inte absolutvärden, utan värden som tas in kan nollställs, vilket gör att givaren får ett nytt nollvärde. Detta är till för att göra det enklare att hålla koll på de förskjutningar som skett sen ett test startades. Värden från givarna uppdateras två gånger i sekunden för att göra det lätt att se vad värdet ligger på. Det är de nollade värdena som visas i graferna i visningsfönstret.

Det finns en möjlighet att spara givarvärden till ett textdokument. De värden som sparas till textdokumentet är de nollade värdena. Frekvensen på hur ofta ett värde sparas är samma frekvens som samplingsfrekvensen på insignalerna från givarna.

6.1.5 Reglerad styrning

En begränsning som finns vid reglerad och återkopplad styrning är att kommandon endast kan skickas med en viss hastighet. Som tidigare förklarats behövs en fördröjning mellan varje tecken av ett kommando då detta skickas. Detta ger en viss begränsning i den maxhastighet som reglering kan ske med. Reglering sker genom att hastigheten på motorn ändras. Regleringen kommer alltid att ha en dödtid som är lika med den tid det tar att skicka kommandot. På grund av att motorn är en elmotor bortses från tiden för ändring av hastighet efter det att kommandot mottagits. Den totala tiden det tar att skicka ett kommando blir 110-130 ms.

Det reglerprogram som är uppbyggt för tillfället är ett program där ärvärdet ökar med en konstant hastighet. Detta fungerar som en rampfunktion. Värdet från rampfunktionen jämförs med lastcellens värde och efter reglering skickas en styrsignal som styr motorhastigheten.

Programmet är i en stadie där det är möjligt att testa om det fungerar. Ett tidigare test där en insignal från lastcellen simulerats har gjorts där regleringen visade sig fungerade.

6.2 Provning av lås

Programmet för provning av lås är gjort för att vara lätt att använda och endast de nödvändiga inställningarna och parametrarna kan ändras. Programmet cyklar motorn mellan två positioner och är indelat i två flikar där den enda är optimerad för en bestämd rörelse och den andra fliken är mer allmän.

6.2.1 Nyckelvridning

Nyckelvridningsfliken är en flik där inställningar gällande hur snabbt motorn ska rotera och hur långt den ska rotera är förinställt. Det som går att ställas in är hur många gånger en cykel ska ske samt maximalt positivt och negativt vridmoment.

Programmet används för uthållighetsprovning av lås. Varje gång programmet går genom exekverar cykeln sker följande:

1. Skicka kommando för att motorn ska roterar till position ¼ varv.
2. Liten fördröjning för att motorn ska hinna rotera till det angivna läget.
3. Skicka kommando för att motorn ska rotera tillbaka till sitt ”hemma-läge” vilket är position 0.
4. Liten fördröjning för att motorn ska hinna rotera till det satta läget samt räkna upp ”Number of completed cycles”.
5. Jämför ”Number of cycles completed” med ”Number of cycles”, om de är lika eller större så lämnas loopen. Annars börja om från steg 1.

Målet var att optimera cykeln och den tid som varje cykel tog. Det finns tre faktorer som påverkar hur snabbt den kan gå.

- Fördröjningen mellan varje tecken.
- Fördröjningen mellan varje kommando.
- De fysiska egenskaperna som motorn har.

För att minimera fördröjningen mellan varje tecken utfördes ett test där två kommandon med olika längd skickades och tiden mellan varje tecken successivt minskades. Testet visade att den minsta fördröjningen är 11-15 ms. Om svaret, på kommandot till BIVECTOR, används behövs en fördröjning på 45-50 ms för att svaret ska bli stabilt. Om svaret inte används kan en kortare fördröjning användas.

Den totala tiden som det tar att skicka ett kommando plus den fördröjning som finns mellan varje kommando måste vara större än den tid det tar för motorn att rotera ¼ varv. Av detta fås den fördröjning som behövs mellan varje kommando.

När ett test utfördes, där fördröjningen mellan varje tecken var 30 ms samt fördröjningen mellan varje kommando var 15 ms och de fysiska inställningarna angivna så att motorn hann rotera ¼ varv, angavs det av uppgiftsledaren att testet inte behövde gå snabbare. Det bedömdes att de inställningar som var gjorda skulle behållas då det är mer stabilt att ha en större fördröjning mellan varje tecken än att ha en större fördröjning mellan kommandona. Inställningarna gav att varje cykel tog ca 770 ms vilket get en frekvens på 1,3 Hz.

Cykelvridningsfliken använder sig av en funktion som heter ”homing”. Vad homing gör är att den sätter den aktuella positionen till nolläge. Denna funktion har en nackdel och det är att den behöver utföras i tre steg där ett av stegen kräver att en signal skickas på en digital ingång som är kopplad till en funktion som kallas ”home input high”. Stegen som krävs är följande:

1. Den fysiska switchen aktiveras som startar ”home input high” funktionen.
2. Kommandot för homing-funktionen skickas till BIVECTOR.
3. Den fysisk switchen sätts i off -läge.

6.2.2 Position mode

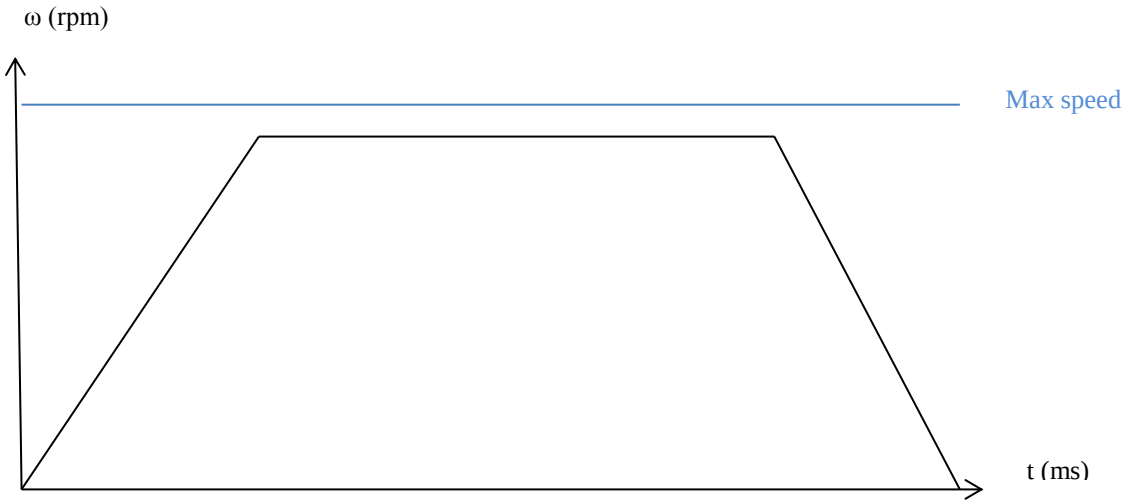
Positionsmode fliken fungerar till stor del som nyckevridningsfilken men har ett antal extra inställningsmöjligheter. Det kan anges hur snabbt motorn skall rotera samt mellan vilka två gränser som den ska gå.

De gränser som anges är, en startposition och en slutposition angivna i grader. Cykeln fungerar på samma sätt som den för nyckelvridning (kapitel 6.2.1) men med några små ändringar. De ändringar som är gjorda är att fördröjningen mellan de två kommandona har ökats för att motorn alltid ska hinna till nästa position innan nästa kommando skickas.

Positionerna den roterar mellan är angivna i grader men motorn tar endast emot signaler i form av antal varv och delar av varv. Kommandosträngen ser ut som: ”*&FF9EAAAAbbbb=”* där ”AAAA” står för antal hela varv och ”bbbb” står för delar av ett varv. Motorn har en växel inkopplad som har utväxlingen 8:1 vilket ger att, då motorn roterar ett varv motsvarar detta 45 verkliga grader. Beräkningen av ”AAAA” och ”bbbb” sker enligt följande:

- $\text{"AAAA"} = \frac{\text{Angivna grader}}{45}$ (3)
- $\text{"bbbb"} = \frac{\text{Angivna grader \% 45}}{\frac{1}{16^4}}$ (4)

I denna flik finns tre inställningar för motorns maxhastighet samt för hur snabbt den ökar sin hastighet. De tre parametrarna kallas ”Position proportional gain”, ”Max positive speed” och ”Max negative speed”



Figur 5. Hastighetskurva i positions mode.

”Proportional gain” är ett mått på hur snabbt motorn ställer in sig till önskat värde. Detta motsvarar lutningen på ramperna enl. fig. 3.X. Om lutningen på rampen är för brant bildas en översväng innan den stabiliseras vid en position.

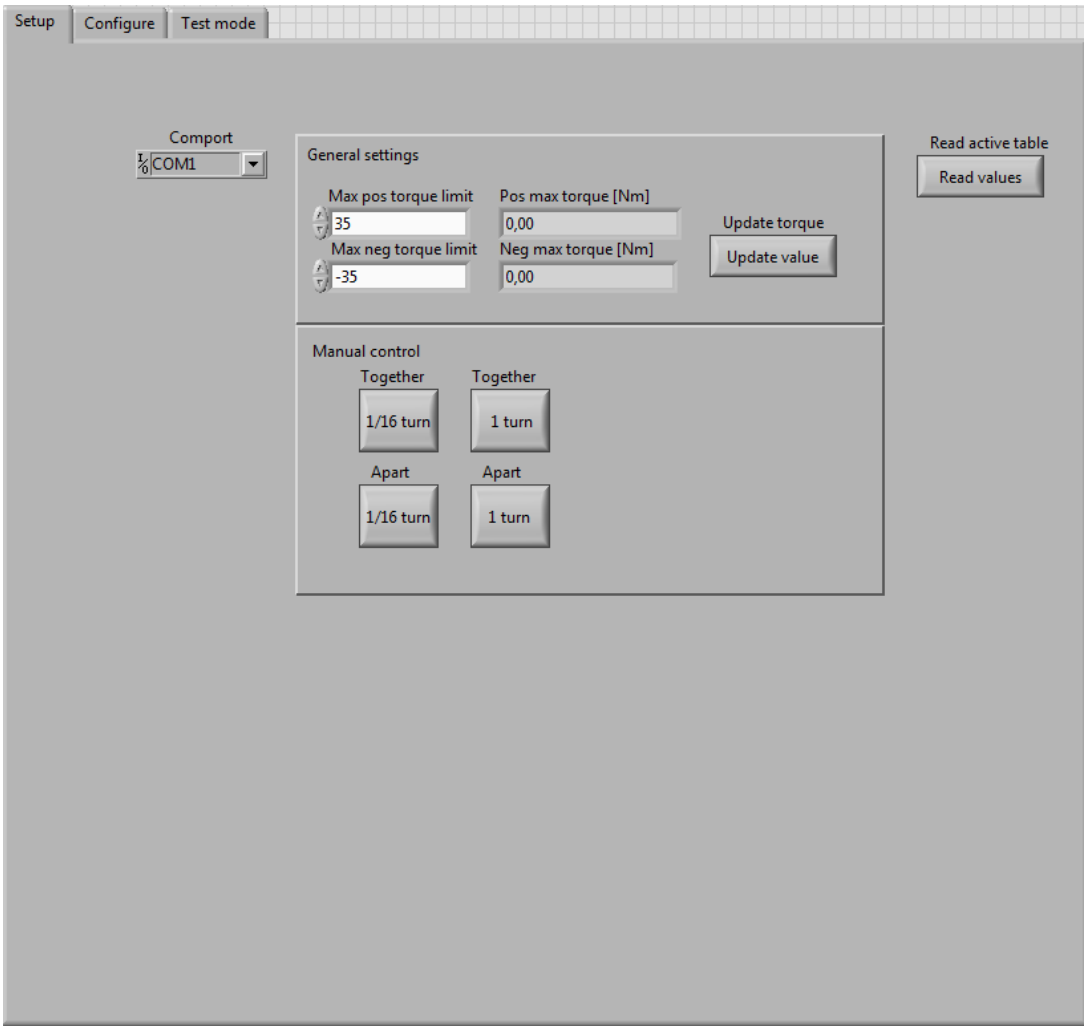
”Max pos speed” och ”max neg speed” är de två gränser som motorn har på hastighet. När motorn körs i positionsläge så fungerar den på så sätt att den höjer sin hastighet successivt tills den når sin maxhastighet och när den närmar sig sin slutgiltiga position så sänker den hastigheten lika snabbt. Gränserna är till för att begränsa hastigheten så att motorn inte roterar för snabbt för de objekt som provas.

7 Användningen av program

Följande kapitel beskriver hur en typisk körning går till, från att programmen startas upp till att provningen avslutas. Det visar även hur programmen och dess GUI är utformade.

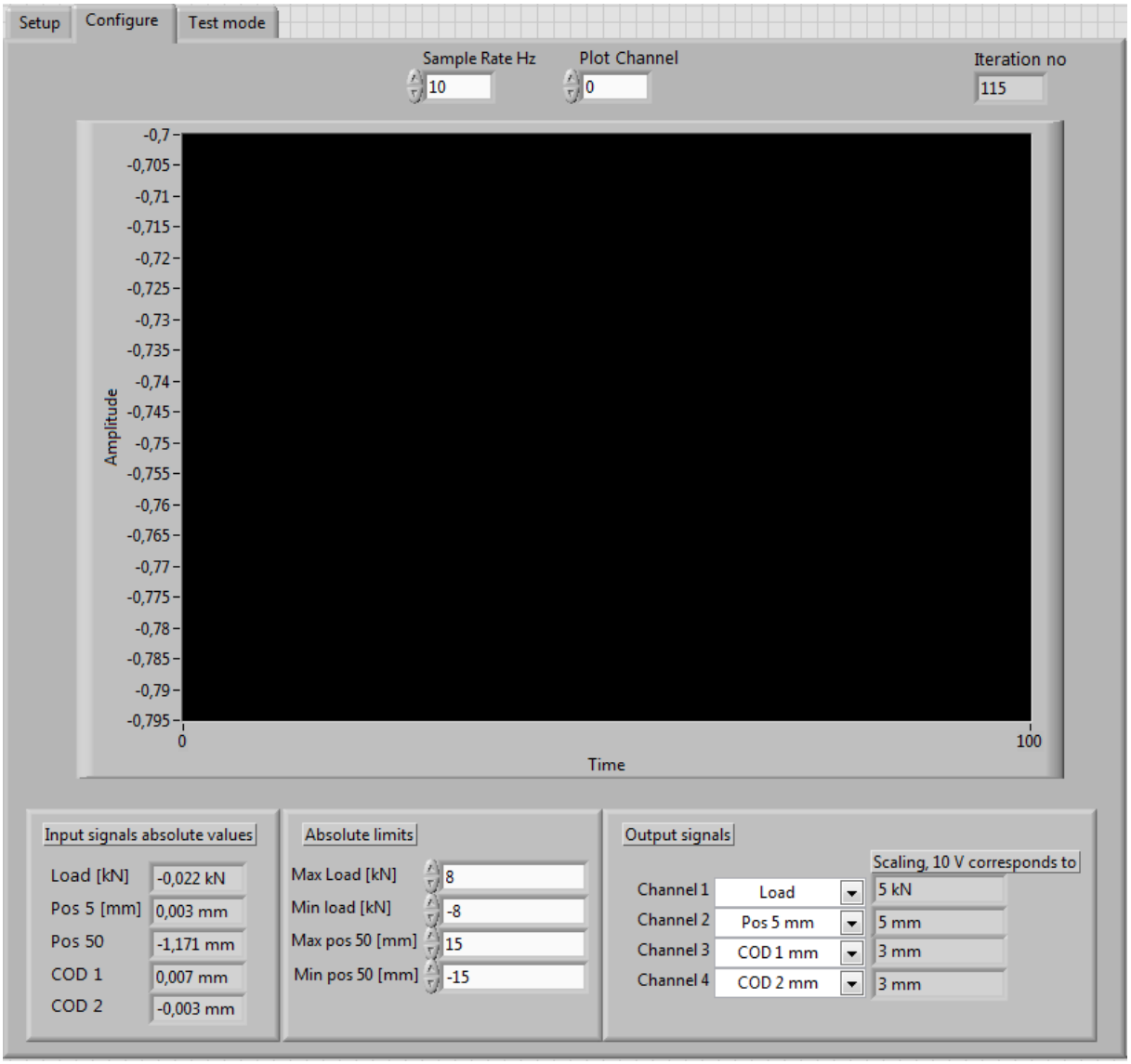
7.1 Dragprovningen

Det första steget är att starta upp programmet. Programmet öppnar automatiskt setup-fliken vid uppstart.



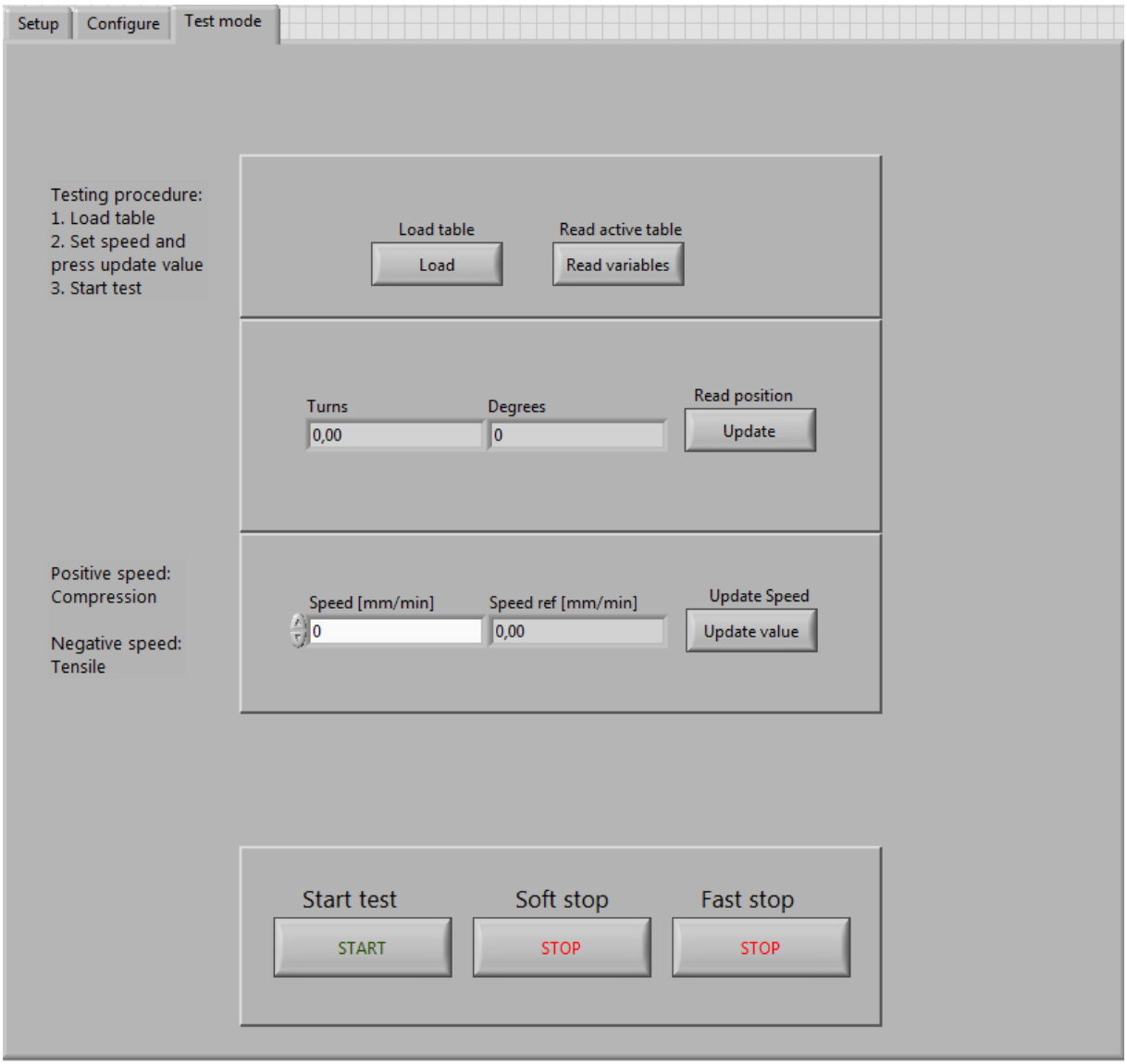
Figur 6. Grafiska gränssnittet för Setup fliken.

1. Val av com-port sker för att kommunikationen skall fungera.
2. Materialet som ska provas sätts in i dragprovningsutrustningen. Om provstaven inte får plats eller att gapet mellan de två block som ska hålla provet är för stort används antingen ”Together”- eller ”Apart”-knapparna för att flytta plattorna närmare/längre ifrån varandra.
3. Inställning av max- och minvridmoment görs och knappen ”Update torque” aktiveras (läs av värdet i rutorna för att säkerställa att BIVECTOR har uppdaterat värdena).
4. Öppna ”Configure”-fliken genom att klicka på fliken.



Figur 7. Grafiska interfacet för Configure fliken.

5. I rutan ”Absolute limits” i Fig.7 anges de absoluta givargränserna. När en av dessa gränser nås stoppas motorn.
6. Inställning av samplingstiden. Detta ställer även in hur ofta ett nytt värde skrivs till den fil som sparas vid körning.
7. Öppning av test mode fliken.

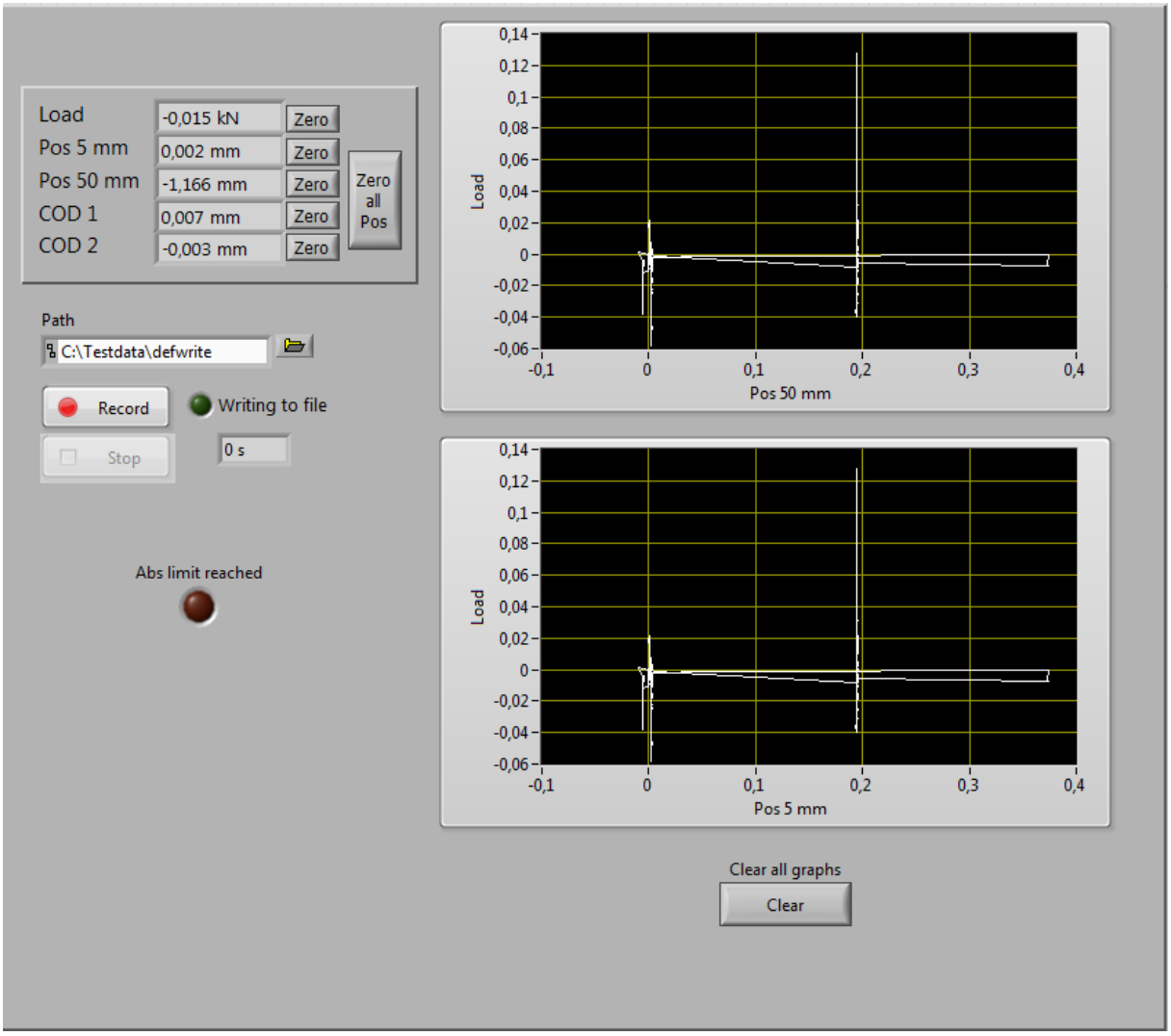


Figur 8. Grafiska interfacen av Test mode flik.

8. Load table knappen aktiveras för att hämta in den tabell som används till test.
9. Deformationshastigheten i drag eller tryck ställs in. Positiv hastighet anges vid tryckprovning och en negativ hastighet anges vid dragprovning. Knappen ”Update Speed” aktiveras, värdet läses av för att säkerställa att BIVECTOR har mottagit rätt värde.
10. Alla inställningar är nu gjorda och ”Start test” kan aktiveras för att starta motorn.

Motorn stannar då en av gränserna nås eller då en av de två stoppknapparna aktiveras.

Under körning är det mest relevant att visa den statiska fliken.



Figur 9. Grafiska gränssnittet för den statiska fliken.

I denna flik visas data från givare plottade i två grafer där den över grafen i Fig. 9 visar lasten på Y-axeln och förskjutningen av 50 mm positionsgivaren på X-axeln. När ett prov är monterat rekommenderas det att nollställa alla värden på givarna genom att aktivera ”Zero all Pos”-knappen.

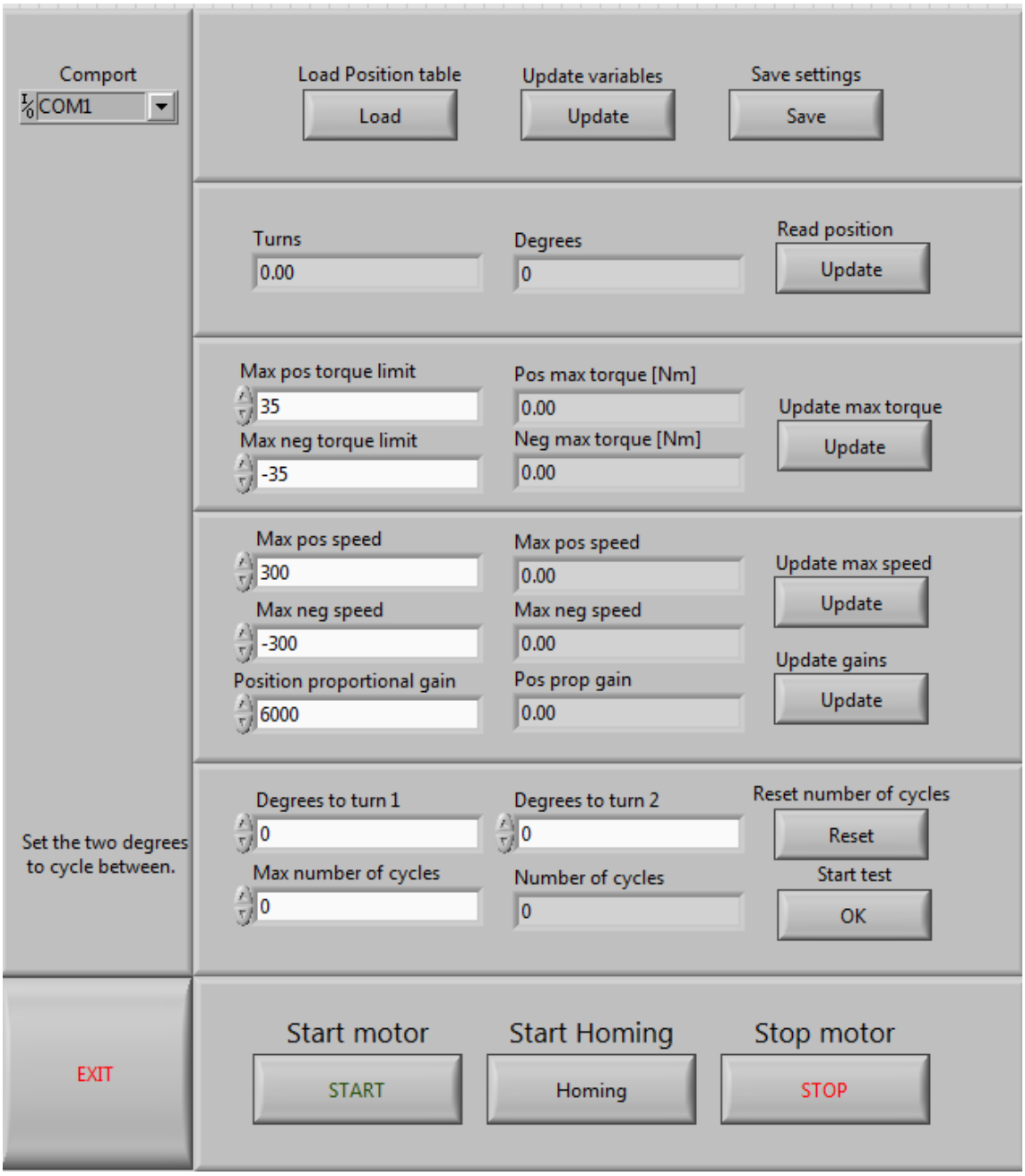
De värden som visas i graferna kan även sparas till en textfil. Skrivningen till fil startar genom att aktivera knappen ”Record”. Mapp och filnamn för filen som ska sparas anges genom att trycka på mappikonen som visas vid ”Path”.

7.2 Provning av lås

Det finns två olika varianter av provning av lås, position mode där det är möjligt att ändra på alla parametrar som har en inverkan på provningen. Den andra är nyckelvridning vilket är en optimerad variant av position mode för att endast gå mellan positionerna 0 och högst 90 grader.

Position mode

Följande kapitel ger en förklaring till hur en typisk körning går till vid användning av position mode.

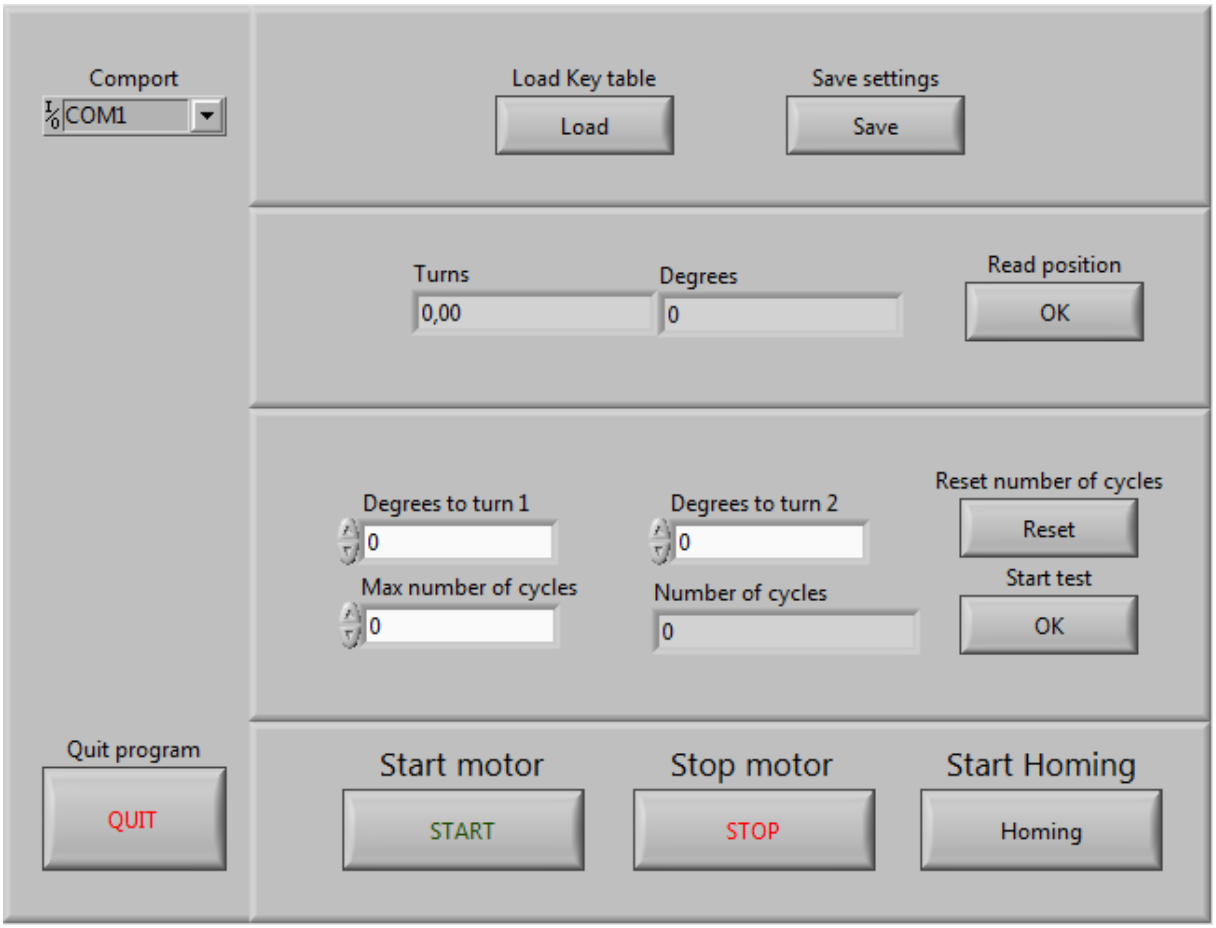


Figur 10. Position mode flik.

1. Tabell för positions table laddas in genom ”Load Position table” aktiveras.
2. Inställning för max och min vridmoment görs och skickas till BIVECTOR via knappen ”Update max torque”, värdet läses av från BIVECTOR för att säkerställ att värdet har ändrats.
3. Inställning av min- och maxhastighet[rpm] görs och skickas till BIVECTOR via knappen ”Update max speed”, värdet läses av från BIVECTOR för att säkerställ att värdet har ändrats.
4. Inställning av ”Position proportional gain”, värdet skickas till BIVECTOR med knappen ”Update gains”, värdet läses av från BIVECTOR för att säkerställa att värdet har skickats.
5. Inställning av vilka två positioner som motorn skall rotera emellan samt inställning av antal cykler motorn skall utföra före den stannar.
6. Låset sätts in så att det är i startposition
7. Positionen på motorn sätts till 0 genom att aktivera knappen ”Start Homing”.
8. ”Read position” aktiveras för att säkerställa att motorn är på position 0.
9. Motorn startas med knappen ”Start motor”, efter motorn är startad kan testet startas genom att knappen ”start test” aktiveras.

Nyckelvridning

Följande kapitel ger en beskrivning av hur en typisk körning går till då fliken ”nyckelvridning” används.



Figur 11. Nyckelvridningsflik.

1. Nyckelvridningstabellen laddas genom att ”Load key table” aktiveras.
2. Inställning av vilka positioner motorn skall gå emellan (optimerad för <90 grader).
3. Låset sätts in så att det är i startposition
4. Positionen på motorn sätts till 0 genom att aktivera knappen ”Start Homing”.
5. ”Read position” aktiveras för att säkerställa att motorn är på position 0.
6. Motorn startas med knappen ”Start motor”, efter motorn är startad kan testet startas genom att knappen ”start test” aktiveras.

8 Resultat

Resultatet är två program där det ena programmet styr dragprovningsutrustningen och det andra programmet styr utrustningen för provning av lås. Test av båda programmen har utförts för att säkerställa att de fungerar som planerat.

Test av programmet för dragprovningsutrustningen utfördes i sammarbete med David Samuelsson, testet delades in i ett antal steg för att testa programmets funktionalitet. Följande funktioner testades:

1. **Undersöka om kommunikationen fungerade mellan datorn och de olika enheterna som var kopplad till den.**

   För att undersöka om värden togs in från givarna samt om de var korrekta gavs en belastades på givarna. Värdet på givarna lästes av i programmet och jämfördes med den belastning som lagts. Värdena stämde översens vilket betydde att denna del av programmet fungerade.

   För att undersöka om kommunikationen fungerade mellan datorn och styrenheten skickades kommandot för att starta motorn. BIVECTOR tog emot kommandot och motorn startade, kommunikationen fungerade.

2. **Undersöka om ”jog” funktionen fungerade korrekt.**

   För att undersöka om ”jog” funktionen fungerade korrekt startades motorn och kommandot för de olika varianterna skickades. Det antal varv som motorn roterade stämde överens med det värdet som skickats in. Det konstaterades att omvandlingen mellan programmet och BIVECTOR var korrekt samt att ”jog” funktionen fungerade korrekt.

3. **Undersöka om parameterändringar fungerade samt att omvandlingen mellan programmet och BIVECTOR var korrekta.**

   Ändring av vridmoment och hastighetsparametrar testades. Den önskade hastigheten sattes till 1 rpm varpå motorhastigheten fastställdes vara 1 rpm. Ökning av vridmoment ledde till att lastcellens värde ökade. Parameterändringar fungerade samt omvandlingen av hastigheten var korrekt.

4. **Undersökning av säkerhetsfunktioner.**

   Den fysiska brytaren samt de säkerhetsfunktioner som finns i programmet testades. När brytaren påverkades stannade motorn. När något värde på givarna översteg säkerhetsgränserna som ställts in i programmet stoppades motorn.

Testen visade att programmet fungerade som det skulle och att omvandlingarna var korrekta.

Programmet för styrning av lås testades genom följande steg:

1. **Undersöka om kommunikationen fungerade mellan datorn och BIVECTOR.**

   För att undersöka om kommunikationen fungerade skickades kommandot för att starta motorn. Motorn startade, detta visade att kommunikationen fungerade.

2. **Undersöka om ändring av parametervärden var möjliga.**

   Värden på parametrarna för maxhastighet, hasighetsökning och vridmoment ändrades. Ändringarna på parametrarna ledde till ändringar på motorns hastighet och vridmoment.

3. **Test av cykling mellan två positioner, samt att den gick till korrekt position.**

   Motorn startades och cyklingen startade. En låg motorhastighet sattes för att det skulle vara lätt att se om motorn gick till rätt position. Motorn roterade till rätt positioner samt stannade efter att angivet antal cykler upnåtts.

Programmet för provning av lås fungerade som det skulle och inget problem uppkom under testets gång.

9 Diskussion

I följande kapitel diskuteras det om målsättningen har uppnåtts och om möjliga framtida arbeten.

9.1 Uppföljning av syfte och målsättning

Stora delar av syftet och frågeställningarna har uppfyllts men inte allt. Resultatet av projektet är ett fungerande program som styr motorn mot en konstant deformationshastighet. I programmet tas det även in information från givare vilka visas i form av grafer och kan välja att spara mätdata till en fil. Det grafiska interfacet är välkonstruerat och är uppbyggt med tanken att det ska vara lätt att bygga vidare på.

Den del som saknas i programmet är att motorn inte kan köras laststyrt eller förskjutningsstyrt. Ett testprogram finns i nuläget som testar laststyrd körning men detta hann inte färdigställas fullt ut, dock är programmet färdigt och funktionaliteten tros finnas.

9.2 Framtida arbete

Vi anser att det finns utvecklingsmöjligheter både när det gäller nyckelprovning och reglerad mekanisk provning.

När det gäller nyckelprovning och liknande provning där motorn cyklar mellan ett antal olika positioner finns det en möjlighet att bygga vidare på programmet, tidigare har en cylinder kopplats till motorn som gått ut och in varefter att varje position uppnåtts. Detta finns inte längre då det var väldigt krångligt och inte fungerade helt. En möjlighet är då att bygga in denna funktion i nuvarande program då BIVECTOR har en inbyggd funktion vilken skickar ut en signal på en digital utgång då en specificerad position uppnåtts.

Vi tror att det är möjligt att köra motorn mot en konstant last- eller förskjutningsökning. Ett program för att utföra detta är utvecklat men det fanns inte tid för att testa programmet. Nackdelen vid reglerad körning, som tidigare nämnts, är att det alltid kommer att finnas en dödtid på grund av att kommunikationen mellan datorn och BIVECTOR är långsam.

Referenser

[1] Lindgren, P. och Östergren A. (2010) *Konstruktion av dragsteg*. Göteborg: Chalmers tekniska högskola. (Examensarbete inom Institutionen för Tillämpad mekanik).

[2] Eitel, E. (2011) The difference between AC induction, permanent magnet, and servomotor technologies. *Machine design.* *http://machinedesign.com/motorsdrives/difference-between-ac-induction-permanent-magnet-and-servomotor-technologies* (2014-05-19).

[3] ABB, DRIVE SYSTEMS with 8C SERIES Brushless Servomotors and 500 Series BIVECTOR Converters. Ref. MANIU10.9906 GB (1999). http://www.lejonflagg.se/abb/manabbservo.pdf (2014-04-29).

[4] Buchanan,W. (2000) Computer busses. Butterworth Heinemann.

[5] ABB, DRIVE SYSTEMS with 8C SERIES Brushless Servomotors and 300 & 500 BIVECTOR Converters. Ref. MANSER09.9810 GB (1999). http://www05.abb.com/global/scot/scot234.nsf/veritydisplay/202fbff27919ee0cc1257b130056e6dc/$file/Manser09e.pdf (2014-04-27).

[6] *National Instruments.* http://www.ni.com/LabVIEW/why/multiple-targets/(2014-05-08).

Bilaga A: Manual för Dragprov Styrning

Detta program hanterar körning av ett dragprov samt datainsamling från givare. Manualen beskriver hur programmet används, hur det är uppbyggt och vad sub VI utför.

Hur programmet används

Programmets uppbyggnad är uppdelat i två delar, i den ena delen ställs alla inställningar in före ett test påbörjas samt start och stop av testet. Den andra delen hanterar datainsamling från givare som kan sparas samt visas både i form av grafer och värden.

Programmet har tre stycken flikar och en statisk flik som visas hela tiden, dessa är ”Setup”, ”Configure” och ”Test mode” samt Visning av givardata.

OBS! Inga ändringar behöver göras mellan användning av BivWin och dragprovstyrningen.

Flik ett - Setup

Setup används för allmänna inställningar som gäller för alla flikar.

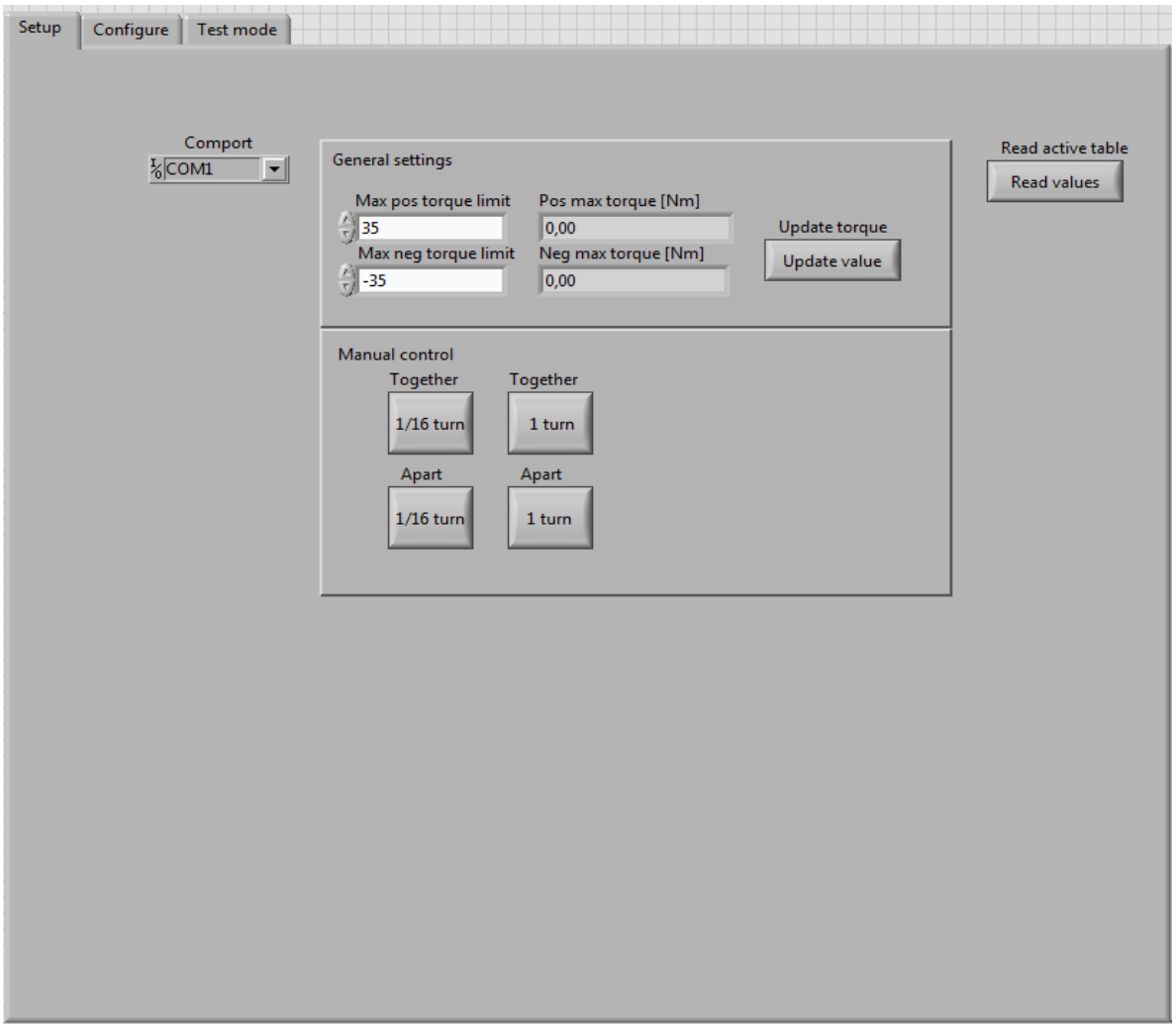


Fig. 1 Grafiska interfacet av ”Setup”.

Det finns fyra olika indelningar i Setup vilka är inställning av COM-port, ”General settings”, Manual control” och ”Read active table”.

För att möjligöra kommunikation med BIVECTOR behövs det ställas vilken COM-port som används mellan datorn och BIVECTOR.

I ”General settings” rutan finns de inställningar som gäller för alla olika test lägen och flikar, i detta fall ändras endast inställningarna för ”test mode” då programmet endast har ett testläge för tillfället. Den enda parametern som hittills kan ändras är motorns vridmoment. När vridmomentet önskas ändras skrivs ett nytt värde in i textrutorna ”Max pos torque limit” och/eller ”Max neg torque limit” varefter knappen ”Update torque” aktiveras, det nya värdet skickas då till BIVECTOR. Efter det nya värdet skickats läses värdet på vridmomenten av från BIVECTOR för att säkerställa att den mottagit värdena, vilket visas i rutorna ”Pos max torque” och ”Neg max torque”.

”Manual control” innehåller funktioner som styr motorn genom att rotera motorn till en viss position. Motorn roterars med- eller moturs ett varv eller en sextonsdel av ett varv. Motorn kan rotera så att dragprovet går isär, ”Apart”, eller ihop, ”Together”. Funktioner aktiveras via respektive knappar enl. Fig.1. ”Apart” funktionen ökar avståndet mellan de två blocken för dragprovet och ”Together” funktionen minskar avståndet mellan blocken.

Flik två - Configure

Configure används för inställningar och hastighet av datainsamling från givarna.

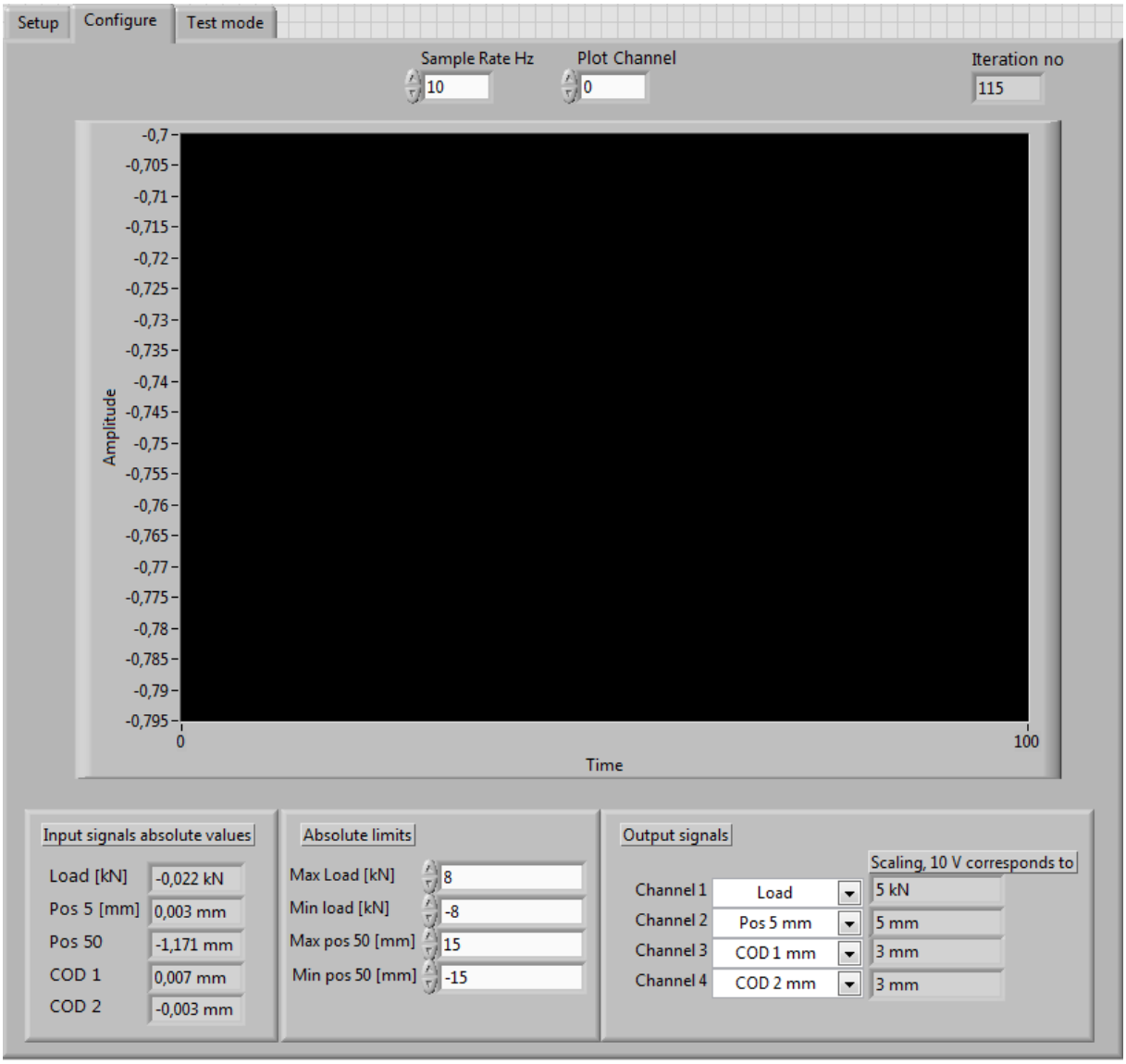


Fig. 2 Grafiska interfacet av ”Configure”.

Vid ett dragprov används tre givare för behandling av krafter och töjningar under dragprovet. Givarna skickar signaler till datorn och visas som värden. För att ställa in hastigheten av samplingstiden skrivs frekvensen in i ”Sample Rate Hz”, denna frekvens bestämmer även datainsamlingshastigheten.

I rutan ”Input signals absolute values” visas givarnas absoluta värden. Alltså det exakta värdet givaren befinner sig på. Det kan ställas in gränser på max/min kraft samt deformation under ”Absolute limits” som stoppar motorn då någon av gränserna överstigs.

Under ”Output signals” väljas vilka värden som skickas vidare till andra enheter, ex: en annan dator.

Flik tre – Test mode

För att utföra ett test används Test mode som hanterar hastigheten hos motorn.

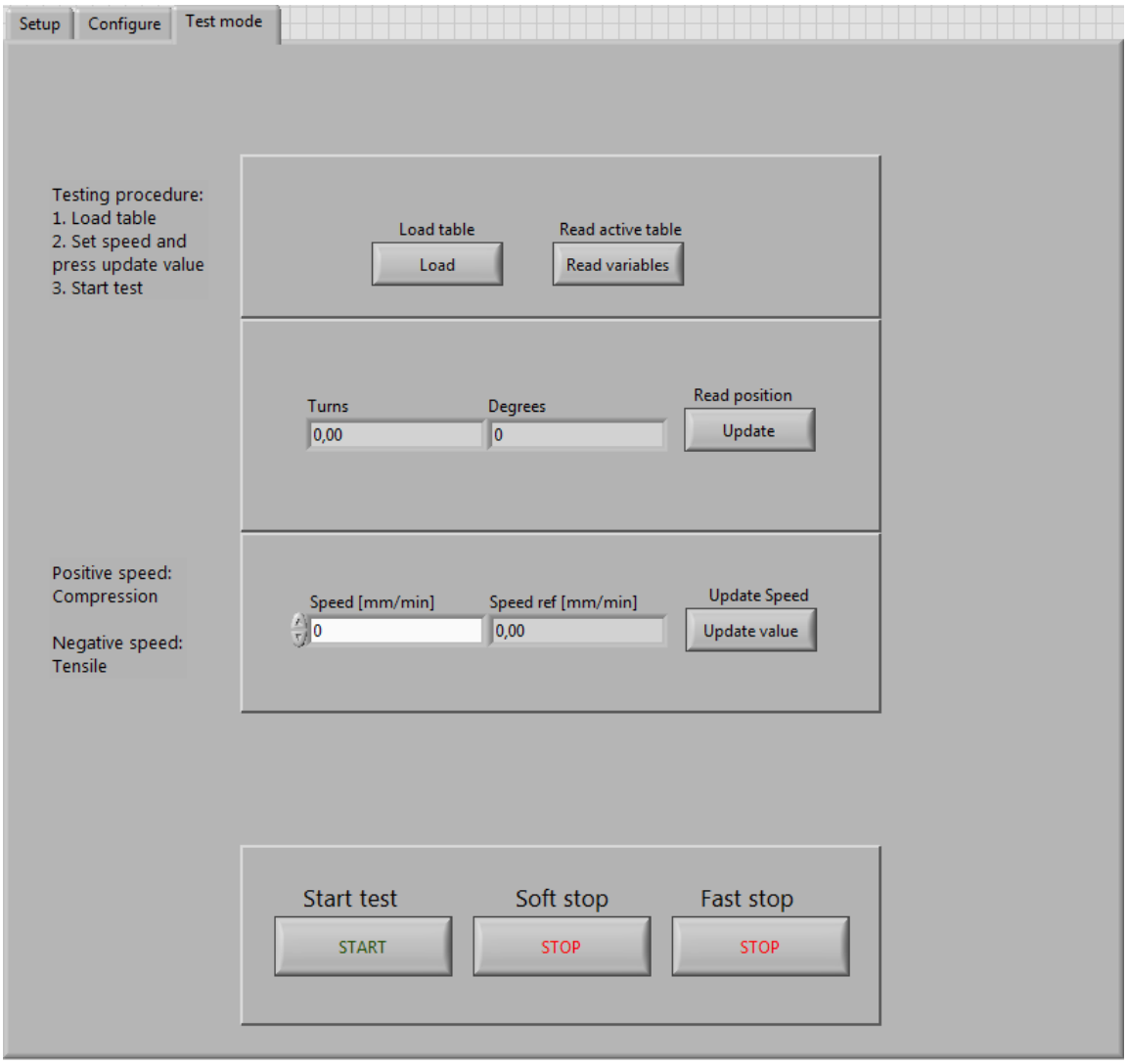


Fig. 3 Grafiska interfacet av ”Test mode”.

När ett test ska utföras rätt behövs inställningarna i Setup och Configure vara inställda först. För att ett test ska bli möjligt behövs inställningarna laddas in från BIVECTOR genom knappen ”Load Table”, det är först då som parametrarna blir ändringsbara.

Innan ett test utförs kan hastigheten läsas av med ”Read active table” för att se den förinställda hastigheten, vilket visas på ”Speed ref”. Om det inte är den önskade hastigheten skrivs det nya värdet in på ”Speed” och uppdateras med ”Update speed”, värdet som visas i ”Speed ref” hämtas från BIVECTOR.

För att starta testet används ”Start test” då skickas en startsignal till BIVECTOR som börjar ge styrsignaler till motorn med de aktiva inställningarna. Om hastigheten önskas ändras under körning behövs det endast skriva in på ”Speed” och sen ”Update speed”.

Det enda problem som kan uppstå är om ”Load”-knappen inte aktiverats och har en av ”Manual control”-funktionerna tidigare används. Det som händer då motorn startas är att den använder sig av den senast använda hastigheten.

För att stoppa motorn finns det två stop, ”Fast stop” och ”Soft stop”, som stoppar motorn på olika sätt. ”Fast stop” stoppar motorn direkt och ”Soft stop” sätter först hastigheten på motorn till noll och stannar därefter motorn.

Statisk flik – Visning av givardata

Detta är en flik som används vid testning och ligger parallellt med flikarna och visas hela tiden.

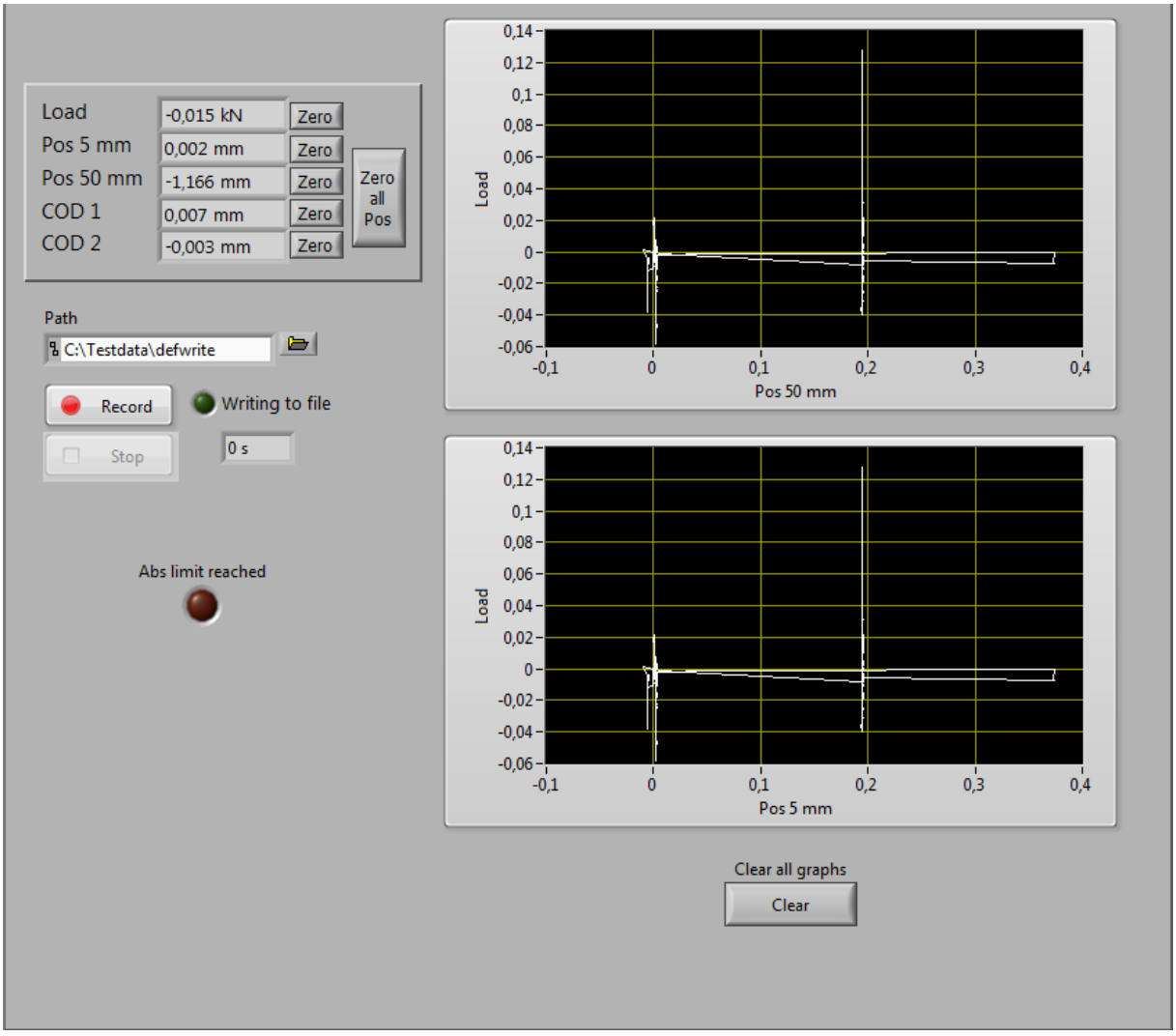


Fig. 4 Grafiska interfacet av ”Visning av givardata”.

Vädena som visas i fliken är inte absolutvärden, de har blivit behandlade för att lättare kunna läsa av förskjutningar och lastökningar som skett sen ett test startades. Värdena från givarna uppdateras två gånger i sekunden för att göra det lätt att se vad värdet ligger på. Det är även de nollade värdena som visas i graferna i visningsfönstret.

Det finns möjlighet att spela in dragprovstestet med ”Record” knappen. När man spelar in data sparas det i mappen som är vald under ”Path” och visas att det sparas då ”Writing to file” är tänd. För att stoppa inspelningen används ”Stop”.

Det finns även en gränsvärdeslampa som tänds om någon av de absoluta gränserna överstigs.

Programmets uppbyggnad

Logiken bakom programmet är uppbyggt i två huvuddelar där den ena delen hanterar datainsamling från givare och den andra delen sköter kommunikationen med BIVECTOR 500, styrenheten till servomotorn. För att delar i programmet ska kunna utföras parallellt och i olika hastigheter används ”While” loopar. Kommunikationen kan ske med användning av RS-232 och RS-485, på grund mer stabilitet används RS-485.

Givardata

Givarna skickar kontinuerligt in analoga signaler till datorn via en dosa gjord av National Instrument som har ett färdiggjort omvandlingsblock i LabView som heter ”DAQ Assistant”, i Fig. 5. Med ”Sample Rate Hz”, Fig. 5, bestäms frekvensen av hur fort avläsningen av givarna sker. Beroende på vad de analoga ingångarna är ställda att omvandlas till sköter ”DAQ Assistant” det automatiskt. Under dragprovet kan last, position och sprickbildning mätas. Från ”DAQ Assistant” skickas de värdena i en seriekoppling till både en graf och ett block som delar upp signalerna. Värden är absoluta och anger givarnas exakta värden, de visas i programmet med indikatorer.

För att få fram justerbara värden används ett block, blocket har som insignal signalernas absolutvärden och dess utsignal är de justerade värdet, som möjliggör nollställlning av det aktuella värdet. Värdena visas i programmet och för att de ska uppdateras i en behaglig hastighet för ögat används loppen enl. Fig. 6 som uppdaterar värdena med ett intervall på 500 ms.

De justerbara signalerna är kopplade emot en inspelningsenhet, i Fig. 5, som sparar data från dragprovet. För att ställa in var filen ska sparas används ”Path” och då det sparas aktiveras en lampa, ”Writing to file”, samtidigt som en tidräknare börjar räkna. För start och stop av datainsamlingen används ”Record Button” och ”Media Stop”.

Det som är till höger i ”while”-loopen i Fig. 5 används för att skicka vidare signalerna till andra enheter via fyra portar. Det kan väljas i programmet vilket värde som ska skickas från vilken port via ”Channel” knapparna.

I Fig. 7 läses de absoluta värdena av och jämnförs med gränsvärderna för vad givarna maximalt får utsättas för innan testet stoppas. Det visas två grafer av de justerbara värdena.

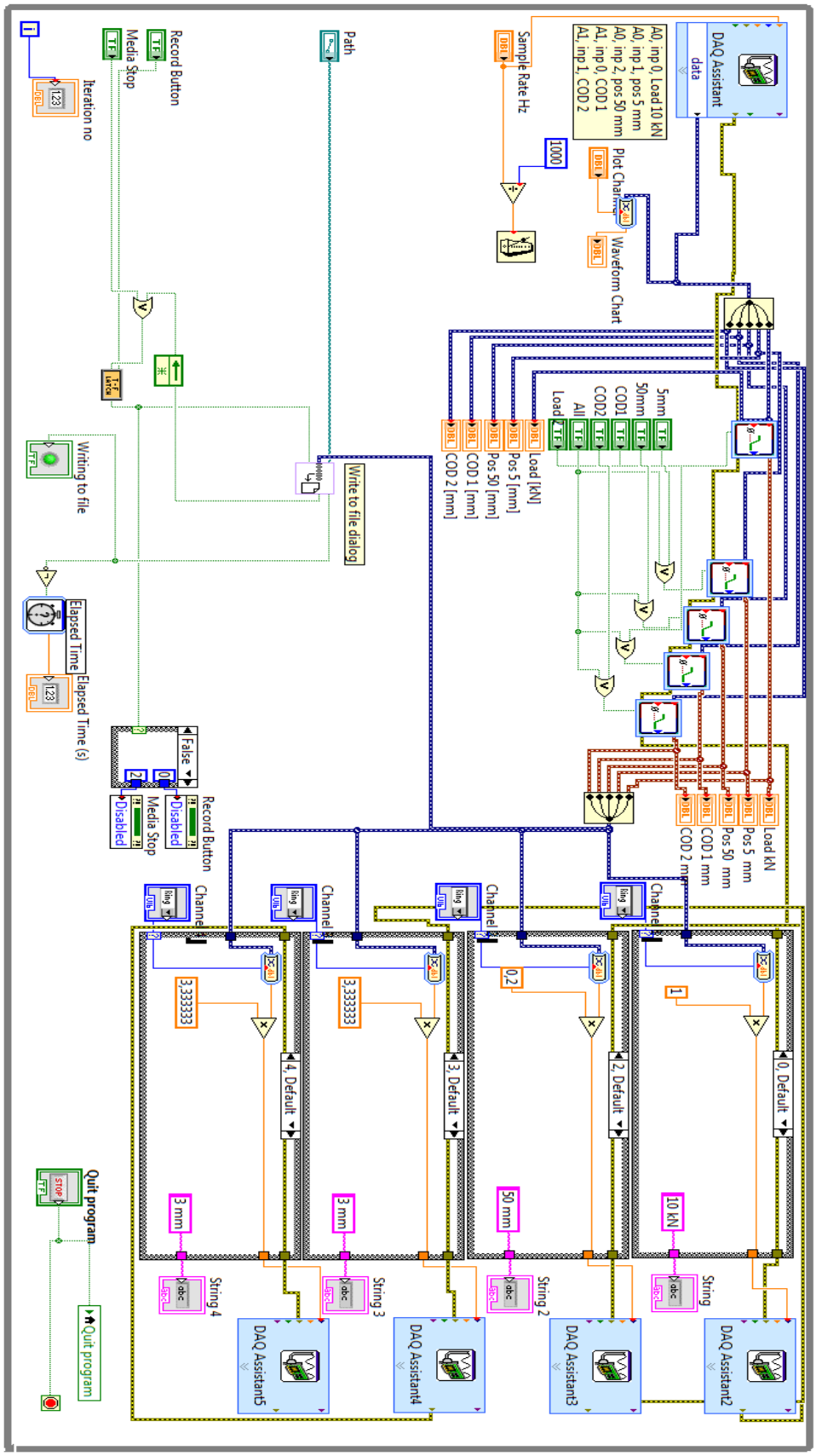


Fig. 5 Givardatainsamling

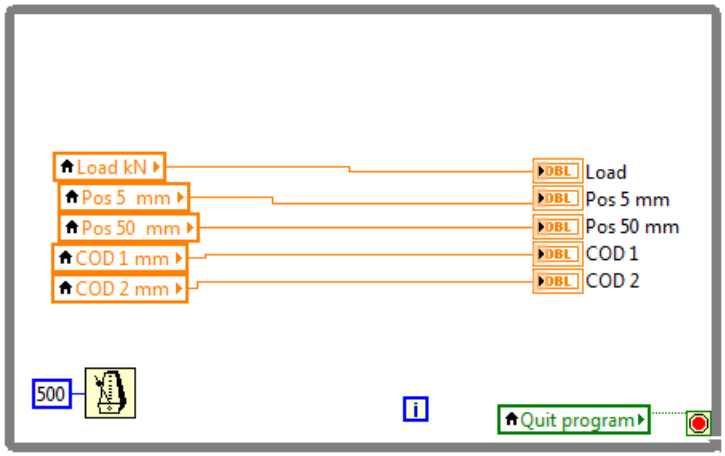


Fig. 6 Uppdatering av visningsvärdena

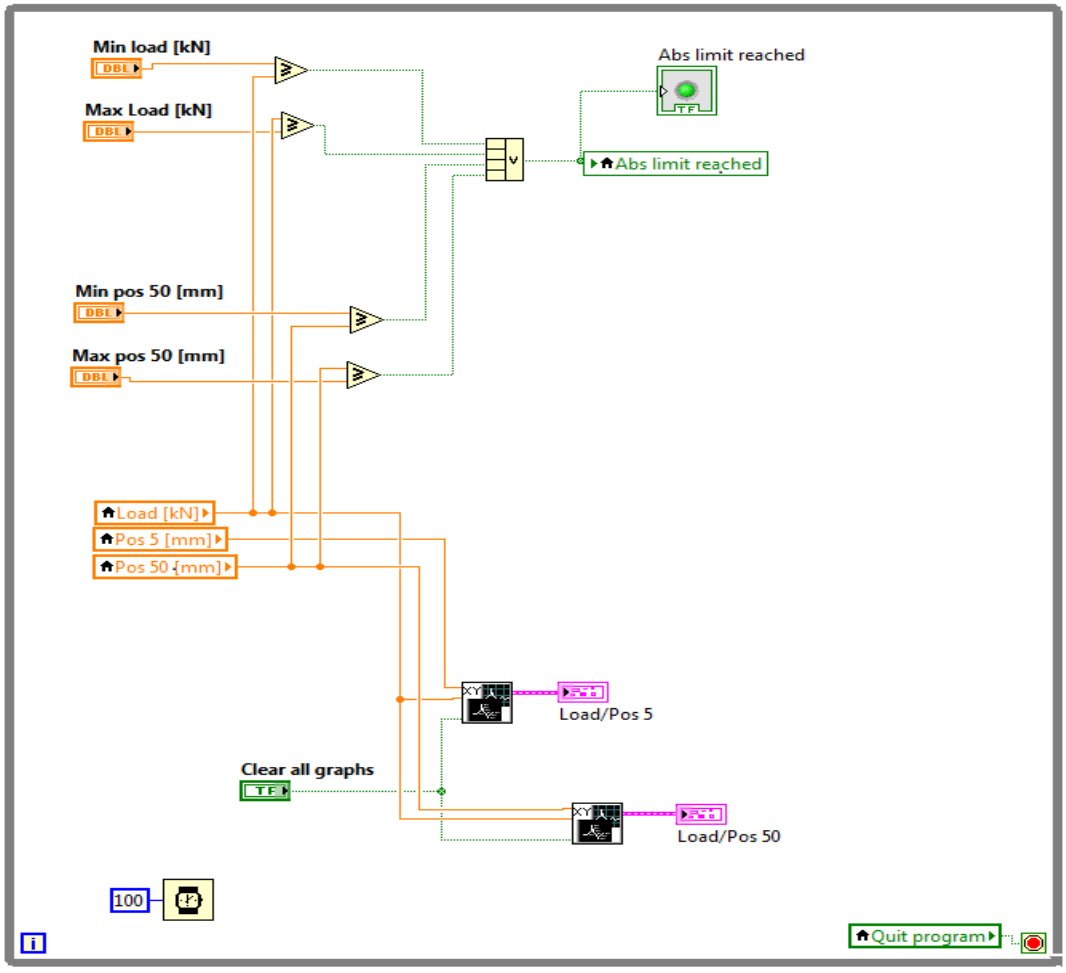


Fig. 7 Uppdatering av absolutvärdesgränserna och visningsgraferna

Kommunikation mot BIVECTIOR 500

Kommunikationen med BIVECTOR sker via en COM-port och för att välja vilken av datorns olika COM-portar som den använder används ”Comport”. För att säkerställa att det är rätt information som kommer fram används ”Even parity”. ”VISA serial” startar kommunikationen.

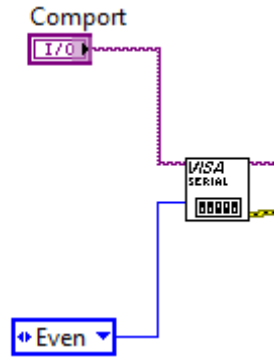


Fig. 8 Upprättning av kommunikation.

För att avsluta kommunikationen med BIVECTOR används ”Visa Close”. Vid användning av programmet används RS-485 men för att BIVECTOR ska kunna anslutas mot BivWin behövs ett RS-232 kommando skickas innan avslutning av kommunikation. Anledningen är att när ett RS-232 eller RS-485 kommando skickas ställs BIVETOR in mot användning av just den standarden, BivWin använder RS-232.

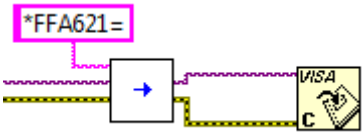


Fig. 9 Nedstängning av kommunikation.

För att programmet ska kunna använda sig av flikar används ett block som heter ”Tab Control”, fig. X. Då en flik är aktiv skickar ”Tab Control” ett villkor, för den specifika fliken, till en ”Case” funktion som utför operationer beroende på villkoret (bara en flik kan vara aktiv åt gången). Då Test mode är aktivt är bara kommandon från ”test mode” samt de parallella ”while” looparna som är tillgängliga.

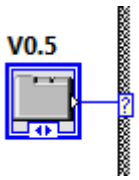


Fig. 10 Flikhanteraren.

Förklaring av vad som händer då blocken i ”Setup” fliken aktiveras, från vänster till höger.

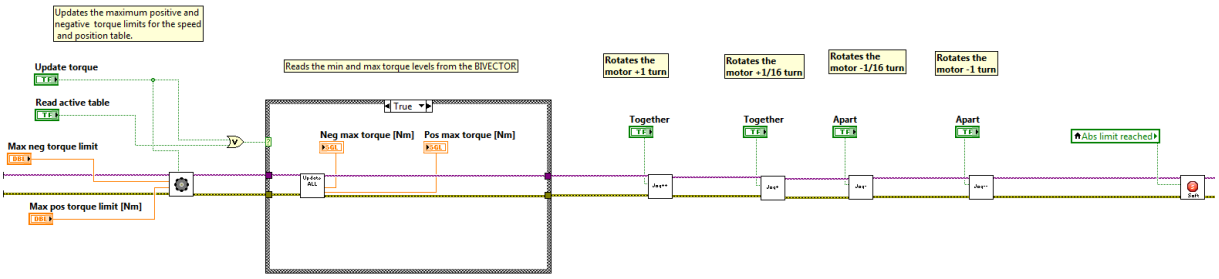


Fig. 11 Uppbyggnaden av ”Setup”-fliken

- Ändrar inställningarna för de maximala positiva och negativa vridmomentsgränserna hos motorn.
- Uppdatera de existerande värdena hos BIVECTOR och visar dom i programmet.
- Roterar motorn ett varv i positiv riktning, plattorna går ihop.
- Roterar motorn en sextondels varv i positiv riktning, plattorna går ihop.
- Roterar motorn ett varv i negativ riktning, plattorna går isär.
- Roterar motorn en sextondels varv i negativ riktning, plattorna går isär.
- Stoppar motorn mjukt om gränsvärdena av givarnas absolutvärden överstigits.

Förklaring av vad som händer då blocken i ”Configure” fliken aktiveras, från vänster till höger.

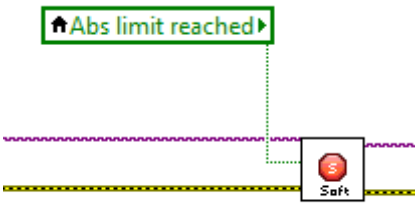


Fig. 12 Uppbyggnaden av ”Configure”-fliken

- Stoppar motorn mjukt om gränsvärdena av givarnas absolutvärden överstigit.

Förklaring av vad som händer då blocken i ”Test mode” fliken aktiveras, från vänster till höger.

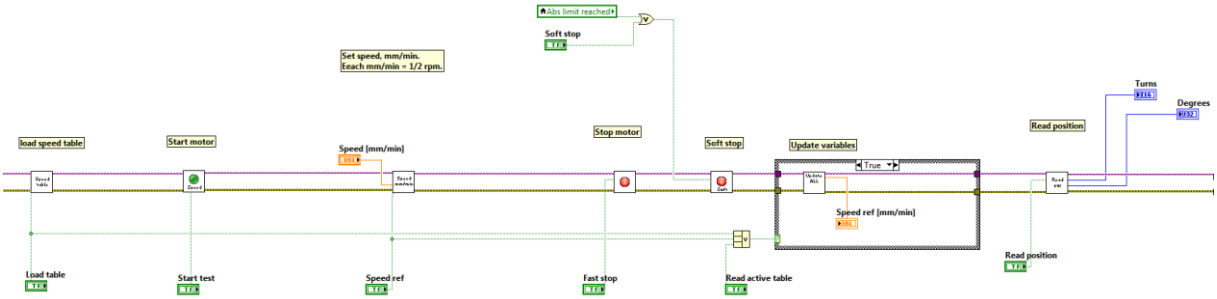


Fig. 13 Uppbyggnaden av ”Test mode”-fliken

- Laddar in tabellinställningarna för körning med hastighet.
- Startar motorn med de förinställda hastighetsinställningarna.
- Ändrar inställningarna för hastigheten på motorn samt uppdarterar dom så de visas i programmet.
- Stoppar motorn.
- Stoppar motorn mjukt om gränsvärdena av givarnas absolutvärden överstigits.
- Läser av de existerande värdena hos BIVECTOR och visar dom i programmet.
- Läser av motorns position och visar den uttryckt i antal varv och grader.

Beskrivning av ”Sub VI”

”Sub VI” används för att få en mer ren layout i programmet så det blir lättare att följa. Nedan beskrivs en detaljerad förklaring om de olika ”Sub VI”. ”VISA” och ”error” blocken i början och i slutet av ”Sub VI” är till för kommunikationen mot BIVECTOR. När en ”Sub VI” aktiverars arbetar den från vänster till höger. Rubriken för varje ”Sub VI” motsvarar filnamnet.

- “String assemble” 

“String assemble” skapar en kommando sträng med angiven specifik kommando kod (x) och värde (y). Kommandovärdet kan vara ett positivt eller negativt decimaltal som omvandlas till ett tecknat hexadecimaltal. Då omvandlingen sker från negativt decimaltal till hexadecimalt tal fylls värdet ut så det blir sexton bytes lång. Kommandovärdet som skickas får max vara fyra bytes så de 4 minst signifikanta bytes från omvandlingen tas ut.
Kommando koden byggs upp enl. mönstret ”&FFxy=” där de olika delarna betyder:
    1. ”&” = användning av RS-485
    2. ”FF” = motoradress (FF används om kommandon ska gälla för flera motorer på serien men på SP är bara en motor inkopplad på serien)
    3. ”x” = kommandokoden
    4. ”y” = kommandovärde eller adresser
    5. ”=” = checksumma

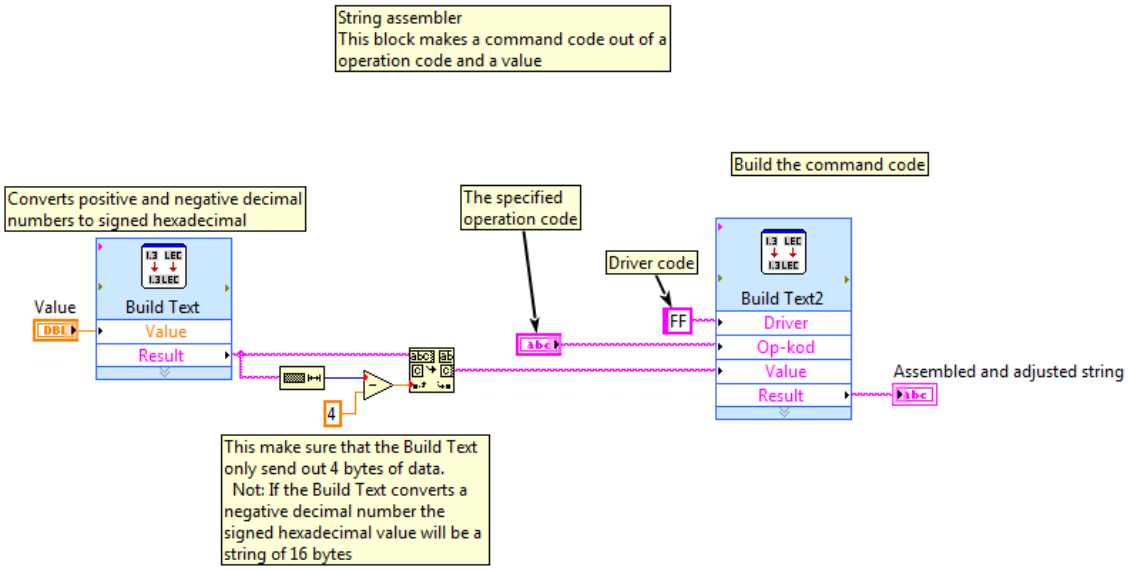


Fig. 14 ”String assemble”.

- ”String splitter”

Det finns två olika ”String splitter” en allmän och en som är anpassad för ”Read position”. Efter att ett kommando som ska läsa av ett visst register i BIVECTOR skickats returneras en sträng av formen ”Yxxxx/CC” där ”Y” står för ett godkänt svar, ”CC” för checksumman och ”xxxx” (kan variera i längd) för registervärdet (beroende på vad som skulle läsas av). ”String splitter” delar upp den returnerade strängen genom att ta bort ”Y” så satt strängen blir ”xxxx/CC” för att sen ta ut allt innan ”/” så att strängen blir ”xxxx”, alltså registervärdet. Sista blocket omvandlar det hexadecimala registervärdet till decimalt.

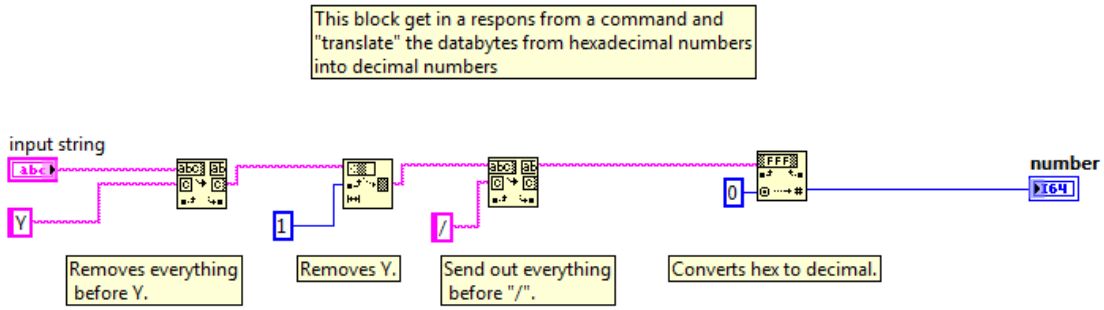


Fig. 15 ”String splitter”.

”String splitter - position” skiljer sig från den allmänna eftersom att den är specialanpassad för ”Read position”. ”String splitter – position” får in en kod enl. ”YAAAAbbbb/CC” som den enl. ovan minskar till ”AAAAbbbb” vilket innehåller motorns position. ”AAAA” står för hela varv och ”bbbb” för delar av ett varv enl. BIVECTOR, motorn har en utväxling på 8:1 vilket betyder att då BIVECTOR tror att den roterat motorn ett varv har den endast roterat en åttondels varv.
För att få ut antal varv och grader delas ”AAAA” och ”bbbb” upp och omvandlas från hexadecimala tal till decimala. På grund av utväxlingen blir antalet varv $\frac{"AAAA"}{8}$.
Resten av divisionen plus det från ”bbbb” står för antal grader. Eftersom ”bbbb” står för delar av varv hexadecimalt blir omvandlingen från hexadecimalt till decimalt inte korrekt utan det divideras med en faktor på 16^4 så att talet blir decimalt decimala tal.
Vid omvandlingen till grader multipliceras 360 och på grund av utväxlingen divideras 8, detta blir $\frac{360}{8} = 45$.

Omvandlingen sker enl. följande formler:

$$\frac{"AAAA"}{8} = x \qquad där\ x = Varv$$

$$(rest + ("bbbb" * 16^4)) * 45 = grader$$

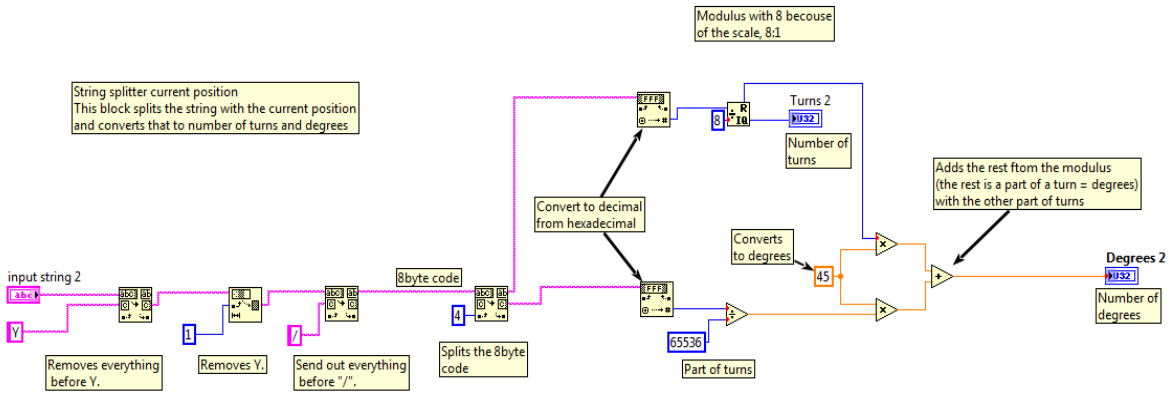
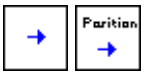


Fig. 16 ”string splitter – position”.

- “Write to BIVECTOR” 

  Detta är ett block som finns i två varianter, den första har bild av pil och på den andra står det ”Position” med en bild av en pil under. Blocken används för att skapa en tidsfördröjning mellan varje byte i ett kommando som skickas till BIVECTORN, den klarar inte av att ta emot en sträng av bytes.
  Till ”Write buffer” skrivs kommandon in och antalet byte i läggs till ”N”. Kommandot skickas vidare till en ”for” loop, som loopar ”N” gånger, som plockar ut en byte i taget ur kommandot och skickar vidare den till ”VISA write” som skriver till BIVECTORN. Efter varje byte skrivits läggs en tidsfördröjning på antingen 55 (första blocket) eller 62 ms (andra blocket) vilket beror på hur långt kommunikationen är samt hur säker responsen behöver vara.
  För att blocket inte ska skicka till BIVECTOR då det inte är en byte att skicka används blocket ”Bytes at Port” som väntar på en byte innan det skickas.

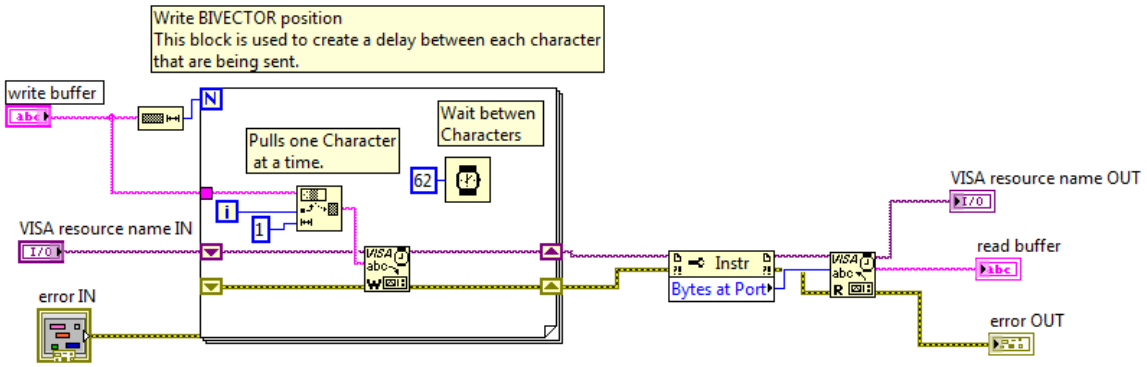


Fig. 17 "Write to BIVECTOR".

- "Load table"

"Load table" används för att ladda in olika tabeller med inställningar, på olika tabeller kan unika inställningar ställas in för olika uppgifter. Tabellerna som används i programmet är nummer 57 för hastighets styrning (i BivWin kallas denna "General purpose 25") och 59 för position styrning / "Jog" (i BivWin kallas denna "General purpose 27").
För att kunna ändra på värden på BIVECTOR måste först en editor vara öppnad varefter den specifika tabellen öppnas.

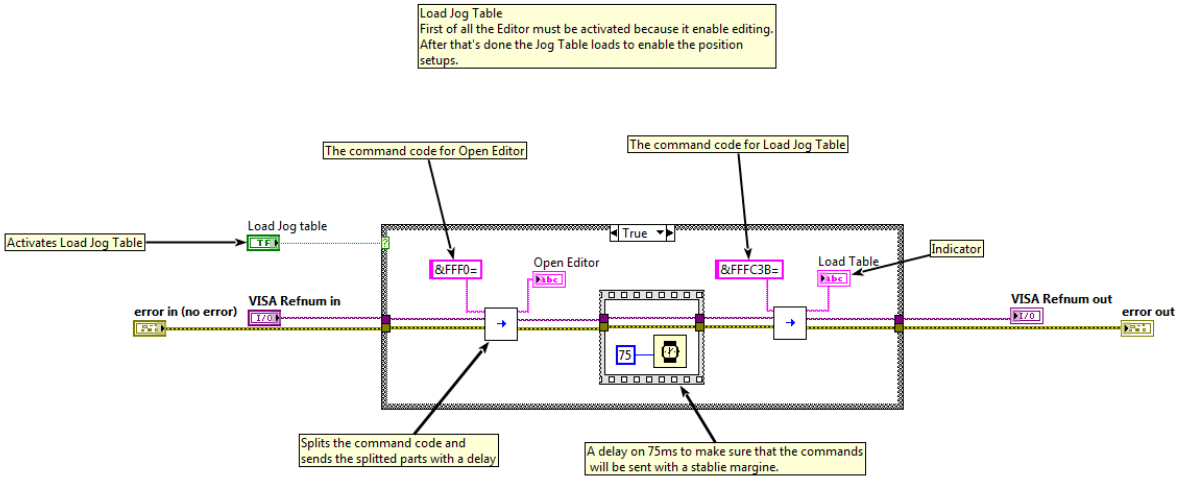


Fig. 18 "Load table".

- ”Max pos and neg torque” 

För att ställa in de maximala positiva och negativa vridmomentgränserna behövs kommandokoden och kommandovärdet för både maximala positiva och negativa vridmomentgränserna. Kommandokoden för max pos = ”28” och för max neg = ”29”. Omvandling från kommandovärde till vridmoment beräknas med storleken på ett tecknat hexadecimalt kommandovärde, på fyra bytes, genom maximala motorvridmomentet:

$$\frac{2^{15}}{36{,}4} = \frac{32768}{36{,}4} \approx 901$$

I omvandlingen är inte motorutväxling på 1:8 inräknad.
Då kommandokoderna och värdena angivets sammanställs de två koderna med ”string assemble” och skrivs ut med ”Write to BIVECTOR”.

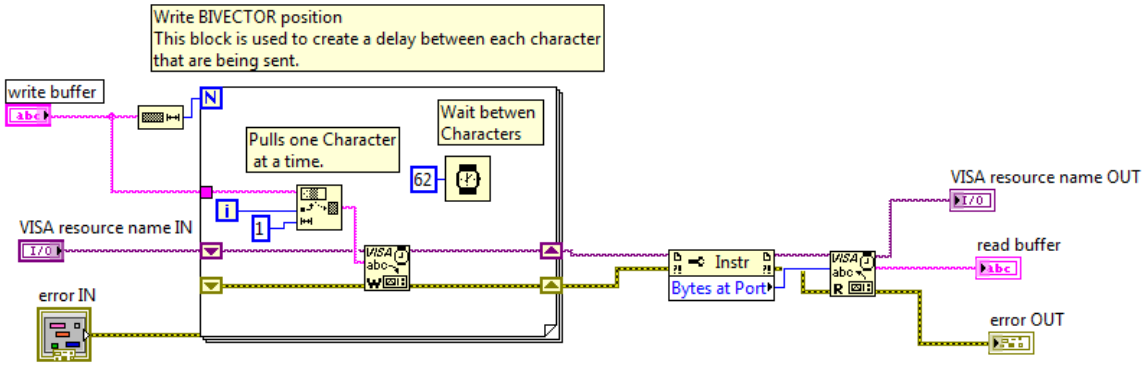


Fig. 19 ”Max pos and neg torque”.

- ”Set speed mm/min” 

  Hastigheten hos på dragprovets mäts i mm/min och ställs in lika så i programmet. Utväxlingen mellan dragprovets deformationshastighet och kommandovärdet beräknas utifrån motorns maximala hastighet, motorutväxlingen på 8:1 (hastighetsutväxlingen är motsatt till vridmomentsutväxlingen), skruvarnas stigning på 1mm på båda plattorna (skruvutväxling 2:1)och storleken på ett tecknat hexadecimalt kommandovärde, på fyra bytes:

$$\frac{2^{15}}{2 \cdot 7324{,}2} \cdot 8 = \frac{2^{17}}{7324{,}2} = \frac{131072}{7324{,}2} \approx 17{,}9$$

  När kommandovärdet beräknats assembleras kommando koden med ”String assemble” och skrivs ut med ”Write to BIVECTOR”.

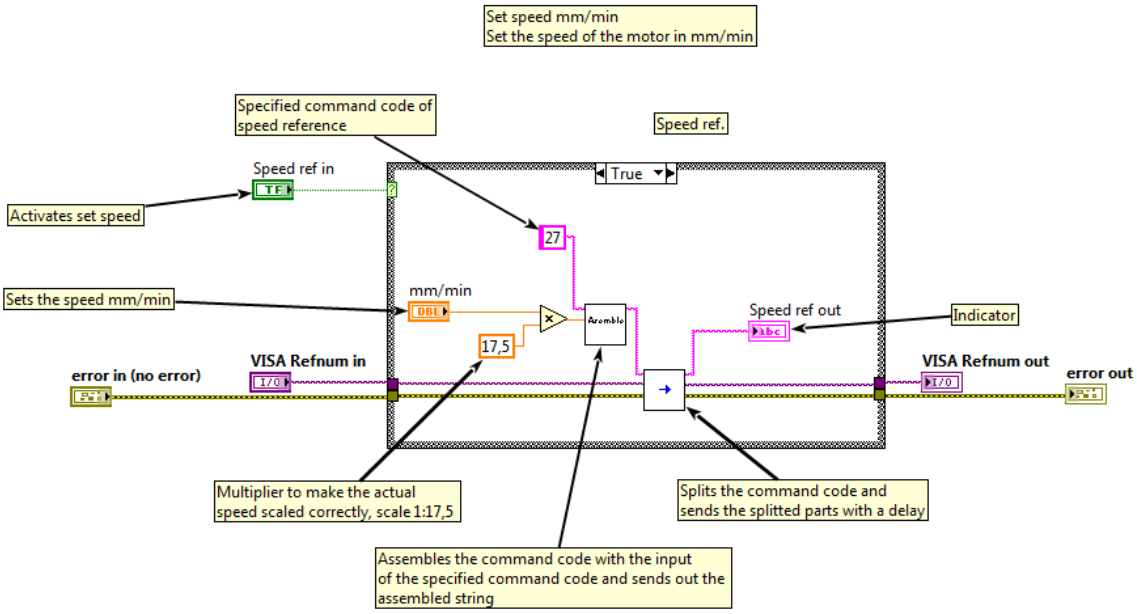


Fig. 20 ”Set speed mm/min”.

- "Start motor" 

  När motorn startas behövs det bestämmas vilka inställningar som ska vara aktiva under körning. Inställningarna är sparade i en tabell. I kommandot är tabell numret angivet samt kommandokoden för att starta motor, sen skickas kommandot med ”Write to BIVECTOR”.

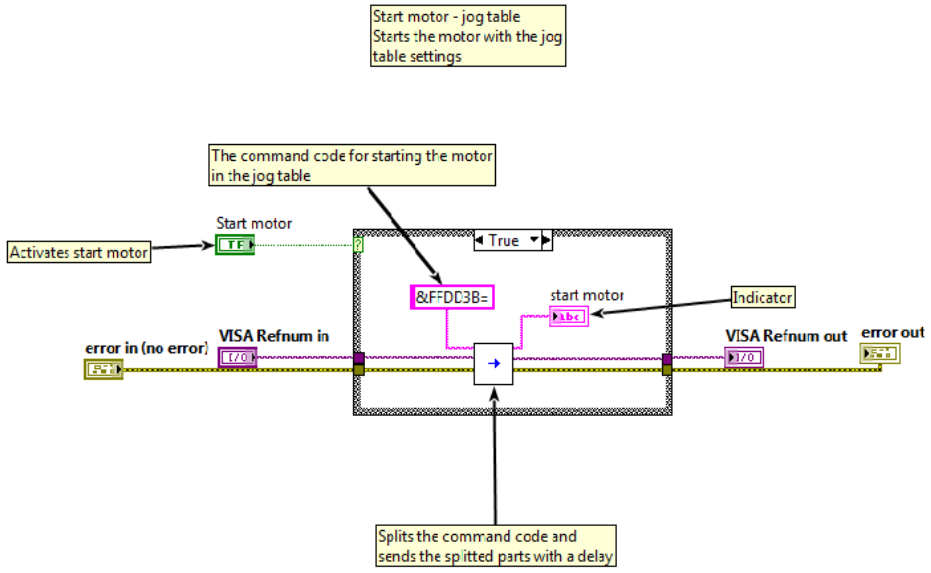


Fig. 21 "Start motor".

- "Stop motor" 
  Skickar kommando till BIVECTOR med ”Write to BIVECTOR” om att stoppa motorn. Ibland kan ”Stop motor” bromsa hårt vilket skapar onödigt slitage.

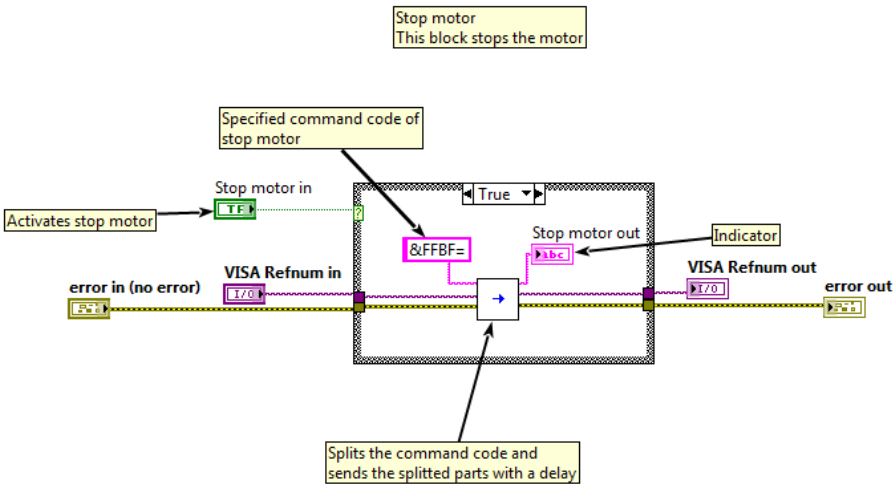


Fig. 22 "Stop motor".

- “Soft stop”

  “Soft stop” stoppar motorn mjukt. Först ställer den motorns hastighet till noll och väntar 150 ms för att säkerställa hastighetsminskningen (om motorhastigheten var stor) för att till slut stoppa motorn.

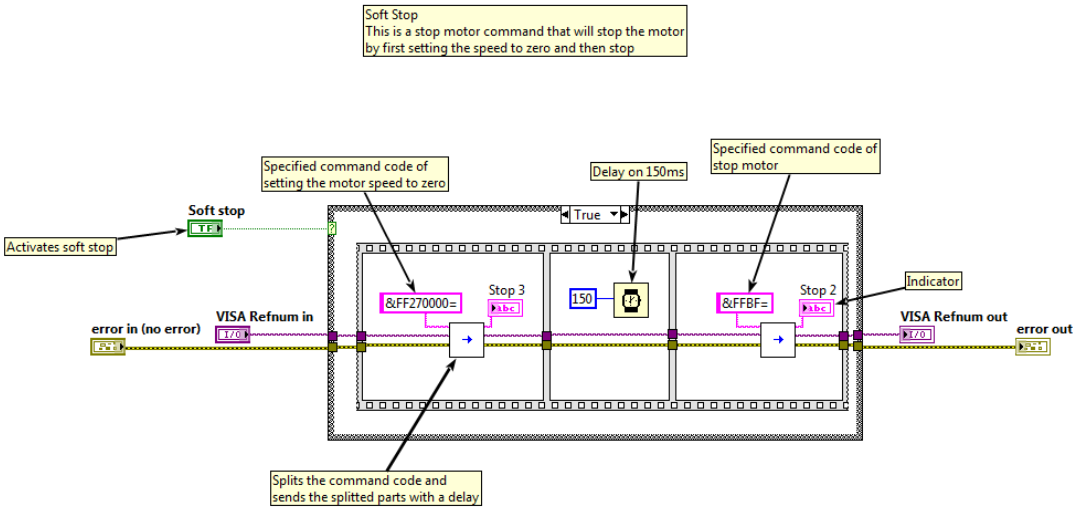


Fig. 23 “Soft stop”.

- “Read position”

  ”Read position” läser av motorns aktuella position från en adress som automatiskt uppdateras. Kommandot för att läsa av adressen skickas med ”Write to BIVECTOR – position”. Det returnerade värdet från andressen behandlas med ” String splitter - position” så att antalet varv och grader anges.

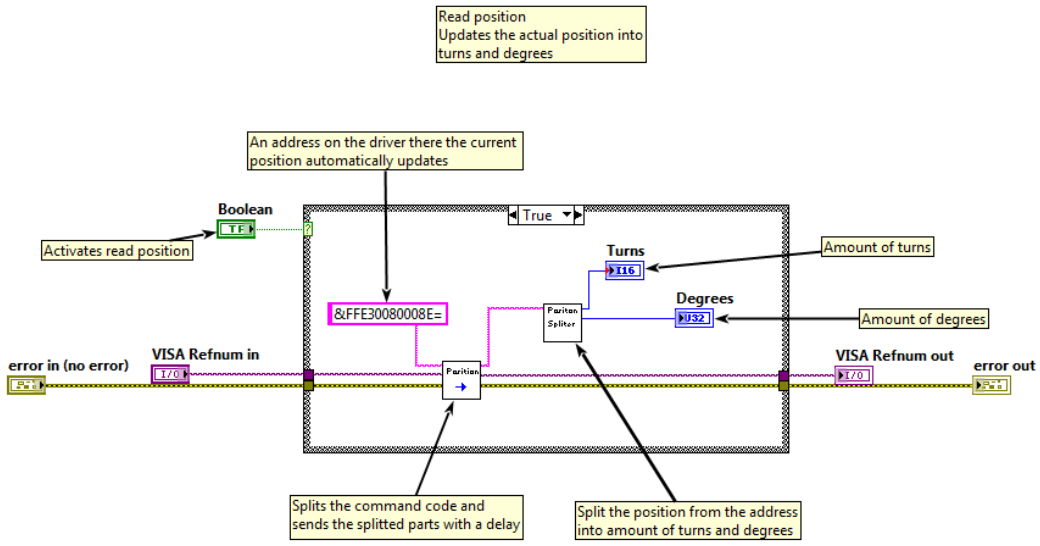


Fig. 24 “Read position”.

- ”Jog” (Roterar motorn x antal varv)

  ”Jog” är en funktion som läser av motorns aktuella position och flyttar den till en annan. Den finns i fyra olika valmöjligheter av vart den flyttar motorn. Två stycken flyttar motorn ca en sextondels av ett varv åt positiv motsvarande negativ riktning, de två andra flyttar motorn likadant men ett helt varv. Funktionerna kallas ”Jog+”, ”Jog-”, ”Jog++” och ”Jog--”. ”Jog” hämtar dess inställningar med ”Load table – Jog” för att sen hämta den aktuella positionen på adressen ”0090008E” hexadecimalt och omvandlas till decimalt med ”String splitter – Jog”. Det decimala talet adderas/subtraheras med den summan med antalet varv motorn ska rotera. Sen skickas värdet med ”Write to BIVECTOR” och startas med ”Start motor – Jog”.

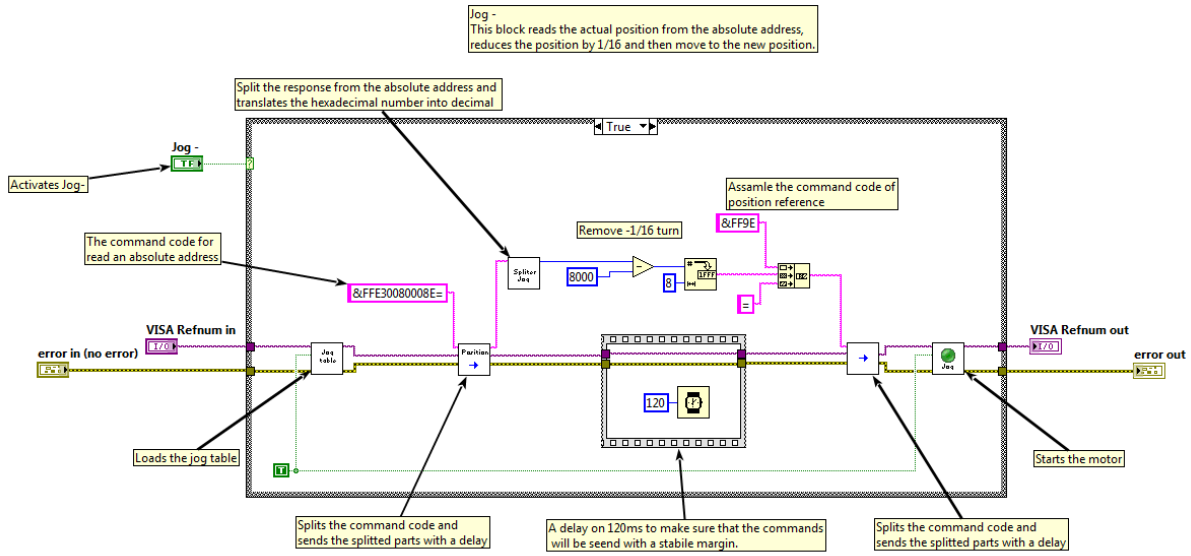


Fig. 25 ”Jog”.

- ”General settings torque limits”

  ”General settings torque limits” ställer in de maximala positiva och negativa vridmomentsgränserna på samtliga tabeller i användning, i detta program används två tabeller men tabellen för ”Jog” är förinställd och ändras inte på.
  Först öppnas hastighetstabellen med ” Load table - speed” varefter en fördröjning på 75 ms innan vridmomentsgränserna sätts med ” Max pos and neg torque”.

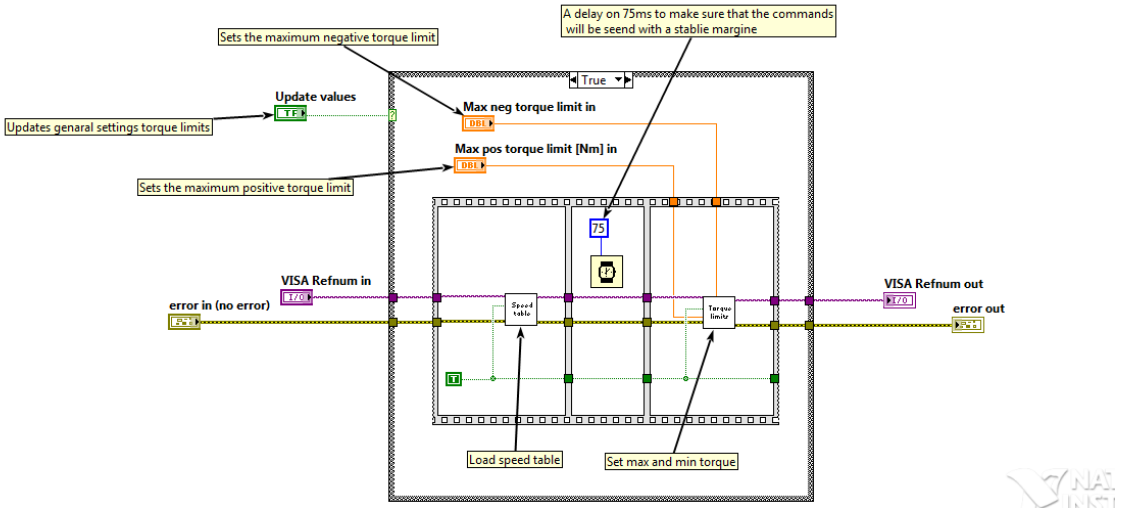


Fig. 26 ”General settings torque limits”.

- ”Update variables”

  ”Update variables” används för att läsa av parametrar från den aktiva tabellen i BIVECTOR. De parametrar som den läser av från är:

  1. Speed ref [mm/min].
  2. Pos max torque.
  3. Neg max torque

  De följande parametrar används av ett annat program kallat ”provning av lås”.

  4. Pos proportional gain.
  5. Pos integral gain. (Används för tillfället inte på grund av att BICVECTOR inte kan hantera den.)
  6. Max pos speed [rpm].
  7. Max neg speed [rpm.]

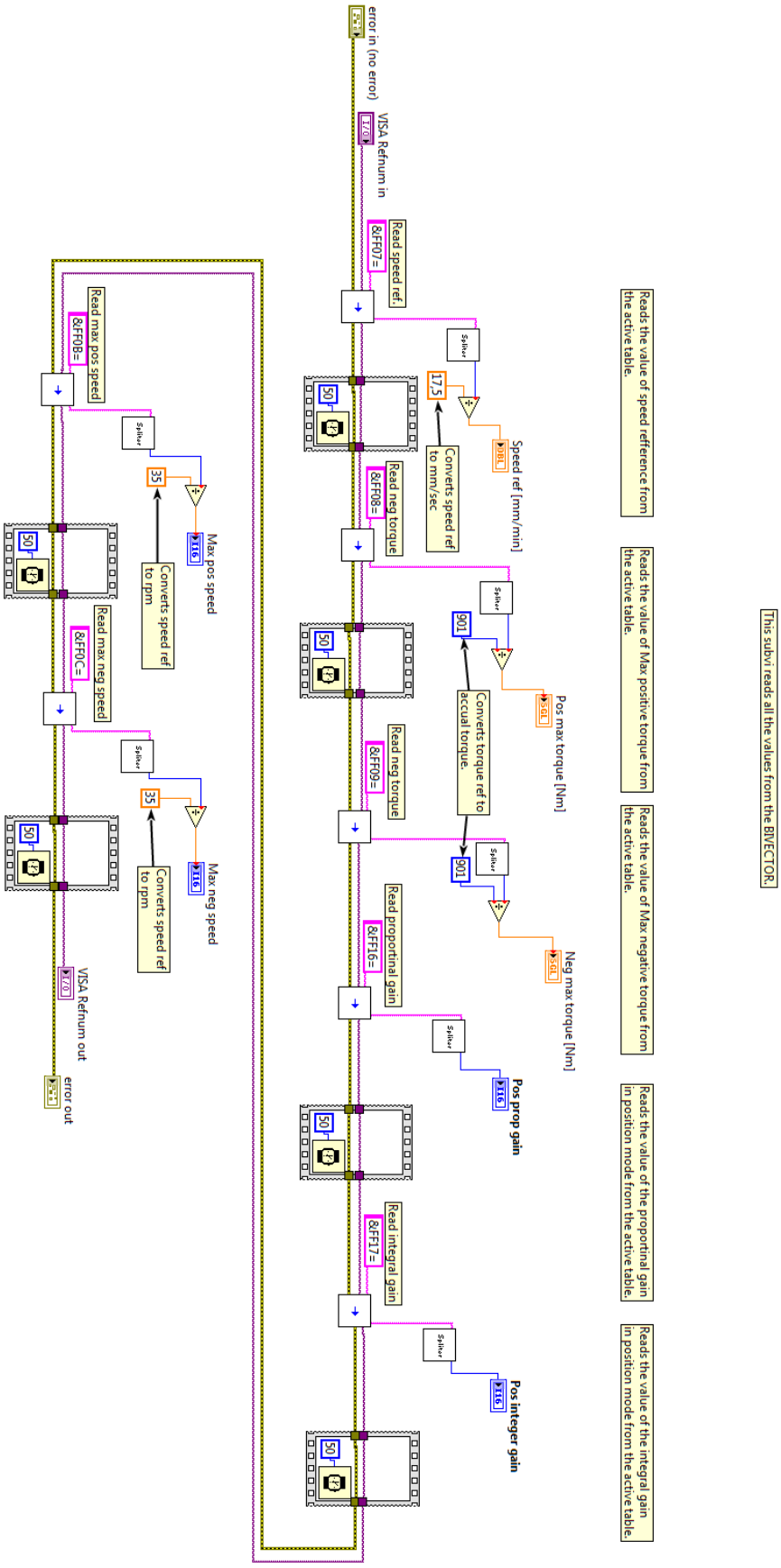


Fig. 27 ”Update variables”.

Bilaga B: Manual för Provning av lås

Detta program hanterar utmaningstester för lås. Nedan är en beskrivning över hur programmet används och hur det är uppbyggt, beskrivning av vad blocken utför i programmet beskrivs längst ner.

Hur programmet används

Programmet används för provning av lås i den formen att den cyklar motorn mellan två antingen förbestämda eller angivna positioner. Dessa två delar är uppdelade i två flikar som är oberoende av varandra. I båda flikarna visas funktioner så som att välja COM-port, stänga av programmet, start och stop av motor samt en funktion ”Homing” som nollställer den aktuella positionen.

OBS! Inga ändringar behöver göras mellan användning av BivWin och dragprovstyrningen.

Flik ett – Key turner

”Key turner” används för att cykla ett nyckellås mellan 0° och 90°. Nedan beskrivs hur man använder fliken vid ett test.

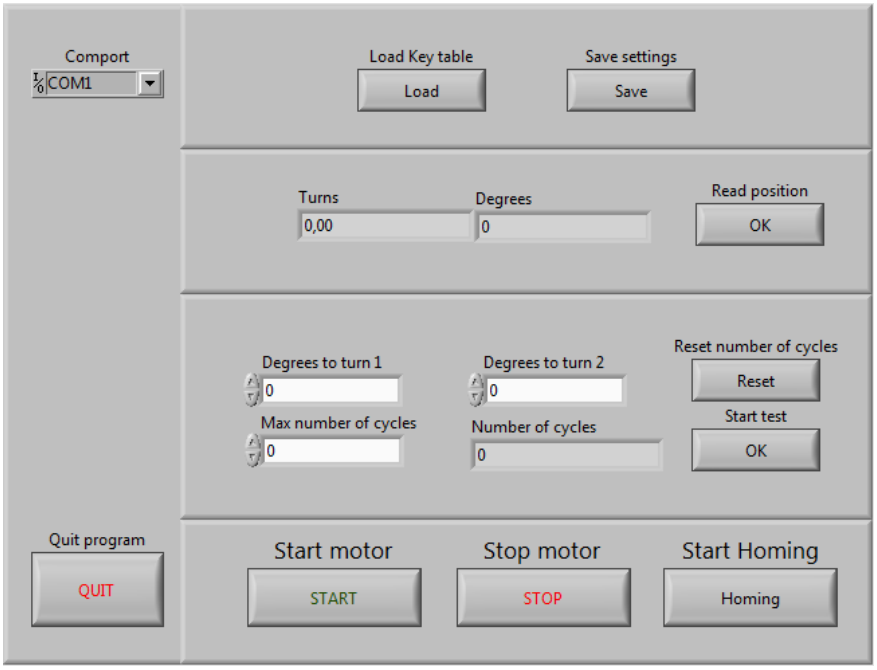


Fig. 28 "Key turner".

1. Nyckelvridningstabellen laddas genom att ”Load key table” aktiveras.
2. Inställning av vilka positioner motorn skall gå emellan (optimerad för <90 grader).
3. Låset sätts in så att det är i startposition
4. Positionen på motorn sätts till 0 genom att aktivera knappen ”Start Homing”.
5. ”Read position” aktiveras för att säkerställa att motorn är på position 0.
6. Motorn startas med knappen ”Start motor”, efter motorn är startad kan testet startas genom att knappen ”start test” aktiveras.

Flik två – Position mode

”Position mode” används för att cykla motorn mellan två positioner. Nedan beskrivs hur man använder fliken vid ett test.

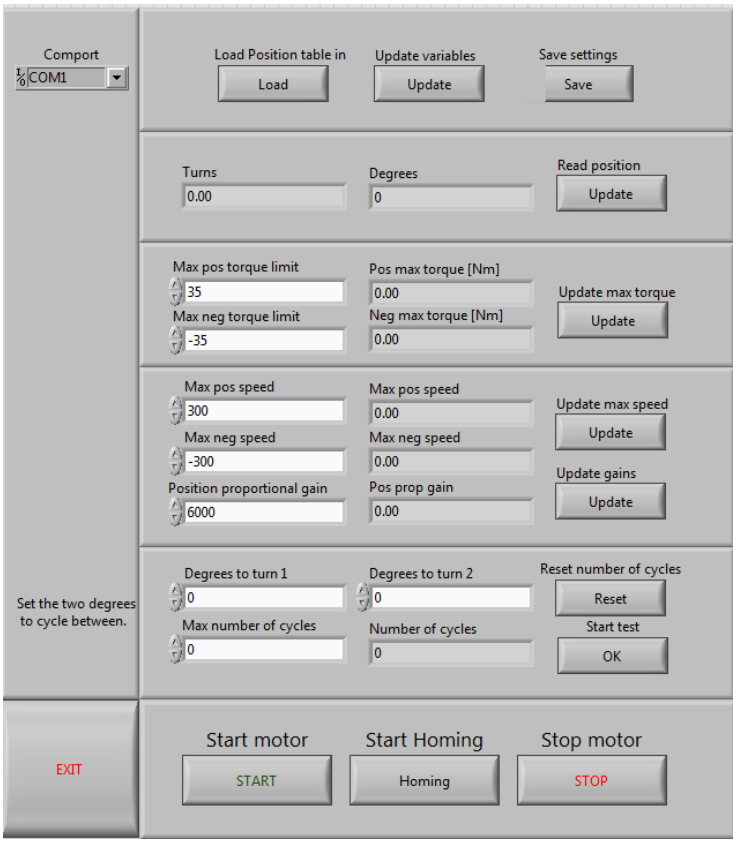


Fig. 29 "Position mode".

1. Tabell för positions table laddas in genom ”Load Position table” aktiveras.
2. Inställning för max och min vridmoment görs och skickas till BIVECTOR via knappen ”Update max torque”, värdet läses av från BIVECTOR för att säkerställ att värdet har ändrats.
3. Inställning av min- och maxhastighet[rpm] görs och skickas till BIVECTOR via knappen ”Update max speed”, värdet läses av från BIVECTOR för säkerställ att värdet har ändrats.
4. Inställning av ”Position proportional gain”, värdet skickas till BIVECTOR med knappen ”Update gains”, värdet läses av från BIVECTOR för att säkerställa att värdet har skickats.
5. Inställning av vilka två positioner som motorn skall rotera emellan (maxintervall på 360 grader) samt inställning av antal cykler motorn skall utföra före den stannar.
6. Låset sätts in så att det är i startposition
7. Positionen på motorn sätts till 0 genom att aktivera knappen ”Start Homing”.
8. ”Read position” aktiveras för att säkerställa att motorn är på position 0.
9. Motorn startas med knappen ”Start motor”, efter motorn är startad kan testet startas genom att knappen ”start test” aktiveras.

Programmets uppbyggnad

Programmet kommunicerar med BIVECTOR vilket sker via en COM-port, för att välja vilken av datorns olika COM-portar som den använder används ”Comport”. För att säkerställa att rätt information har skickats används ”Even parity”. ”VISA serial” startar kommunikationen.

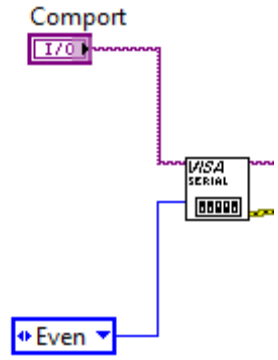


Fig. 30 Upprättning av kommunikation.

För att avsluta kommunikationen med BIVECTOR används ”Visa Close”. Vid användning av programmet används RS-485 men för att BIVECTOR ska kunna anslutas mot BivWin behövs ett RS-232 kommando skickas innan avslutning av kommunikation. Anledningen är att när ett RS-232 eller RS-485 kommando skickas ställs BIVETOR in mot användning av just den standarden, BivWin använder RS-232.

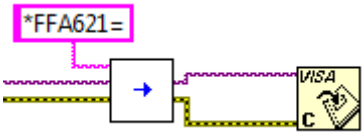


Fig. 31 Nedstängning av kommunikation.

För att programmet ska kunna använda sig av flikar används ett block som heter ”Tab Control”, fig. X. Då en flik är aktiv skickar ”Tab Control” ett villkor, för den specifika fliken, till en ”Case” funktion som utför operationer beroende på villkoret (bara en flik kan vara aktiv åt gången). Då Test mode är aktivt är bara kommandon från ”test mode” samt de parallella ”while” looparna som är tillgängliga.

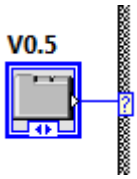


Fig. 32 Flikhanteraren.

Förklaring av vad som händer då blocken i ”Key turner” fliken aktiveras, från vänster till höger

- Laddar in tabellinställningar för position styrning.
- Nollställer aktuell position på motorn.
- Startar motor med förinställda värden (inställningarna för ”key turn” kan bara ändras på i BivWin).
- Stoppar motorn.
- Sparar inställningarna och stänger av editorn.
- Läser av motorns aktuella position.
- Stoppar motorn efter ett test är färdigt.
- (Hela sista rutan) Cyklar motorn mellan 0° och 90°.

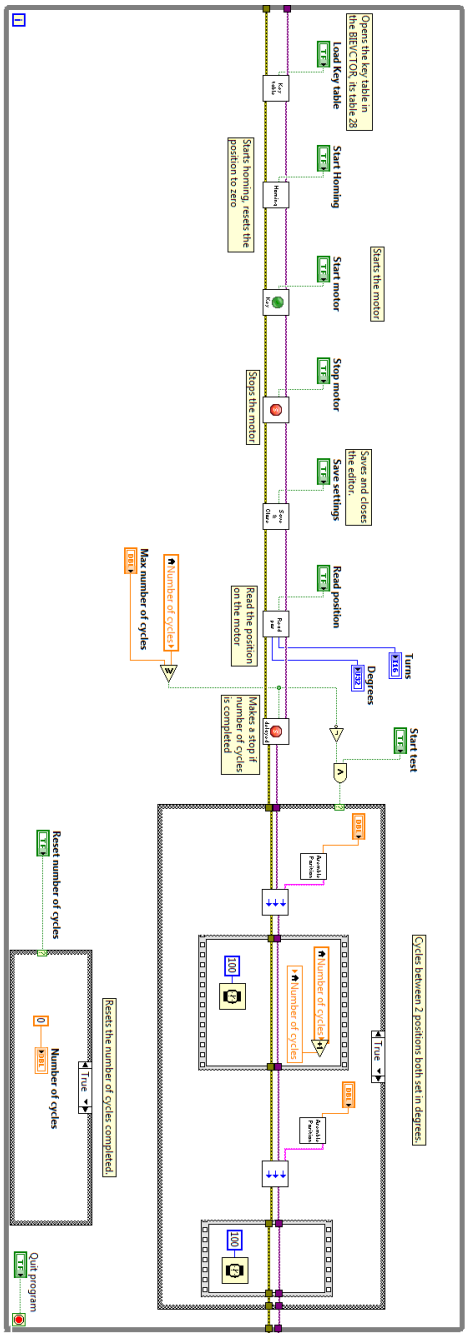


Fig. 33 Uppbyggnad av "Key turner".

Förklaring av vad som händer då blocken i ”Position mode” fliken aktiveras, från vänster till höger.

- Laddar in tabellinställningar för position styrning.
- Nollställer aktuell position på motorn.
- Startar motor med förinställda värden (inställningarna för ”key turn” kan bara ändras på i BivWin).
- Stoppar motorn.
- Sparar inställningarna och stänger av editorn.
- Ändrar inställningarna för maxhastigheten på motorn samt uppdarterar alla parametrar.
- Ändrar inställningarna för max vridmomenten på motorn samt uppdarterar alla parametrar.
- Ändrar inställningen för hur snabbt motorn når sin maxhastighet på motorn samt uppdarterar alla parametrar.
- Läser av motorns aktuella position.
- Uppdaterar alla parametrar.
- Stoppar motorn efter ett test är färdigt.
- (Hela sista rutan) Cyklar motorn mellan två angivna positioner.

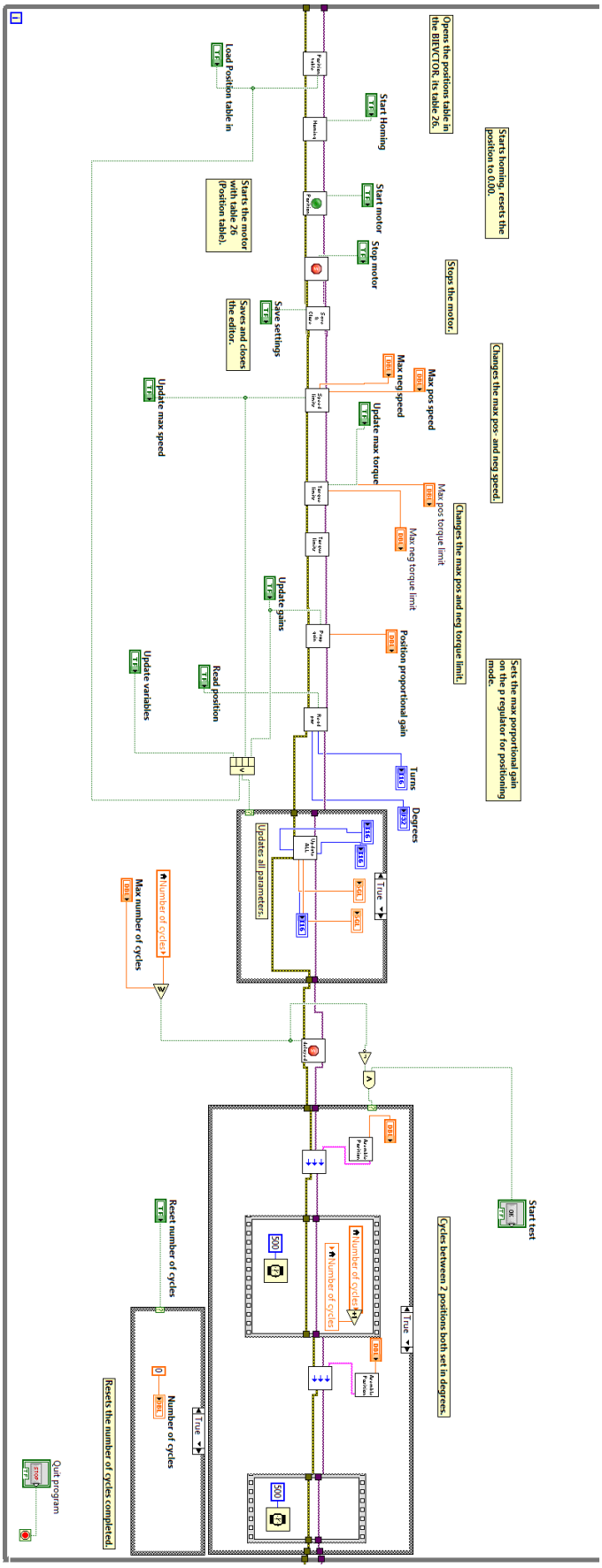


Fig. 34 Uppbyggnaden av "Position mode".

Beskrivning av ”Sub VI”

”Sub VI” används för att få en mer ren layout i programmet så det blir lättare att följa. Nedan beskrivs en detaljerad förklaring om de olika ”Sub VI”. ”VISA” och ”error” blocken i början och i slutet av ”Sub VI” är till för kommunikationen mot BIVECTOR. När en ”Sub VI” aktiverars arbetar den från vänster till höger.

- “String assemble” 

  “String assemble” skapar en kommandosträng med angiven specifik kommandokod (x) och värde (y). Kommandovärdet kan vara ett positivt eller negativt decimaltal som omvandlas till ett tecknat hexadecimaltal. Då omvandlingen sker från negativt decimaltal till hexadecimalt tal fylls värdet ut så det blir sexton bytes lång. Kommandovärdet som skickas får max vara fyra bytes så de 4 minst signifikanta bytes från omvandlingen tas ut.
  Kommandokoden byggs upp enl. mönstret ”&FFxy=” där de olika delarna betyder:

  6. ”&” = användning av RS-485
  7. ”FF” = motoradress (FF används om kommandon ska gälla för flera motorer på serien men på SP är bara en motor inkopplad på serien)
  8. ”x” = kommandokoden
  9. ”y” = kommandovärde eller adresser
  10. ”=” = checksumma

  ”String assemble – position” används för att omvandla grader till kommandon. Först tar den reda på hur många hela grader som är angivna men på grund av utväxlingen på 8:1 delas graderna med $\frac{360°}{8} = 45°$. Antalet varv (A) omvandlas till hexadecimalt och säkerställer att endast 4 bytes används. Resten från divisionen är antalet delar av ett varv (B). För att resten ska representera delarna av varvet delas den med 45°. Omvandling från decimala decimaltal till decimala hexadecimaltal sker med en faktor på $16^4 = 65536$. Värdena sammanställs sedan till formen ”&FF9EAB=”.

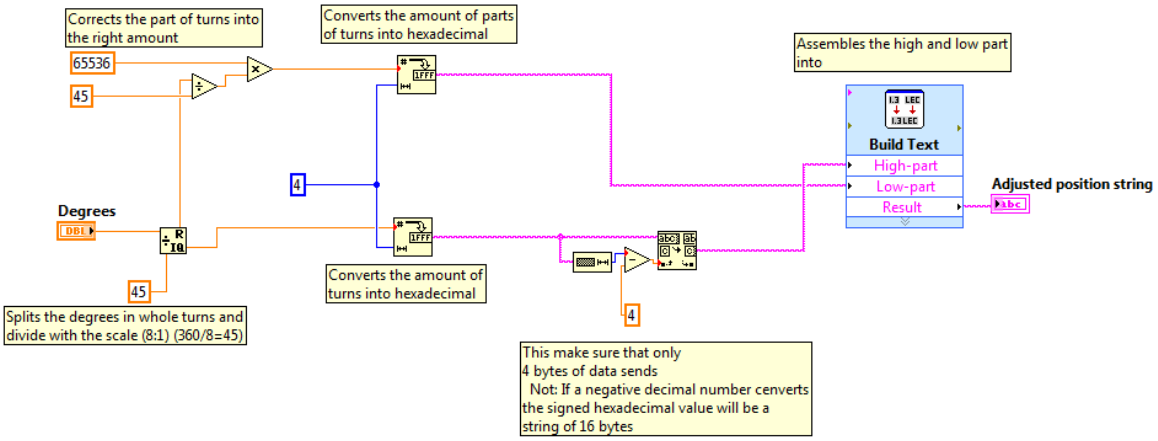
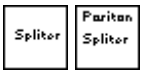


Fig. 35 ”String assemble”.

- ”String splitter”



Det finns två olika ”String splitter” en allmän och en som är anpassad för ”Read position”. Efter att ett kommando som ska läsa av ett visst register i BIVECTOR skickats returneras en sträng av formen ”Yxxxx/CC” där ”Y” står för ett godkänt svar, ”CC” för checksumman och ”xxxx” (kan variera i längd) för registervärdet (beroende på vad som skulle läsas av). ”String splitter” delar upp den returnerade strängen genom att ta bort ”Y” så satt strängen blir ”xxxx/CC” för att sen ta ut allt innan ”/” så att strängen blir ”xxxx”, alltså registervärdet. Sista blocket omvandlar det hexadecimala registervärdet till decimalt.

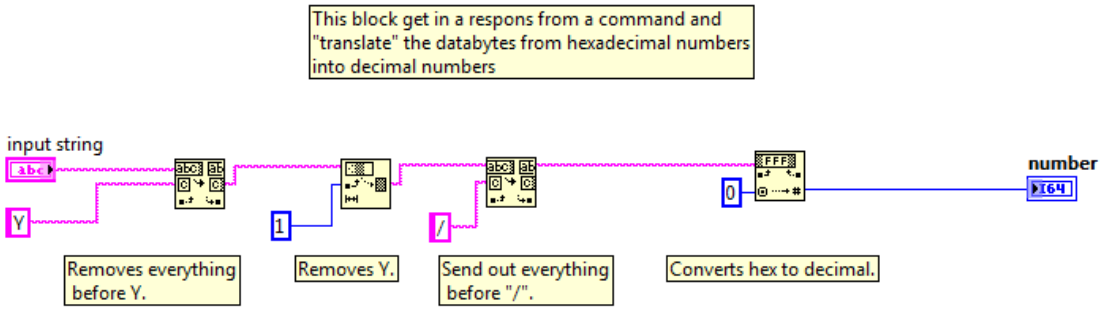


Fig. 36 ”String splitter”.

”String splitter - position” skiljer sig från den allmänna eftersom att den är specialanpassad för ”Read position”. ”String splitter – position” får in en kod enl. ”YAAAAbbbb/CC” som den enl. ovan minskar till ”AAAAbbbb” vilket innehåller motorns position. ”AAAA” står för hela varv och ”bbbb” för delar av ett varv enl. BIVECTOR, motorn har en utväxling på 8:1 vilket betyder att då BIVECTOR tror att den roterat motorn ett varv har den endast roterat en åttondels varv.
För att få ut antal varv och grader delas ”AAAA” och ”bbbb” upp och omvandlas från hexadecimala tal till decimala. På grund av utväxlingen blir antalet varv $\frac{"AAAA"}{8}$.
Resten av divisionen plus det från ”bbbb” står för antal grader. Eftersom ”bbbb” står för delar av varv hexadecimalt blir omvandlingen från hexadecimalt till decimalt inte korrekt utan det divideras med en faktor på 16^4 så att talet blir decimalt decimala tal.
Vid omvandlingen till grader multipliceras 360 och på grund av utväxlingen divideras 8, detta blir $\frac{360}{8} = 45$.

Omvandlingen sker enl. följande formler:

$$\frac{"AAAA"}{8} = x \qquad \text{där } x = Varv$$

$$(rest + ("bbbb" * 16^4)) * 45 = grader$$

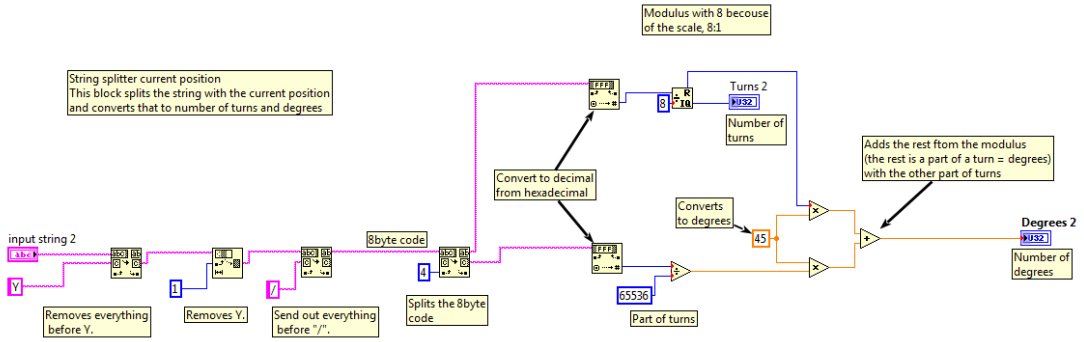


Fig. 37 ”string splitter – position”.

- “Write to BIVECTOR”

Detta är ett block som finns i tre olika utseenden där den första har en pil, den andra har tre pilar och på den tredje står det ”Position” med en pil under. Alla blocken används för att skapa en tidsfördröjning mellan varje byte i ett kommando som skickas till BIVECTORN, den klarar inte av att ta emot en sträng av bytes.
Till ”Write buffer” skrivs kommandon in och antalet byte i läggs till ”N”. Kommandot skickas vidare till en ”for” loop, som loopar från ”i”=0 till ”i”<”N” (vilket inkluderar ”i”=”N”), som plockar ut en byte i taget ur kommandot och skickar vidare den till ”VISA write” som skriver till BIVECTORN. Efter varje byte skrivits läggs en tidsfördröjning på antingen 55 (första blocket), 30 ms (andra) eller 62 ms (sista) vilket beror på hur säker kommunikationen mot BIVECTORN behöver vara (position blocket används sällan).
För att blocket inte ska skicka till BIVECTOR då det inte är en byte att skicka används blocket ”Bytes at Port” som väntar på en byte innan det skickas.

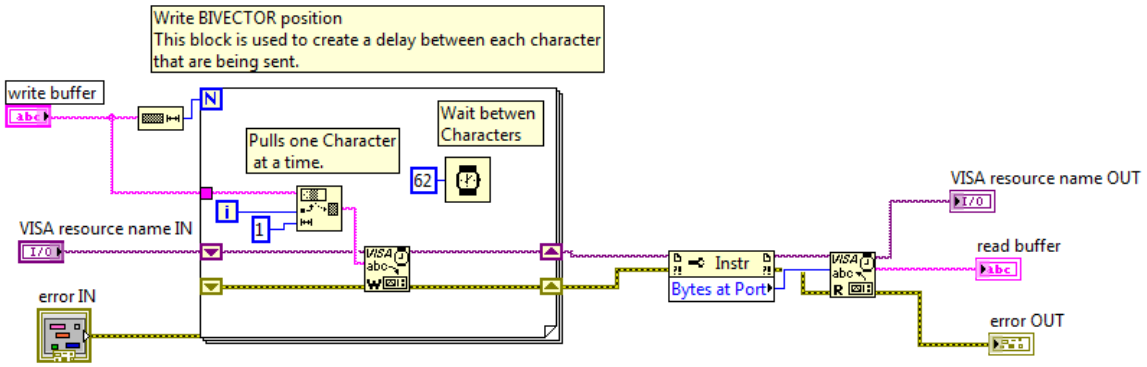


Fig. 38 “Write to BIVECTOR”.

- “Load table”

  ”Load table” används för att öppna upp olika tabeller med inställningar, på olika tabeller kan unika inställningar ställas in för olika uppgifter. Tabellerna som används i programmet är nummer 57 för hastighets styrning (i BivWin kallas denna ”General purpose 25”) och 58 för position styrning (i BivWin kallas denna ”General purpose 26”).

  För att kunna ändra på värden på BIVECTOR måste först en editor vara öppnad varefter den specifika tabellen öppnas.

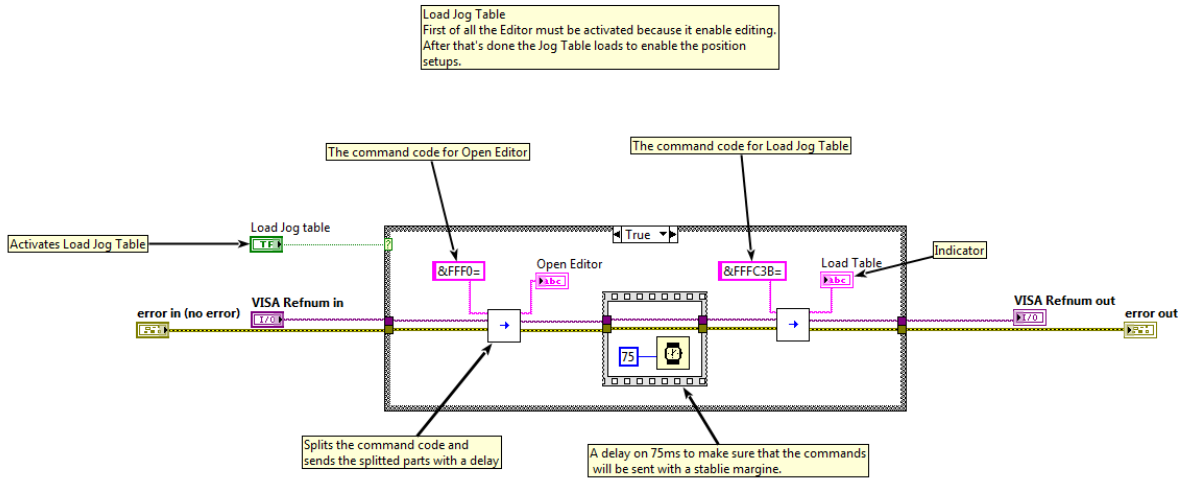


Fig. 39 “Load table”.

- ”Max pos and neg torque”

  För att ställa in de maximala positiva och negativa vridmomentgränserna behövs kommandokoden och kommandovärdet för både maximala positiva och negativa vridmomentgränserna. Kommandokoden för max pos = ”28” och för max neg = ”29”. Omvandling från kommandovärde till vridmoment beräknas med storleken på ett tecknat hexadecimalt kommandovärde, på fyra bytes, genom maximala motorvridmomentet:

$$\frac{2^{15}}{36{,}4} = \frac{32768}{36{,}4} \approx 901$$

  I omvandlingen är inte motorutväxling på 8:1 inräknad.

  Då kommandokoderna och värdena angivets sammanställs de två koderna med ”string assemble” och skrivs ut med ”Write to BIVECTOR”.

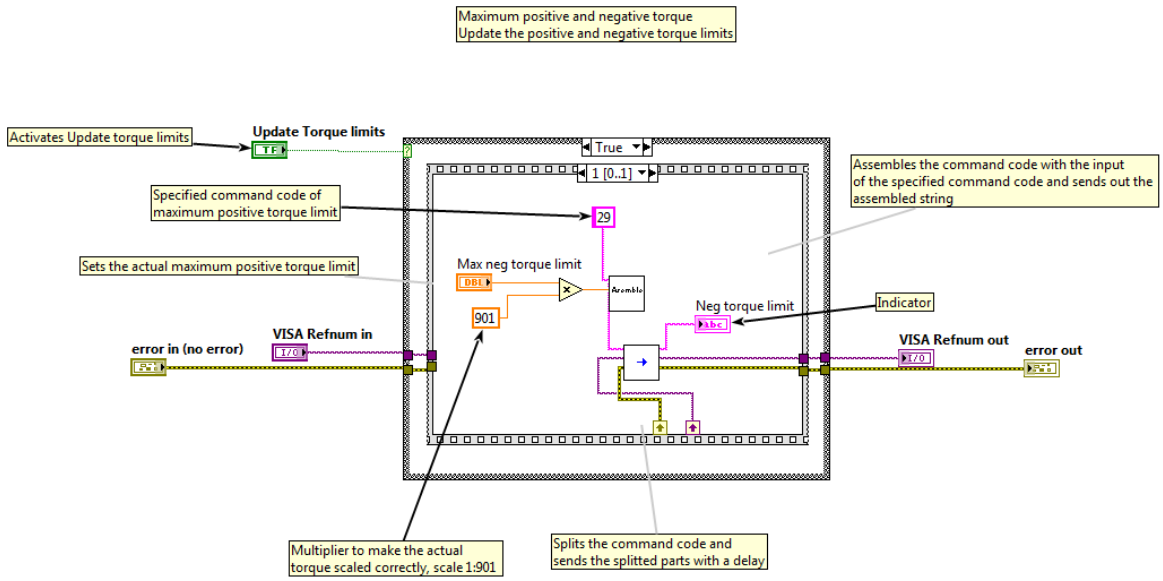


Fig. 40 ”Max pos and neg torque”.

- ”Homing”

  ”Homing” är en funktion som ställer motorns aktuella position till nolläget. För att möjliggöra aktivering av ”homing” krävs att en fysisk knapp som aktiverar en funktion kallad ”input high”. Då ”input high” är aktiverad kan ”homing” aktiveras och motorn nollställs.
  Kommandot skickas då till BIVECTOR med ”Wtrite to BIVECTOR”.

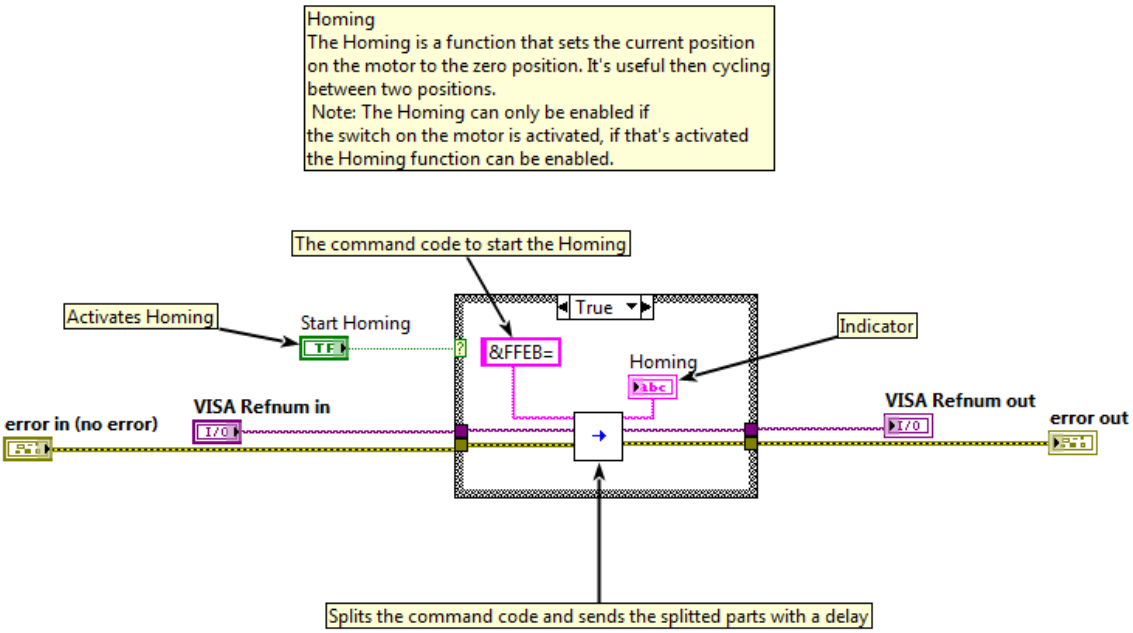


Fig. 41 "Homing".

- ”Set speed mm/min” 

Max hastigheten för motorn mäts i mm/min och ställs in lika så i programmet. Utväxlingen mellan motorvarvtalet och kommandovärdet beräknas utifrån motorns maximala hastighet, motorutväxlingen på 1:8 (hastighetsutväxlingen är motsatt till vridmomentsutväxlingen) och storleken på ett tecknat hexadecimalt kommandovärde, på fyra bytes:

$$\frac{2^{15}}{7324{,}2} \cdot 8 = \frac{2^{18}}{7324{,}2} = \frac{262144}{7324{,}2} \approx 35{,}8$$

När kommandovärdet beräknats assembleras kommandokoden med ”String assemble” och skrivs ut med ”Write to BIVECTOR”.

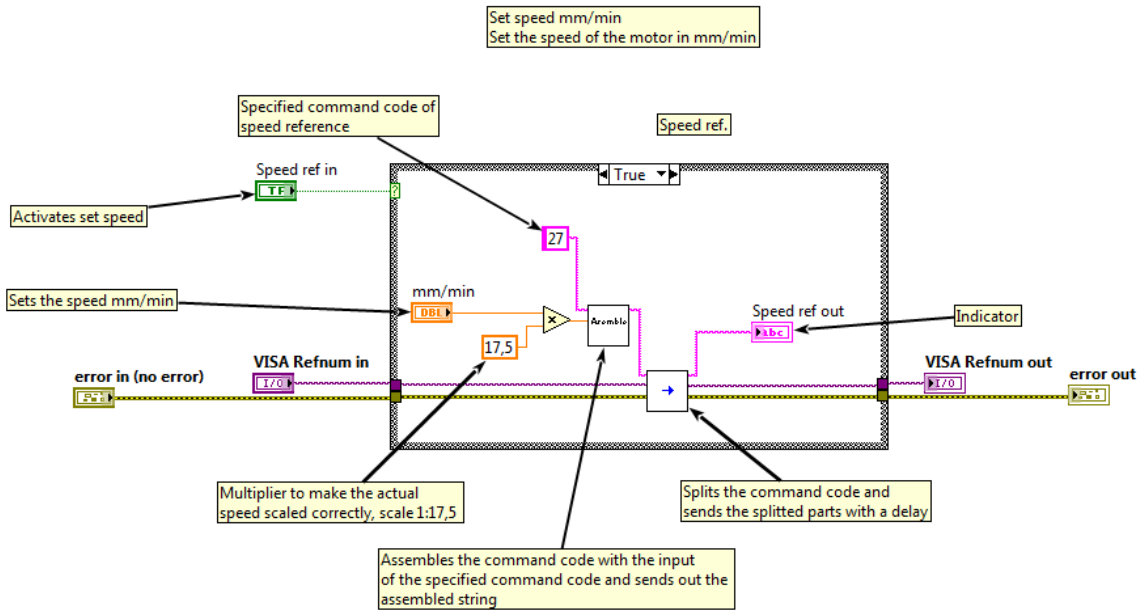


Fig. 42 ”Set speed mm/min”.

- "Start motor" 

  När motorn startas behövs det bestämmas vilka inställningar som ska vara aktiva under körning. Inställningarna är sparade i en tabell. I kommandot är tabell numret angivet samt kommandokoden för att starta motor, sen skickas värdet med "Write to BIVECTOR".

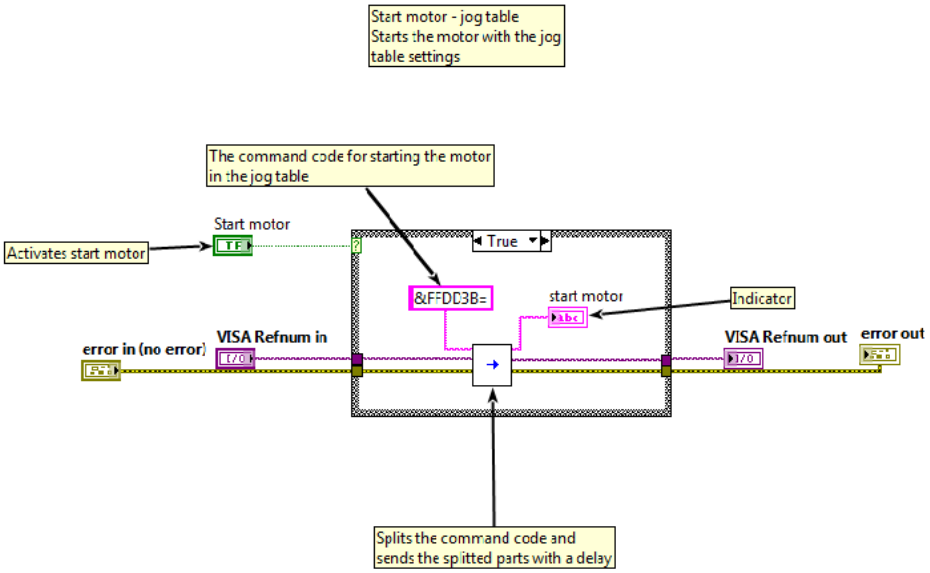


Fig. 43 "Start motor".

- "Stop motor" 
  Skickar kommando till BIVECTOR med "Write to BIVECTOR" om att stoppa motorn. Ibland kan "Stop motor" bromsa hårt vilket skaparonödigt slitage.

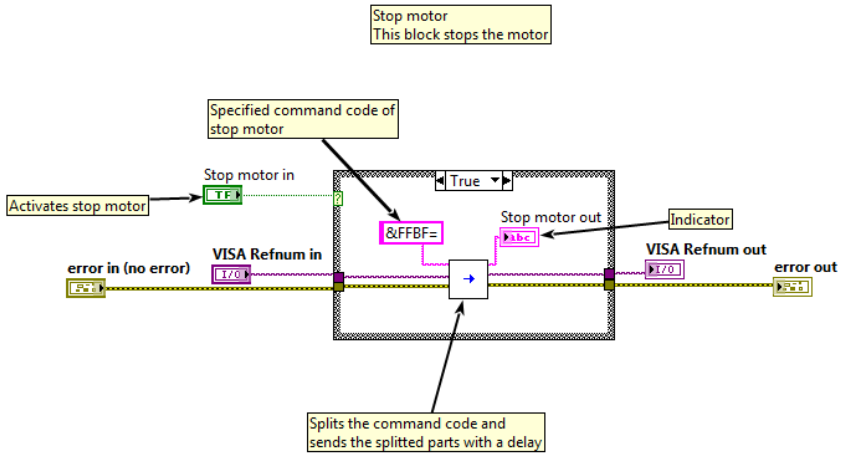


Fig. 44 "Stop motor".

- “Delayed stop”

  “Delayed stop” används efter att ett test har utförs. För att motorn inte ska bromsa innan motorn hunnit stanna används en delay på 1000 ms för att sen stopa motorn med ”Stop motor”.

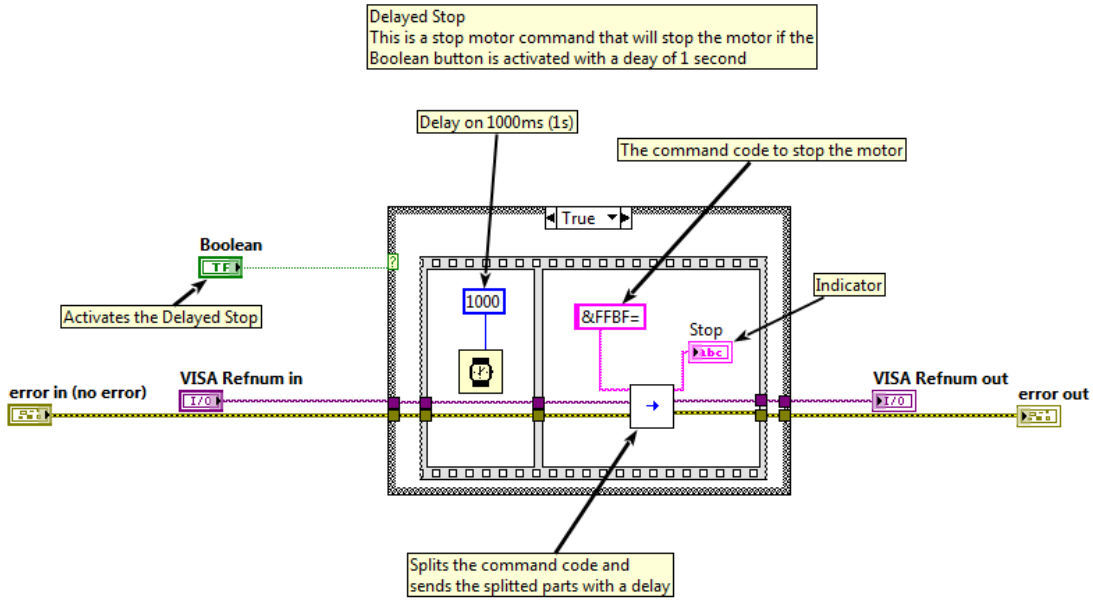


Fig. 45 "Delayed stop".

- ”Save and close editor”

  Skickar kommando till BIVECTOR med ”Write to BIVECTOR” om att spara inställningarna för och stänga av editor.

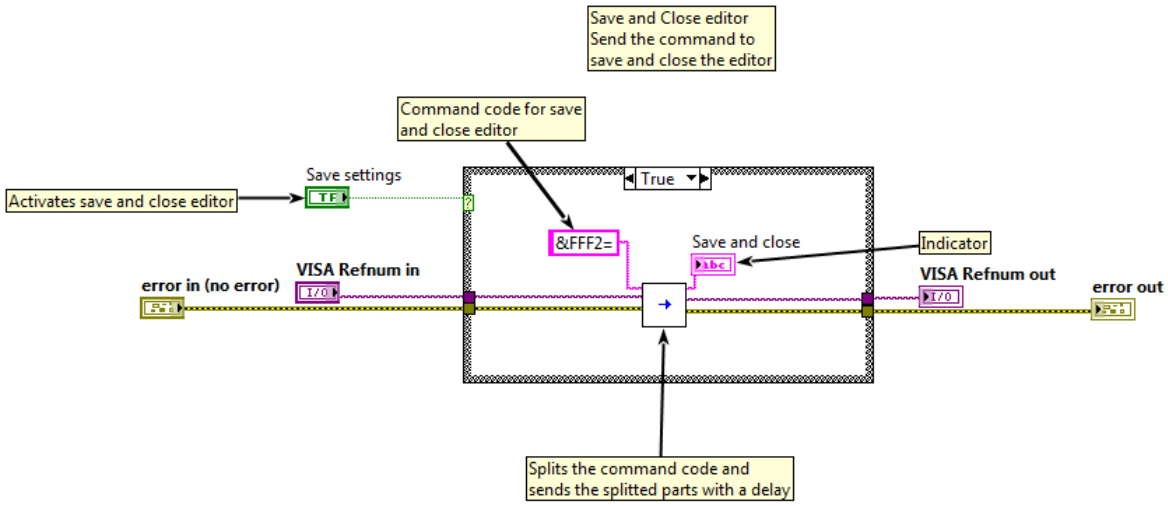


Fig. 46 "Save and close editor".

- ”Proportional and integral gains”

  Detta block ställer in värdena på ”proportional gain” och vad ”integral gain”. ”Proportional gain” används för inställning av hastighet då man kör positionsstyrt, vilket detta program gör. Det var först i slutet av exjobbet som det kom fram att ”integral gain” inte hade utvecklats för den aktuella BIVECTOR och används därför inte.

  För att skicka värden assemblerare ”string assemble” och för att skicka ut ”write to BIVECTOR”.

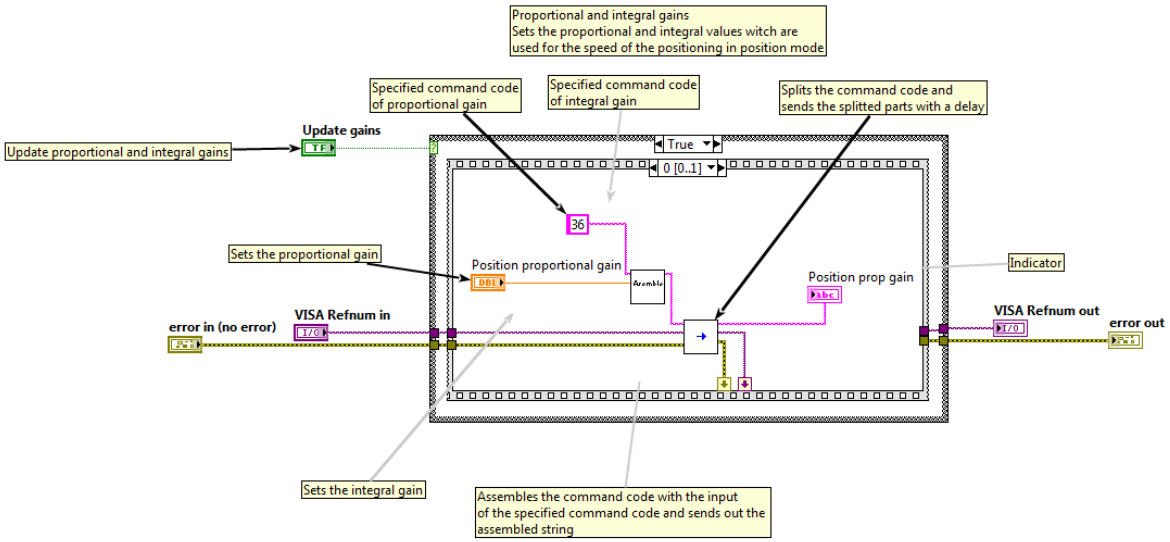


Fig. 47 ”Proportional and integral gains”.

- “Read position”

  ”Read position” läser av motorns aktuella position från en adress som automatiskt uppdateras. Kommandot för att läsa av adressen skickas med ”Write to BIVECTOR – position”. Det returnerade värdet från andressen behandlas med ” String splitter - position” så att antalet varv och grader anges.

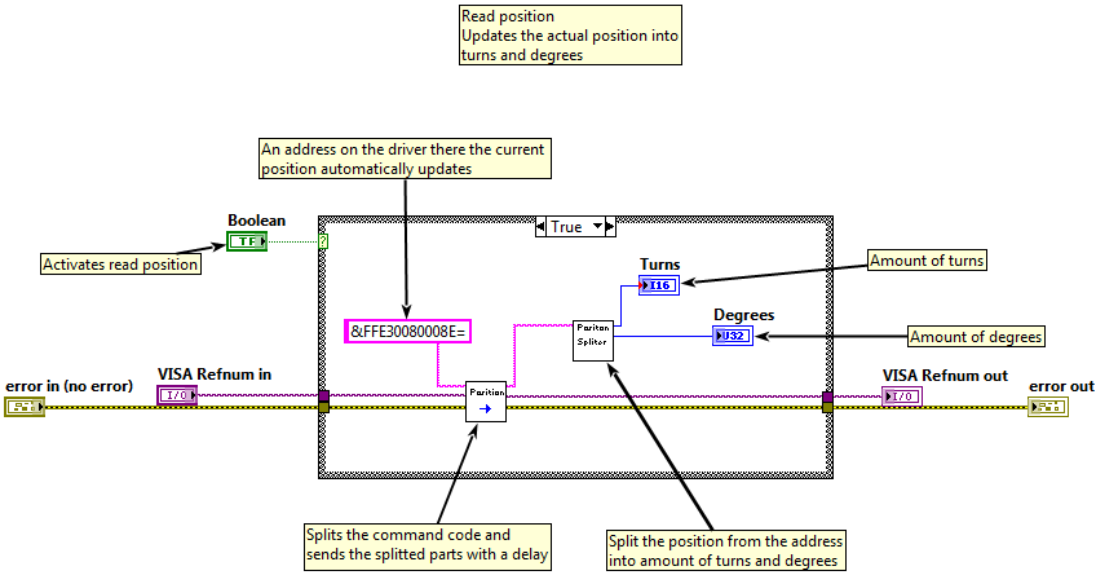


Fig. 48 “Read position”.

- ”Update variables”

  ”Update variables” används för att läsa av parametrar från den aktiva tabellen i BIVECTOR. De parametrar som den läser av från är:

  8. Speed ref [mm/min].
  9. Pos max torque.
  10. Neg max torque

  De följande parametrar används av ett annat program kallat ”provning av lås”.

  11. Pos proportional gain.
  12. Pos integral gain. (Används för tillfället inte på grund av att BICVECTOR inte kan hantera den.)
  13. Max pos speed [rpm].
  14. Max neg speed [rpm.]

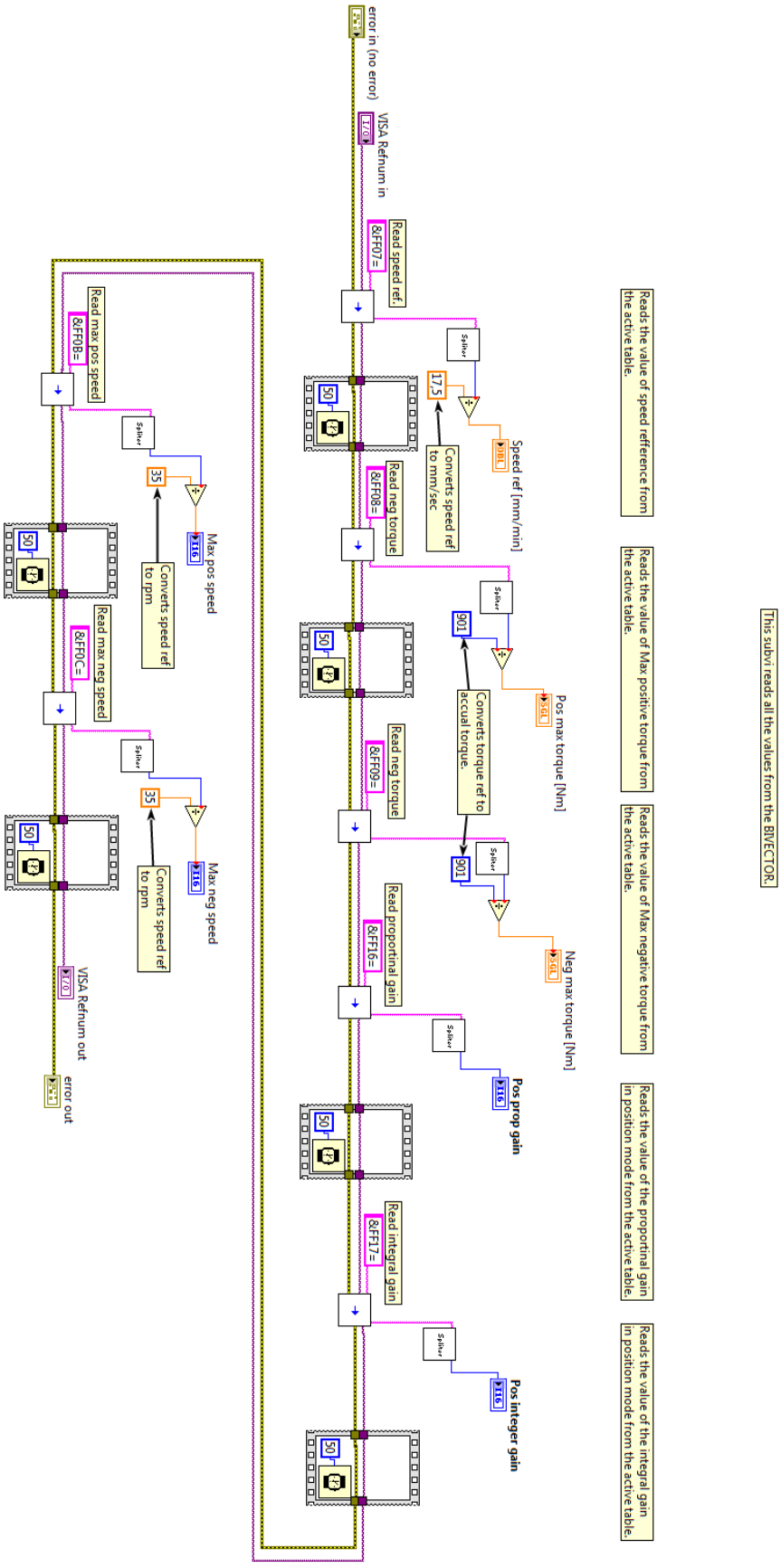


Fig. 49 ”Update variables”.

Bilaga C: Kommando koder till BIVECTOR

Command	Structure	Hex-code	Respons	Comments
Drive status	&DDA615/CC0	&FFA615	Y00/00	RFO
			Y01/01	GO
			Y02/02	FAIL
Start (run)	&DDDDxx/CC	&FFDDxx=	Y/N	xx=what table to start with.
Stop	&DDBF/CC	&FFBF=	Y/N	Stops the motor if it's in GO status with response Y, else it respose with N.
Editor Status	&DDA622/CC	&FFA622=	Yxx/xx	If xx=00 then editor is close, else it's open.
Open Editor	&DDF0/CC	&FFF0=	Y/N	Open editor if it's close, response Y, else it response N.
Close Editor	&DDF1/CC	&FFF1=	Y	Close editor, accepted if not in GO status.
Save	&DDED/CC	&FFED=	Y/N	Save if in RFO status, takes 3seconds.
Close Editor(2)	&DDF2/CC	&FFF2=	Y	Save and close editor, accepted if not in GO status and editor is activ, take 3 seconds.
Load Table	&DDFCxx/CC	&FFFCxx=	Y/N	Loding xx table, there xx is in hexadecimal form.
Operating Mode	&DD60xx/CC	&FF6050=	Y/N	Choose Operating Mode, like Speed Mode.
Position Ref	&DD9EXXXXxxxx/CC	&FF9E00004000=	Y/N	Ange antal varm som motorn ska rotera XXXX för hela varv, xxxx för del av varv.
Status on Position	&DD8E/CC	&FF8E=	YXXXXxxxx/CC	XXXXxxxx stands for the value of turns, XXXX in hole turns and xxxx in parts of a hole turn
Speed Ref	&DD27xxxx/CC	&FF27xxxx=	Y/N	xxxx= the speed of the motor (35:1).
Status on Speed	&DD07/CC	&FF07=	Yxxxx/CC	xxxx stands for whats in the Speed reference register.
Pos Torque Limit	&DD28xxxx/CC	&FF28xxxx=	Y/N	xxxx stands for the Pos Torque Limit (0000H-7FFFH).
tus on Pos Torque Li	&DD08/CC	&FF08=	Yxxxx/CC	xxxx stands for whats in the Pos Torque Limit register.
Neg Torque Limit	&DD29xxxx/CC	&FF29xxxx=	Y/N	xxxx stands for the Neg Torque Limit (8001H-0000H).
tus on neg Torque Li	&DD09/CC	&FF09=	Yxxxx/CC	xxxx stands for whats in the Neg Torque Limit register.
First Failure	&DDA62B/CC	&FFA62B=	YPPPPffff/CC	PPPP = protection flags, ffff = failure flags.
All Failure	&DDA62C/CC	&FFA62C=	YPPPPffff/CC	PPPP = protection flags, ffff = failure flags.
Warning	&DDA62A/CC	&FFA62A=	Yxxxx/CC	xxxx= warning flags.
Position prop gain	&DD36xxxx/CC	&FF36xxxx=	Y/N	Position propotional gain, for the speed in position mode.
Position prop status	&DD16/CC	&FF16=	Yxxxx/CC	xxxx=whats in the Position prop gain register.
Position integral gain	&DD37xxxx/CC	&FF37xxxx=	Y/N	Position integral gain, for the speed in position mode.
osition integral statu	&DD17/CC	&FF17=	Yxxxx/CC	xxxx=whats in the Position integral gain register.
Homeing prosedure	&DDEB/CC	&FFEB=	Y00/00	No Homing executed after the drive is powered.
			Y01/01	Homing in progress.
			Y02/02	Homing ompleted.
Enable ramps	&DD63xx/CC	&FF63xx=	Y/N	xx=00 for disable ramps, xx=11 for enable ramps
Ramps status	&DD43/CC	&FF43=	Yxx/CC	xx=00 for ramps disable, xx=11 for ramps enable
Read absolute adress	&&FFE3xxxxxxxx=		Y<value>	Reads a absolute adress.
Current position	&FFE30080008E=	&FFE30080008E=	Yxxxxxxxx	Reads the absolute adress where the current position is stored.