STADIA
HELSINGIN AMMATTIKORKEAKOULU

TEKNIIKAN JA LIIKENTEEN TOIMIALA

Sähkö- ja tietoliikennetekniikka

Elektroniikka ja automaatio

INSINÖÖRITYÖ

TAAJUUSMUUTTAJIEN SUORITUSKYVYN AUTOMAATTINEN TESTAUSYMPÄRISTÖ

Työn tekijä: Erno Pentzin
Työn valvoja: leht. Marko Uusitalo
Työn ohjaaja: ins. Jussi Rantanen

Työ hyväksytty: __. __. 2006

Marko Uusitalo
lehtori

STADIA
HELSINGIN AMMATTIKORKEAKOULU

ALKULAUSE

Tämä insinöörityö tehtiin ABB Oy, Drivesin Product AC -tulosyksikön tuotekehitysosastolle Helsingissä. Haluan kiittää tuotekehitysosaston tyyppitestaustiimiä sekä erityisesti ohjaajaani ins. Jussi Rantasta tuesta, jota olen saanut työtä tehdessäni.

Helsingissä 19.4.2006

Erno Pentzin

STADIA
HELSINGIN AMMATTIKORKEAKOULU

INSINÖÖRITYÖN TIIVISTELMÄ

Tekijä: Erno Pentzin	
Työn nimi: Taajuusmuuttajien suorituskyvyn automaattinen testausympäristö	
Päivämäärä: 19.4.2006	Sivumäärä: 54 s. + 3 liitettä
Koulutusohjelma: Sähkötekniikka	Suuntautumisvaihtoehto: Automaatio
Työn valvoja: leht. Marko Uusitalo Työn ohjaaja: ins. Jussi Rantanen	

Tämä insinöörityö tehtiin ABB Oy, Drivesin Product AC -tulosyksikön tuotekehitysosastolle Helsingissä. Työssä kehitettiin taajuusmuuttajien suorituskyvyn automaattinen testausympäristö. ABB:n taajuusmuuttajien suorituskykytestejä ei ole aikaisemmin automatisoitu. Testit on tehty käsin ja niiden suorittamiseen ja tulosten käsittelyyn on kulunut paljon aikaa. Automaattisella testauksella pyrittiin testien suorittamiseen ja tulosten käsittelyyn kuluvan ajan huomattavaan pienentymiseen. Työssä ei ollut tarkoituksena tehdä suorituskykytestejä vaan kehittää automaattinen testausympäristö eli suorituskykytestipenkki, jossa suorituskykytestit on mahdollista suorittaa. Työssä keskityttiin taajuusmuuttajan nopeus- ja momenttisäätäjien suorituskykyyn.

Työ toteutettiin suunnittelu- ja ohjelmointityönä. Testausympäristön laitteisto perustuu ABB:n tuotekehityslaboratorioiden olemassaoleviin testipaikkoihin. Testausympäristössä käytetään taajuusmuuttajien lisäksi pääasiassa kolmivaiheisia oikosulkumoottoreita. Lisäksi laitteistoon kuuluu ACS800-sarjan taajuusmuuttaja kuormakäyttönä, momenttianturi ja takometri eli kierrosnopeusmittari. Ohjelmointi tehtiin National Instrumentsin LabVIEW-ohjelmointiympäristön versiolla 8.0. Testausympäristön käyttöliittymänä toimii saman yrityksen TestStand-testausohjelmiston versio 3.5. Testattavien taajuusmuuttajien ohjausta ja momenttianturin lukemista varten ohjelmoitiin virtuaali-instrumentteja. Virtuaali-instrumentteja kutsutaan TestStand-testisekvensseistä. Testisekvenssit luodaan TestStandin sekvenssieditorilla ja suoritetaan sekvenssieditorissa tai operaattorin käyttöliittymässä.

Työn tuloksena syntyi taajuusmuuttajien suorituskyvyn automaattinen testausympäristö. Testausympäristöä voidaan hyödyntää sekä nykyisen että seuraavan sukupolven taajuusmuuttajien testauksessa. Sillä on mahdollista suorittaa yleisimmät taajuusmuuttajien suorituskykytestit, kuten nopeus- ja momenttisäätöjen staattinen ja dynaaminen tarkkuus, hyvin kattavasti. Testit voidaan automaattisesti suorittaa koko testikäytön sallimalla pyörimisnopeus- ja kuormitusalueella. Näytteenottotaajuus voi olla enintään 1 kHz luettaessa pyörimisnopeutta ACS800-sarjan taajuusmuuttajan kautta ja momenttianturia samanaikaisesti. Virtuaali-instrumenteista koostuvia testisekvenssejä voidaan vapaasti muokata ja kehittää testejä edelleen tai luoda kokonaan uusia testejä. Testausympäristö perustuu teollisuudessa yleisesti käytettyihin ohjelmistoihin ja tarjoaa hyvät mahdollisuudet jatkokehitykselle.

Avainsanat: LabVIEW, TestStand, taajuusmuuttaja, suorituskyky, automaattinen, testaus, virtuaali-instrumentti

STADIA
HELSINGIN AMMATTIKORKEAKOULU

ABSTRACT

Name: Erno Pentzin

Title: Automated Test Environment for Performance of Frequency Converters

Date: April 19, 2006 Number of pages: 54 + 3 appendixes

Department: Electrical Engineering Study Programme: EMC and Automation Engineering

Instructor: Mr. Marko Uusitalo, senior lecturer

Supervisor: Mr. Jussi Rantanen, B. Sc.

This graduate study was carried out for the product development department of Product AC profit centre of ABB Oy, Drives in Helsinki. The purpose of this study was to develop an automated test environment for testing the performance of frequency converters. The performance tests of ABB's frequency converters have never been automated before. The tests have been made manually and they have been very time-consuming. The objective was to substantially reduce the time needed to execute the tests and to analyze the results. The aim of this study was not to make the actual performance tests, but to develop the automated test environment, i.e. a test bench, that was to be used for the execution of the performance tests. The focus was on the performance of speed and torque controllers of the frequency converter.

This study was executed as design and programming work. The hardware of the test environment is based on the existing test places of the product development laboratories at ABB. Mainly three phased asynchronous motors are used in the test environment in addition to frequency converters. Additionally, the hardware consists of an ACS800 series frequency converter as a load drive, a torque transducer and a tachometer. The programming was carried out with the LabVIEW development environment version 8.0 from National Instruments Corporation. The same company's TestStand test software version 3.5 was used as a user interface and test executive for the test environment. Virtual instruments were programmed for controlling the frequency converters and for reading the torque transducer. Virtual instruments are used in the TestStand test sequences. The test sequences are created in the TestStand's sequence editor and are executed in the sequence editor or in an operator interface.

This study was successful in creating an automated test environment for testing the performance of frequency converters. The test environment can be used for testing both current and the next generation frequency converters. It is possible to execute the general performance tests of frequency converters, such as the static and dynamic performance of speed and torque controllers, extensively in the test environment. The tests can be made automatically in the whole range of speed and torque allowed by the test drive. The sampling rate can be 1 kHz at the most when reading the speed through ACS800 series frequency converter and the torque through torque transducer at the same time. The test sequences consisting of virtual instruments can be freely modified to further develop the tests or to create new tests. The test environment is based on software that is widely used in the industry. It offers good possibilities for further development.

Keywords: LabVIEW, TestStand, frequency converter, performance, automatic, testing, virtual instrument

SISÄLLYS

LYHENTEET

ABB	Asea Brown Boveri; johtava sähkövoima- ja automaatioteknologiayhtymä
API	Application Programming Interface; sovellusohjelmointikäyttöliittymä
BNC	Bayonet Nut Coupling; erityisesti mittausinstrumenteissa yleisesti käytetty liitin, jossa on lukitusmekanismi
CSV	Comma Separated Values; tiedostoformaatti, jossa tiedot erotetaan pilkuilla
DDCS	Distributed Drives Communication System; ABB:n ACS800-sarjan taajuusmuuttajien käyttämä tiedonsiirtoprotokolla, jota käytetään valokuituyhteydellä
DLL	Dynamic Link Library; Windowsissa käytetty funktiokirjasto
DTC	Direct Torque Control; ABB:n kehittämä suora momentinsäätötekniikka
DUT	Device Under Test; testattava laite
EUT	Equipment Under Test; testattava laite/laitteisto
GPIB	General Purpose Interface Bus; IEEE 488 -standardin mukainen rinnakkaisliitäntä, jota käytetään antureiden ja ohjelmoitavien laitteiden liittämiseen tietokoneeseen
HBM	Hottinger Baldwin Messtechnik; momenttianturivalmistaja
IEC	International Electrotechnical Commission; sähköalan standardointijärjestö
INU	Inverter Unit; vaihtosuuntausyksikkö
NI	National Instruments; mittalaite- ja ohjelmistovalmistaja
PCI	Peripheral Component Interconnect; PC-tietokoneiden laajennuskorttiväylä
PCMCIA	Personal Computer Memory Card International Association; standardointijärjestö, joka kehitti ja standardoi kannettavien tietokoneiden PC-laajennuskortit
RPM	Revolutions Per Minute; pyörimisnopeuden yksikkö (kierroksia minuutissa)

RS-232	Recommended Standard-232; standardi sarjamuotoiseen tiedonsiirtoon tietokoneen ja oheislaitteen (modeemi, hiiri, jne.) välillä
RTF	Rich Text Format; tekstin muotoiluja sisältävä tiedostoformaatti
SQL	Structured Query Language; yleisesti käytetty kieli tietokantojen käsittelyyn
USB	Universal Serial Bus; yleinen tietokoneissa ja oheislaitteissa käytetty sarjaväylä, jonka versioiden 1.0 ja 1.1 suurin tiedonsiirtonopeus on 12 Mbps ja version 2.0 480 Mbps
UUT	Unit Under Test; testattava yksikkö
VI	Virtual Instrument; LabVIEW-sovellus (virtuaali-instrumentti)
VISA	Virtual Instrument Software Architecture; standardoitu ohjelmointirajapinta eri liitännöillä, kuten esimerkiksi GPIB-, USB- tai sarjaväylällä, varustetun laitteiston ja kehitysympäristön, kuten LabVIEW, välillä
XML	EXtensible Markup Language; tiedonkuvauskieli

1 JOHDANTO

Tässä insinöörityössä kehitetään taajuusmuuttajien suorituskyvyn automaattinen testausympäristö. Työ on tehty ABB Oy, Drivesin Product AC -tulosyksikön tuotekehitysosastolle Helsingissä.

ABB:n taajuusmuuttajien suorituskykytestejä ei ole aikaisemmin automatisoitu. Testit on tehty käsin ja niiden suorittamiseen ja tulosten käsittelyyn on kulunut paljon aikaa. Automaattisella testauksella pyritään testien suorittamiseen ja tulosten käsittelyyn kuluvan ajan huomattavaan pienentymiseen. Lisäksi testien kattavuutta on mahdollista lisätä.

Työ on toteutettu suunnittelu- ja ohjelmointityönä. Testausympäristö perustuu suorituskykytestit suorittavaan ohjelmistoon ja sen käyttöliittymään. Työssä ei ole tarkoitus tehdä suorituskykytestejä, vaan kehittää automaattinen testausympäristö, jossa suorituskykytestit on mahdollista suorittaa.

Työn alussa esitellään testausympäristössä käytetty laitteisto yleisesti. Testausympäristössä käytetään taajuusmuuttajien lisäksi pääasiassa kolmivaiheisia oikosulkumoottoreita. Lisäksi laitteistoon kuuluu ACS800-sarjan taajuusmuuttaja, momenttianturi ja takometri eli kierrosnopeusmittari. Seuraavaksi kerrotaan ohjelmoinnissa ja käyttöliittymän suunnittelussa käytetystä kehitysympäristöstä. Ohjelmointi tehtiin National Instrumentsin LabVIEW-ohjelmointiympäristössä ja testausympäristön käyttöliittymänä toimii saman yrityksen TestStand-testausohjelmisto.

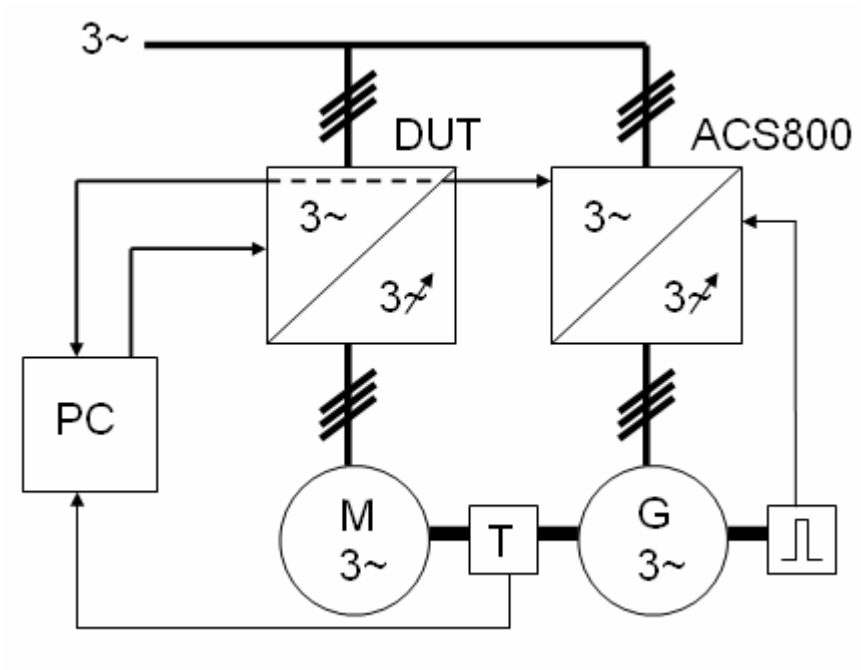
Tämän jälkeen esitellään suorituskykytestejä. Työssä keskitytään taajuusmuuttajan nopeus- ja momenttisäätäjien suorituskykyyn. Kaikki testit, jotka työssä esitellään, tulee olla mahdollista tehdä testausympäristössä eli suorituskykytestipenkissä. Suorituskykytestipenkillä pyritään teettämään yleisimmät taajuusmuuttajien suorituskykytestit, kuten nopeus- ja momenttisäätöjen staattinen ja dynaaminen tarkkuus.

Seuraavaksi esitellään taajuusmuuttajien ohjaukseen tarkoitetut Drive-Debug- ja DTool-funktiot sekä niiden hyödyntämistä testausympäristössä. Lisäksi ohjaukseen käytetään taajuusmuuttajien I/O-liityntöjä USB-väyläisten I/O-moduuleiden kautta. Lopuksi kerrotaan testausympäristöä varten

kehitetyistä virtuaali-instrumenteista ja TestStand-käyttöliittymästä. Testausympäristön ja sen komponenttien toiminnallisuutta ja kehityksen aikana ilmenneitä ongelmia pohditaan.

2 TESTAUSYMPÄRISTÖN LAITTEISTO

Testausympäristön laitteisto perustui olemassaoleviin ABB:n tuotekehityslaboratorioiden testipaikkoihin ja niiden mittalaitteisiin. Lisäksi käytettiin erillisiä tietokoneeseen liitettäviä mittauslaitteita. (Kuva 1.)



Kuva 1. Testausympäristön laitteiston yksinkertaistettu lohkokaavio; testattava taajuusmuuttaja (DUT) kytketään sähköverkon ja moottorin väliin; moottorin kanssa samalla akselilla ovat ACS800-sarjan taajuusmuuttajaan kytketty kuormamoottori sekä momentti- ja pulssianturit; momenttianturia luetaan suoraan tietokoneelta ja pulssianturia luetaan ACS800-taajuusmuuttajan kautta; DUT:tä ja ACS800:sta ohjataan tietokoneelta

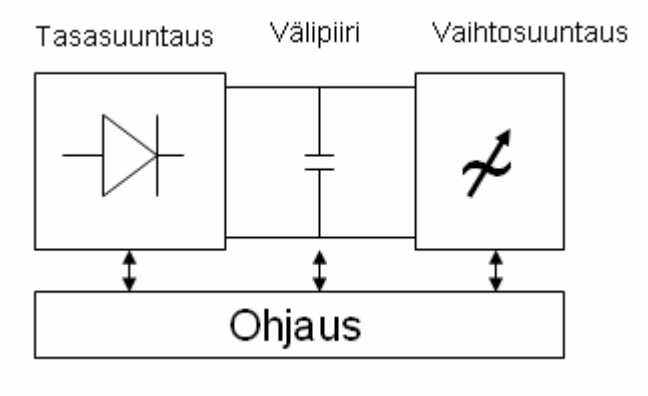
Laitteistoon kuuluu tietokoneen lisäksi ACS800-sarjan taajuusmuuttaja kuormakäyttönä, momenttianturi ja nopeusanturi. Valinnaisena järjestelmään kuului I/O-ohjaus. I/O-ohjausta varten hankittiin ulkoisia tietokoneeseen liitettäviä I/O-moduuleita.

Testattava laite (DUT) voi olla mikä tahansa nykyinen tai seuraavan sukupolven taajuusmuuttaja. Se voidaan kytkeä mihin tahansa moottoriin, jota voidaan testattavalla taajuusmuuttajalla ohjata.

2.1 Taajuusmuuttaja

Taajuusmuuttajalla ohjataan tai säädetään sähkömoottorin pyörimisnopeutta tai momenttia. Nimitys tulee siitä, että vaihtosähkömoottorin pyörimisnopeus riippuu siihen syötettävän sähkön taajuudesta. Aikaisemmin vain sähkön taajuuden säätö oli mahdollista. Nykyään taajuusmuuttajat pystyvät säätämään moottorin pyörimisnopeuden lisäksi vääntömomenttia. Moottorin pyörimisnopeus ja vääntömomentti ovat taajuusmuuttajan tärkeimmät säätösuureet.

Taajuusmuuttaja koostuu neljästä osasta: tasasuuntaajasta (*rectifier*), välipiiristä, vaihtosuuntaajasta (*inverter*) ja näiden kolmen osan toimintaa ohjaavasta ohjausosasta. (Kuva 2.)



Kuva 2. Taajuusmuuttajan yksinkertaistettu lohkokaavio; taajuusmuuttaja koostuu tasasuuntaajasta, välipiiristä, vaihtosuuntaajasta ja näiden kolmen osan toimintaa ohjaavasta ohjausosasta

Taajuusmuuttajan yksinkertaistettu toimintaperiaate on seuraava: Taajuusmuuttajaan sähköverkosta syötetty sinimuotoinen vaihtojännite ensin tasasuunnataan eli muutetaan tasajännitteeksi. Tämän tasajännitteen muoto on kuitenkin vielä epätasainen, joten se syötetään välipiirin kondensaattoreille, jotka tekevät jännitteen muodosta tasaisemman ja varastoivat energiaa. Vaihtosuuntaus muuttaa tasajännitteen halutun taajuiseksi vaihtojännitteeksi, ja vaihtosuunnattu jännite syötetään moottorille.

ABB:n ACS800-sarjan taajuusmuuttajat (kuva 3) ovat tarkoitettu teollisuuskäyttöön. Yksittäiskäyttöjen tehoalue on 0,55 - 2 800 kW. ACS800-sarjan taajuusmuuttajat käyttävät DTC:tä (*Direct Torque Control*) eli suoraa vääntömomentinsäätöä. DTC-tekniikka on maailman kehittynein vaihtovirtakäyttötekniikka.

Kuva 3. ABB:n ACS800-sarjan R2-runkokoon ja IP21-suojausluokan taajuusmuuttaja seinään kiinnitettynä; keskellä näkyvän ohjauspaneelin kautta voidaan ohjata taajuusmuuttajan toimintaa

ACS800-tuoteluettelossa [1, s. 7] mainitut ACS800-taajuusmuuttajien momentinsäädön suorituskykyyn liittyvät arvot ovat seuraavat:

- Momentin nousuaika nimellismomentilla on alle 5 ms takaisinkytkennällä ja ilman.
- Momentin epälineaarisuus nimellismomentilla on ±4 % ilman takaisinkytkentää ja ±1 % takaisinkytkennällä.

Nopeussäädön suorituskykyyn liittyvät arvot ovat seuraavat:

- Nopeussäädön staattinen tarkkuus on 10 % moottorin jättämästä ilman takaisinkytkentää ja 0,01 % nimellisnopeudesta takaisinkytkennällä.
- Nopeussäädön dynaaminen tarkkuus 100 % momentilla on 0,3 - 0,4 %s ilman takaisinkytkentää ja 0,1 - 0,2 %s takaisinkytkennällä.

Kuormaiskun nopeuspoikkeaman pinta-alan yksikkö %s on poikkeaman suhteellisen maksimiarvon (%) ja vasteajan (s) tulo jaettuna kahdella. (Ks. 4.1.2 Nopeussäädön dynaaminen tarkkuus.)

ACS800-taajuusmuuttajan toimintaa ohjataan parametreillä. Parametrejä on yhteensä monta sataa ja ne on jaoteltu ryhmiin. Joitain parametrejä voi vain lukea, kuten esimerkiksi laitteen mittaamia signaaleja. Parametrien arvoja on mahdollista muuttaa ohjauspaneelilta tai tietokoneelta. ACS800-taajuusmuuttaja voidaan liittää tietokoneeseen DDCS-valokuituyhteydellä (*Distributed Drives Communication System*), kunhan taajuusmuuttajassa on RDCO-moduuli ja tietokoneessa vastaava laajennuskortti.

ACS800-taajuusmuuttajille on saatavilla erilaisia ohjelmistoja. Laitteen ohjelmisto määrittää sen käyttämät parametrit ja niiden oletusarvot. Ohjelmisto ja sen versio on otettava testauksessa huomioon.

Tämän työn testausympäristöllä on tarkoitus testata ainoastaan alle 1 000 V:n taajuusmuuttajia. Muita alle 1 000 V:n ABB:n taajuusmuuttajasarjoja ovat muun muassa ACS550 ja ACS350.

2.2 Vaihtosähkömoottori

Tässä testausympäristössä käytetään pääasiassa kolmivaiheisia oikosulkumoottoreita (kuva 4). Oikosulkumoottori on epätahtikone, eli kuormitettuna sen pyörimisnopeus poikkeaa tahtinopeudesta jättämän verran. Se on yleisin teollisuudessa käytetty moottorityyppi muun muassa edullisuuden ja vähäisen huoltotarpeen ansiosta.

Kuva 4. Kolmivaiheinen oikosulkumoottori, jonka nimellisteho on 7,5 kW; akselille on kiinnitetty kuorma

Moottorin tehon P ja vääntömomentin T yhdistää kaava

$$T = \frac{P}{\omega} = \frac{P[W]}{2\pi n[1/s]} \approx 9550 \cdot \frac{P[kW]}{n[RPM]}[Nm] \quad (1)$$

ω on akselin kulmanopeus ja n on pyörimisnopeus. Moottorista saatava vääntömomentti on riippuvainen moottorin fyysisestä koosta. Mitä isompi moottori, sitä suurempi sen tuottama vääntömomentti on. Tässä työssä momentti, vääntömomentti ja kuorma tarkoittavat samaa asiaa.

Moottorin tahtinopeudelle n_s pätee kaava

$$n_s = \frac{f_s}{p}[1/s] \quad (2)$$

f_s on koneen nimellinen syöttötaajuus ja p on napapariluku. Tyypillinen tahtinopeus 50 Hz:n nimellistaajuuden ja napapariluvun p = 1 omaavalle moottorille on 3 000 RPM (*Revolutions Per Minute*), napapariluvun p = 2 moottorille 1 500 RPM ja napapariluvun p = 3 moottorille 1 000 RPM. Absoluuttinen jättämä(nopeus)

$$\Delta n = n_s - n \quad (3)$$

n_s on tahtinopeus ja n on nimellisnopeus. Nimellisnopeus eroaa tahtinopeudesta jättämän verran nimelliskuormalla. [2, s. 14 - 30.]

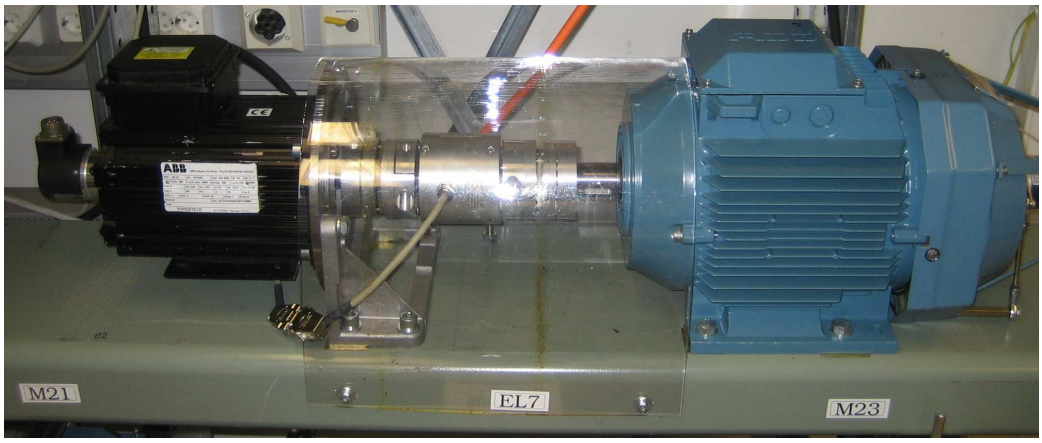
Moottorin pyörimisnopeuden ollessa alle nimellisnopeuden moottori toimii vakiovuoalueella, jossa teho kasvaa nopeuden funktiona nimellisnopeuteen eli kentänheikennyspisteeseen asti. Kentänheikennysalueella eli nimellisnopeutta suuremmilla nopeuksilla syöttöjännite pidetään vakiona (vakiotehoalue) ja magneettivuota heikennetään.

Moottori on mahdollista kytkeä tähti- tai kolmiokytkennällä. Tähtikytkennässä moottorin käämien toiset päät kytketään yhteen eli tähtipisteeseen ja käämien toiset päät kytketään vaiheille. Tällöin moottori kestää enemmän jännitettä, mutta vähemmän virtaa. Kolmiokytkennässä moottorin käämit kytketään sarjaan ja vaiheet kytketään käämien väliin. Tällöin moottori käyttää enemmän virtaa, mutta kestää vähemmän jännitettä.

Sähkömoottorikäyttö koostuu sähkömoottorista ja sen toimintaa ohjaavasta laitteesta, joka tässä työssä on aina taajuusmuuttaja. Sähkömoottorikäytön tarkoitus on muuntaa sähkömagneettista energiaa mekaaniseksi energiaksi.

2.3 Kuormituskoneikko

Testipaikka koostuu kuormituskoneikosta (kuva 5) ja taajuusmuuttajasta. Kuormituskoneikolla tarkoitetaan samalle akselille sarjaan kytkettyjä kahta tai useampaa konetta eli moottoria, joista yksi moottori toimii kuormana eli generaattorina. Kuormamoottori eli kuormakone on kytketty taajuusmuuttajaan, mitkä yhdessä muodostavat kuormakäytön. Muut moottorit kytketään testattavaan laitteeseen eli taajuusmuuttajaan. Moottoreiden kanssa samalla akselilla voi olla myös momenttianturi ja yksi tai kaksi takometriä.



Kuva 5. Tyypillinen testauksessa käytetty kuormituskoneikko, jossa on kaksi moottoria ja niiden väliselle akselille kytketty momenttianturi; testattava taajuusmuuttaja ohjaa vasemmanpuoleista moottoria, ja oikeanpuoleinen moottori kuuluu testattavaa taajuusmuuttajaa kuormittavaan käyttöön

Moottori, joka kytketään testattavaan laitteeseen, ei saa olla tehokkaampi kuin kuormamoottori. Muuten testikäytön moottoria ei voida kuormittaa tarpeeksi. Yleensä kuormamoottori on hieman tehokkaampi kuin testattavaan laitteeseen kytkettävä moottori.

Tarkoitus on, että kannettavassa tietokoneessa olevan suorituskykytestausohjelmiston kanssa voidaan mennä mille tahansa testipaikalle, jossa on kuormituskoneikko ja taajuusmuuttajat, ja suorittaa suorituskykytestit. Testipaikan laitteistossa on oltava ACS800-kuormakäyttö, mutta testattava laite ja siihen kytketty moottori voivat olla mitkä tahansa. Taajuusmuuttajat kytketään kannettavaan tietokoneeseen.

2.4 Takometri

Takometri on kierrosnopeusmittari eli pyörimisnopeusanturi. Taajuusmuuttajakäytössä takometri koostuu yleensä moottorin kanssa samalle akselille kiinnitetystä pulssianturista, ja pulssianturin lähettämiä pulsseja lukevasta laitteesta, joka pulssien perusteella laskee moottorin pyörimisnopeuden ja välittää tiedon taajuusmuuttajalle.

Pulssianturi eli inkrementtianturi lähettää pulsseja, joiden taajuus on suoraan riippuvainen moottorin akselin pyörimisnopeudesta. Pulssiantureita on saatavilla eri pulssilukumäärillä. Yleinen pulssien lukumäärä on 1 024 pulssia yhtä kierrosta kohti. Yleensä pulssiantureissa on pulssien lähettämistä varten kaksi kanavaa (A- ja B-kanavat), joiden ainoa ero on pulssien vaihe. Vaihe-eron avulla saadaan selville akselin pyörimissuunta. Pyörimissuunnasta riippuen joko A- tai B-kanavan pulssit ovat 90 astetta edellä toisten kanavan pulsseja. Lisäksi pulssianturissa voi olla myös nollakanava, joka lähettää yhden pulssin yhtä kierrosta kohti.

Kuva 6. Etualalla moottorin akseliin kiinnitetty Leine & Linden inkrementtianturi

Tässä testausympäristössä moottorin pyörimisnopeuden takaisinkytkentään käytetään aina pulssianturia, kuten esimerkiksi Leine & Linden 861007455-mallista inkrementtianturia (kuva 6). Se tuottaa 1 024 pulssia yhtä akselin kierrosta kohti. Moottorin pyörimisnopeuden ollessa esimerkiksi 1 500 RPM (25 s^{-1}) pulssianturin tuottamien pulssien taajuus on 25,6 kHz, 3 000 RPM:llä (50 s^{-1}) pulssien taajuus on 51,2 kHz ja 6 000 RPM:llä (100 s^{-1}) pulssien taajuus on 102,4 kHz.

Pulssianturin liittämiseksi taajuusmuuttajaan on eri optiomoduuleita, esimerkiksi ACS800-sarjan taajuusmuuttajille RTAC-01- ja RTAC-03-moduulit. Paras mahdollinen moottorin pyörimisnopeudensäätö saavutetaan takometrin avulla, kun taajuusmuuttaja saa tiedon moottorin akselin todellisesta pyörimisnopeudesta. Muuten joudutaan käyttämään moottorimalliin perustuvaa estimoitua eli arvioitua pyörimisnopeutta, joka kuitenkin on monessa sovelluksessa täysin riittävä nykyisten kehittyneiden moottorimallien ja säätömenetelmien ansiosta.

Kun taajuusmuuttajaa käytetään takometrin kanssa, puhutaan takaisinkytketystä säädöstä (*closed loop control*), ja ilman takometriä puhutaan takaisinkytkemättömästä säädöstä (*open loop control*) eli ohjauksesta. Takometri toimii siis moottorin kierrosnopeuden takaisinkytkentänä.

2.5 Momenttianturi

Momenttianturi mittaa voiman momenttia. Momentti on vääntövaikutusta kuvaava suure. Momentin yksikkö on Nm eli newtonmetri.

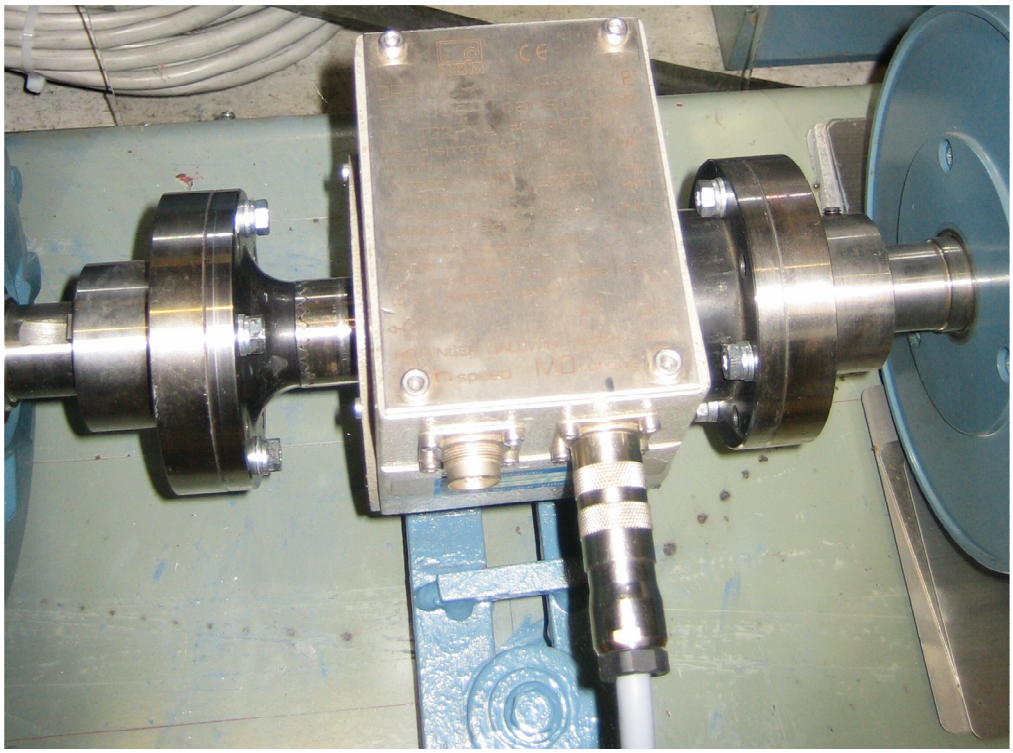
Tässä testausympäristössä tullaan käyttämään pääasiassa HBM:n (*Hottinger Baldwin Messtechnik*) T32FNA- ja KTR:n Dataflex 22 -sarjan momenttiantureita.

2.5.1 HBM:n T32FNA-momenttianturi

T32FNA-momenttianturissa käytetään Wheatstonen siltaan kytkettyjä venymäliuskoja. Venymäliuskat ovat sijoitettu akselille, johon momentti vaikuttaa. Momentin vaikutuksesta akselin muoto muuttuu. Muodonmuutokset ovat hyvin pieniä, mutta kuitenkin havaittavia. Venymäliuskan pituuden muuttuessa sen resistanssi muuttuu ja tämä resistanssin muutos on sähköisesti mitattavissa. Venymäliuska siis muuttaa mekaanisen muutoksen elektronisen resistanssin muutokseksi. Venymäliuskoja on neljä kappaletta. Yksi

venymäliuskapari havaitsee venymän momentin positiiviseen suuntaan ja toinen negatiiviseen suuntaan.

T32FNA-momenttianturi (kuva 7) antaa momenttitiedon taajuusviestinä. Kuormittamattomana taajuus on 10 kHz. Vääntömomentin suunnasta riippuen taajuus on 5 tai 15 kHz nimellismomentilla. Momenttisignaalin jännitetaso on 12 $V_{p\text{-}p}$ (*peak-to-peak*). Momentin mittaustarkkuus on 0,3 %, ja lämpötilan vaikutus on 0,1 %/10 K. Anturista saadaan myös pyörimisnopeus. Pyörimisnopeusliitännästä saadaan kaksi signaalia, joiden vaihe-ero on 90 astetta. Vaihe-eron avulla saadaan selville pyörimissuunta. Kanttimuotoisten signaalien jännite on 25 $V_{p\text{-}p}$, ja taajuus on 15 pulssia yhtä kierrosta kohti. Moottori kierrosnopeuden ollessa esimerkiksi 6 000 RPM momenttianturin pyörimisnopeussignaalien taajuus on 1,5 kHz. [3, s. 9 - 10 ja 20.]

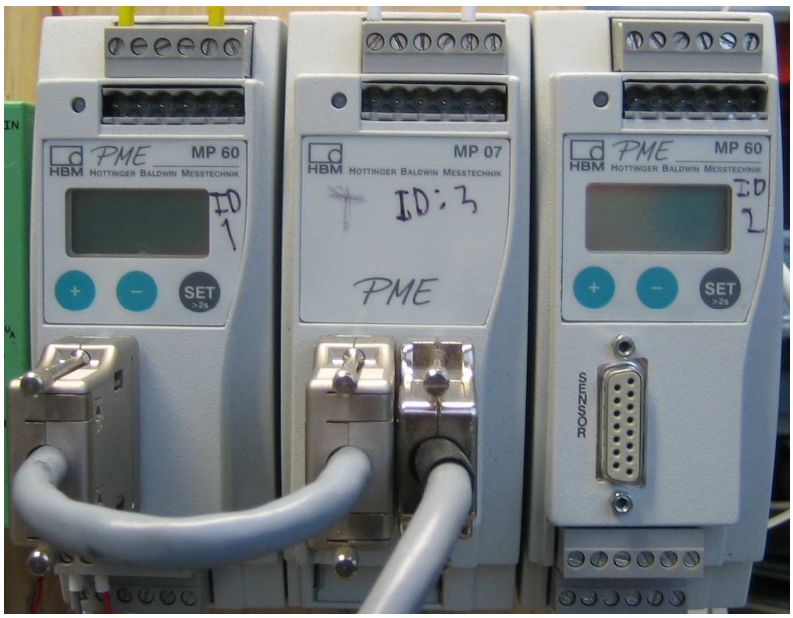


Kuva 7. HBM:n T32FNA-momenttianturi kahden moottorin väliselle akselille kiinnitettynä; vasemmanpuoleisesta liittimestä saadaan pyörimisnopeussignaalit ja oikeanpuoleisesta momenttisignaali

T32FNA-momenttianturi voidaan liittää HBM:n MGC-järjestelmän MC60-signaalinvahvistimeen (kuva 8). MC60:ssä on BNC-liitin (*Bayonet Nut Coupling*), jonka kautta saadaan momenttitieto ±10 V:n jänniteviestinä. T32FNA-momenttianturi voidaan kalibroida MGC-järjestelmän etupaneelin kautta. [4.]

Kuva 8. HBM:n MGC-järjestelmä, johon on liitetty MC60-signaalinvahvistin (BNC-liitin); T32FNA-momenttianturi voidaan kalibroida etupaneelin kautta

T32FNA-momenttianturi voidaan liittää myös HBM:n MP07-moduulin kautta MP60-moduuliin (kuva 9). MP07 antaa anturille sen tarvitseman käyttöjännitteen, ja MP60:n kautta luetaan momentti- tai pyörimisnopeus jännite- tai virtaviestinä. MP60:n kautta saatava jänniteviesti on ±10 V ja virtaviesti on 4 - 20 mA. Koska T32FNA-momenttianturissa on kaksi erillistä lähtöä, joista toinen on momentille ja toinen pyörimisnopeudelle, ja MP60-moduulissa on vain yksi sisääntulo, momenttia ja pyörimisnopeutta ei voida lukea samanaikaisesti MP60:n kautta. T32FNA-momenttianturi voidaan kalibroida MP60:n etupaneelin kautta. [5, s. 14 - 16.]



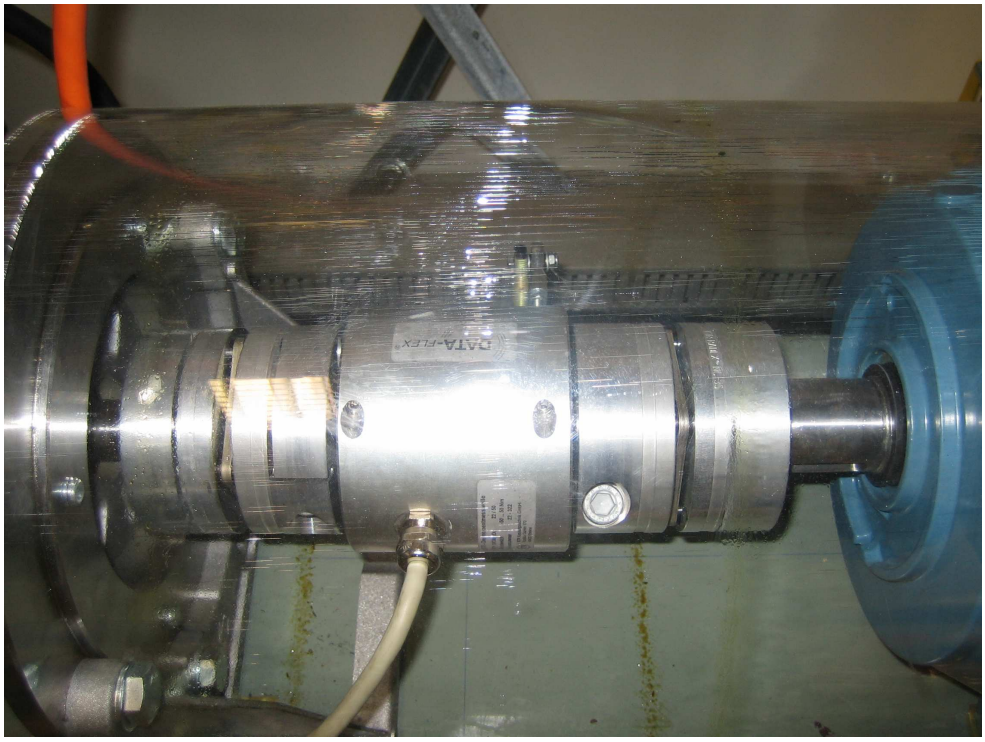
Kuva 9. HBM:n MP60- ja MP07-moduulit; keskeltä lähtevä johto menee T32FNA-momenttianturille, joka voidaan kalibroida MP60-moduulin etupaneelin kautta

2.5.2 KTR:n Dataflex 22 -momenttianturi

KTR:n Dataflex-vääntömomentin mittausakselit perustuvat valon määrän mittaamiseen. Valo johdetaan kahden himmenninkiekon läpi. Himmenninkiekkojen läpi pääsevän valon määrä riippuu vääntömomentista.

Dataflex 22 -momenttianturista (kuva 10) saadaan momenttitieto jännite- ja virtaviestinä. Momentista ja sen suunnasta riippuva lähtöjännite on 0 - 10 V ja virtaviesti on 4 - 20 mA. Kuormittamattomassa tilanteessa lähtöjännite on 5 V ja lähtövirta on 12 mA. Anturin nimellismomentilla momentin mittaustarkkuus on 1 % ja lämpötilan vaikutus on 0,05 %/°K.

Anturissa on myös pyörimisnopeuslähtö. Lähdöstä tulee 24 V:n pulsseja, joiden taajuus on 60 pulssia yhtä kierrosta kohti. Jos moottorin pyörimisnopeus on esimerkiksi 6 000 RPM, anturin tuottamien pulssien taajuus on 6 kHz. [6; 7.]



Kuva 10. Läpinäkyvän suojamuovin alla oleva Dataflex 22/50 -momenttianturi kahden moottorin väliselle akselille kiinnitettynä kummallakin puolella olevien liittimien välityksellä; edestä lähtevä johto menee Dataflex DF1 -kytkentäkotelolle

Dataflex 22 -momenttianturi voidaan liittää Dataflex DF1 -kytkentäkoteloon, jonka kautta voidaan lukea sekä momentti että pyörimisnopeus samanaikaisesti. Kytkentäkotelon lähtösignaalit ovat samat kuin anturin tuottamat signaalit. Momenttisignaalit voidaan suodattaa kytkentäkotelon

alipäästösuodattimella, jonka suodatustaajuus on aseteltavissa 1, 10, 100, 1 000 tai 15 000 Hz:iin. Lisäksi DF1-kytkentäkotelossa on painonapit kalibrointia varten. Momenttianturiin voidaan asettaa automaattinen offset-kompensointi (*automatic offset alignment*) päälle kuormittamattomassa tilassa pitämällä T1-painonappi pohjassa kahden sekunnin ajan. Tällöin momentin arvo nollataan ja kompensoinnin arvo talletetaan. [7.]

Jokaisesta momenttianturista saa momenttitiedon jänniteviestinä ulos. Jänniteviestin lukemiseen tietokoneelta tarvitaan tietokoneeseen liitettävä analoginen tulo.

2.6 USB-väyläiset I/O-moduulit

Momenttianturia ja I/O-ohjausta varten tietokoneeseen on liitettävä I/O-moduuleita. Analogista tuloa käytetään pääasiassa momenttianturin lukemiseen. Sen on kyettävä lukemaan momenttianturista tulevaa jännitesignaalia. Digitaalisia lähtöjä käytetään antamaan testattavalle laitteelle komentoja. Lähtöjen jännitetasojen on sovittava taajuusmuuttajien digitaalisiin I/O-liitäntöihin. Tietokoneeseen liitettävien analogisten ja digitaalisten tulojen ja lähtöjen vaatimukset määriteltiin seuraavasti:

- vähintään 8 kpl digitaalisia 24 V:n tuloja
- vähintään 8 kpl digitaalisia 24 V:n lähtöjä
- 4 kpl analogisia -10 - +10 V:n tuloja
- 4 kpl analogisia -10 - +10 V:n lähtöjä, jotka pystyvät syöttämään virtaa 4 - 20 mA.

USB-väyläiset (*Universal Serial Bus*) laitteet valittiin siksi, että USB-moduulit on helppo kytkeä ja irrottaa sekä niiden fyysinen koko on pieni, joten ne on helppo kuljettaa kannettavan tietokoneen mukana. Mitään erillistä I/O-korttia esimerkiksi PCI-väylään (*Peripheral Component Interconnect*) ei standardimalliseen kannettavaan tietokoneeseen ole mahdollista liittää. Saatavilla on myös PCMCIA:n (*Personal Computer Memory Card International Association*) standardin mukaisia PC-kortteja, mutta niitä ei ole mahdollista suoraan käyttää tavallisessa pöytätietokoneessa. Vain USB-väyläiset laitteet toimivat suoraan kannettavassa tietokoneessa ja pöytäkoneessa. Lisäksi ainakin National Instrumentsin tarjoamien PC-korttien ominaisuudet olivat rajoittuneemmat kuin USB-väyläisillä moduuleilla, joten vaikka PC-kortteja olisi

hankittu, niiden lisäksi olisi silti pitänyt hankkia myös USB-väyläisiä moduuleita.

National Instrumentsin tarjoamat USB-väyläiset I/O-moduulit ovat suoraan LabVIEW:ssä tuettuja. National Instrumentsilla ei kuitenkaan vielä ollut tarjota USB-väyläistä 4 - 20 mA virtaa antavaa analogista lähtöä. Tästä syystä päädyttiin käyttämään ACS800:n RMIO-kortin (moottorin ohjaus- ja I/O-kortti) analogista lähtöä. Sen tarkkuus on vain 10 bittiä. Sitä käytetään paremman puutteessa siihen asti, kunnes tarpeeksi virtaa syöttävä USB-väyläinen analoginen lähtö löytyy joltakin valmistajalta.

NI USB-9215A (kuva 11) on USB 2.0 -väyläinen analoginen tulomoduuli, jossa on neljä kanavaa samanaikaisella näytteenotolla (*simultaneous sampling*). Kanavissa on BNC-liittimet ja niihin saa suoraan esimerkiksi oskilloskoopin mittapäät kiinni. Näin laitteella voidaan helposti toteuttaa esimerkiksi nelikanavainen oskilloskooppi. Suurin näytteenottotaajuus on 100 kS/s (*kilo-Samples/second*, tuhatta näytettä sekunnissa) yhtä kanavaa kohti. Kanavien mittaama jännitealue on -10 - +10 V 16 bitin resoluutiolla (0,0015 %), joten pienin havaittava jännitteen muutos on noin 0,31 mV mitta-alueen ollessa 20 V. Jännitemittauksen tarkkuus on 0,6 % kalibroimattomana 20 - 30 °C:n lämpötilassa. 9215-analogiatulomoduuli on toimitettava National Instrumentsille kalibrointia varten. [8, s. 11 - 16.]

Kuva 11. National Instrumentsin USB 2.0 -väyläinen 9215-analogiatulomoduuli, jossa on neljä kanavaa BNC-liittimillä; jokaista kanavaa varten on oma AD-muunnin, joten näytteet saadaan kaikista kanavista samanaikaisesti

Samanaikaisen näytteenoton etu on siinä, että eri kanavien näytteisiin ei tule viivettä toisiinsa nähden, koska kaikkien kanavien näytteet otetaan samalla hetkellä. Perinteisellä multiplekseriä käyttävällä kanavointimenetelmällä jokainen kanava yhdistetään AD-muuntimeen (*Analog/Digital*) vuorotellen, ja tästä yhdistämisestä aiheutuu eri kanavien näytteiden välille pieni viive. Samanaikaista näytteenottoa käytettäessä jokaista kanavaa varten on oma AD-muunnin.

NI USB-9421 on USB 2.0 -väyläinen digitaalinen tulomoduuli, jossa on kahdeksan kanavaa. Kanavat toimivat 11 - 30 V jännitteellä ja kuluttavat enintään 7 mA virtaa kanavaa kohti. [9, s. 9.]

NI USB-9472 on USB 2.0 -väyläinen digitaalinen lähtömoduuli, jossa on kahdeksan kanavaa. Kukin kanava voi syöttää enintään 30 V jännitettä ja 0,75 A virtaa. Virransyöttöä varten 9472-moduuliin on kytkettävä ulkoinen virtalähde, koska USB-väylä pystyy syöttämään virtaa enintään vain 0,5 A. Virtalähteen lähtöjännite voi olla 6 - 30 V. Virtalähde kytketään VSUP- ja COM-liittimiin. [10, s. 6 - 11.]

National Instrumentsin USB-väyläisiä I/O-moduuleita ohjataan LabVIEW:n DAQmx-funktioilla.

3 TESTAUSYMPÄRISTÖN OHJELMISTON KEHITYSTYÖKALUT

Testausympäristön ohjelmistolla oli muutamia vaatimuksia. Ohjelmiston komponenttien tuli olla uudelleenkäytettäviä ja joustavia, sekä ohjelmointilähdekoodin helposti ylläpidettävää ja luettavaa. Lisäksi versionhallintaa tuli hyödyntää. Tavoitteena oli, että testitapaukset ovat mahdollisimman geneerisiä, joustavia ja helposti ylläpidettäviä. Kaikki testit tuli olla mahdollista suorittaa millä tahansa pyörimisnopeudella testikäytön sallimalla pyörimisnopeusalueella. Tulosten tuli olla hallittavissa, ja suorituskykytestipenkillä tuli olla dokumentaatio. Suorituskykytestipenkin näytteenottotaajuuden tuli olla vähintään 200 Hz.

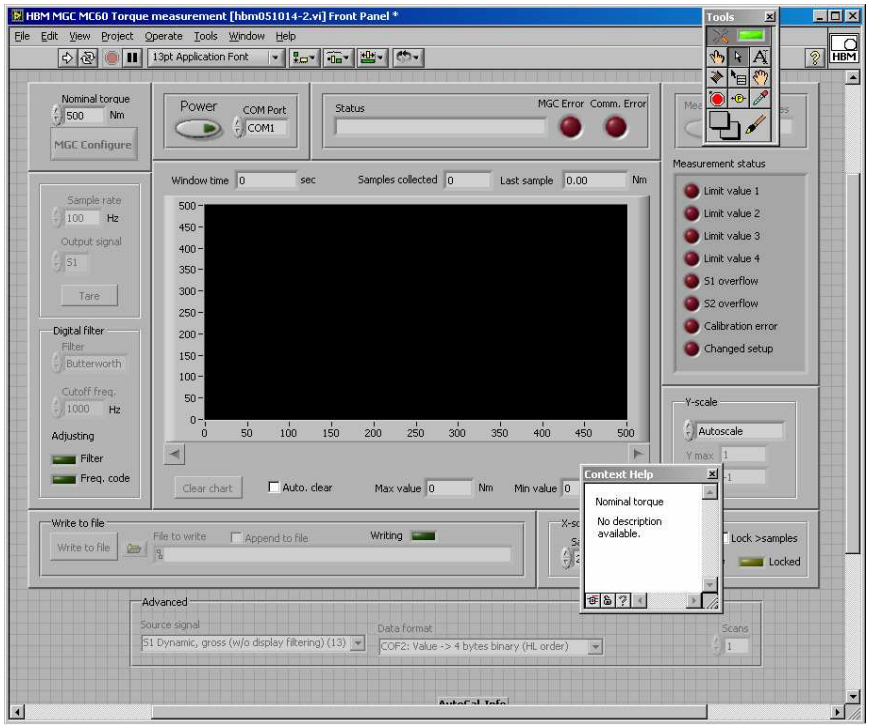
Työn tekemiseen käytettiin National Instrumentsin LabVIEW-ohjelmointiympäristön versiota 8.0 ja TestStand-testausympäristön versiota 3.5. LabVIEW:n lisäksi toinen vaihtoehto ohjelmointikieleksi oli C++. Sen etuna on parempi suorituskyky, mutta muuten mittausten suorittaminen olisi ollut hankalampaa kuin LabVIEW:llä. Kehitystyö päätettiin toteuttaa LabVIEW:llä ja

käyttää C++:aa vain siinä tapauksessa, jos näytteenottotaajuus ei olisi tarpeeksi suuri LabVIEW:llä. Lisäksi LabVIEW-ohjelmoinnin helppous ja nopeus puolsivat sen käyttöä.

Ohjelmointityön määrää vähensi National Instrumentsin TestStand-testausohjelmiston valitseminen testausympäristön käyttöliittymäksi. Minimaalisesti TestStandin toiminnallisuutta vastaavan käyttöliittymän olisi voinut alusta asti kehittää myös LabVIEW:llä. Siihen olisi tarvittu vähintään yhtä paljon aikaa ja työtä kuin testausympäristön LabVIEW-sovelluksien tekoon käytettiin. TestStand tarjoaa paljon ominaisuuksia ja on teollisuudessa yleisesti käytetty testausohjelmisto muun muassa tuotannon sovelluksissa.

3.1 LabVIEW-ohjelmointiympäristö

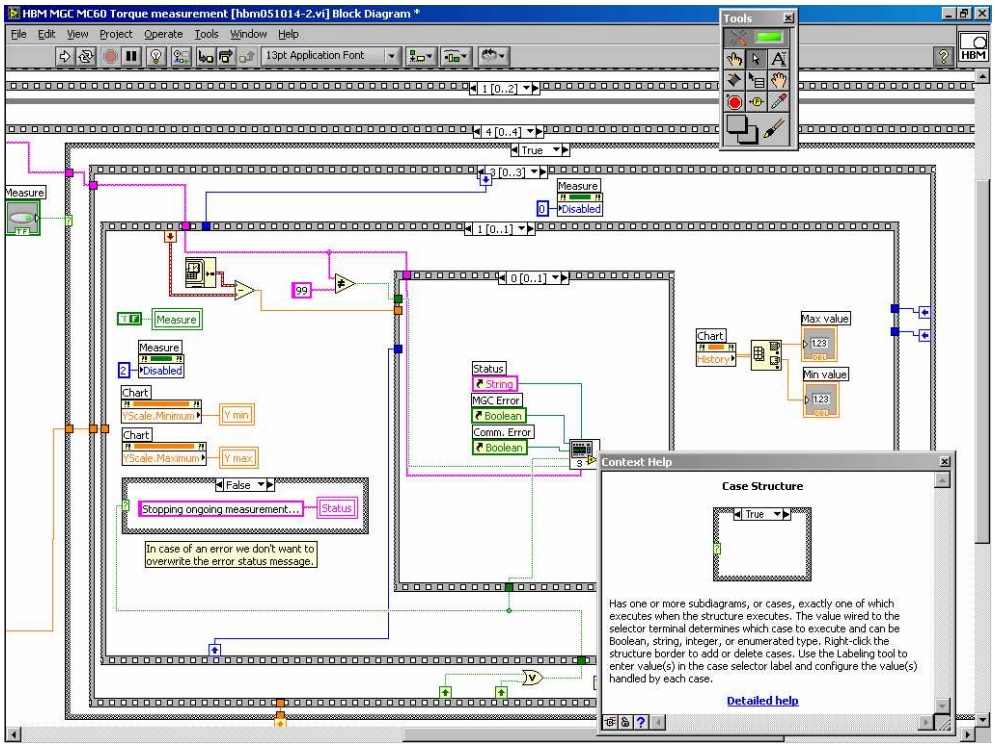
National Instrumentsin LabVIEW on graafinen ohjelmointiympäristö. LabVIEW on kehitetty erityisesti signaalien mittausta ja käsittelyä varten. LabVIEW-sovelluksia kutsutaan virtuaali-instrumenteiksi (*Virtual Instrument*, VI). Virtuaali-instrumentteja voidaan myös kutsua toisista virtuaali-instrumenteista, jolloin kutsuttavista virtuaali-instrumenteista käytetään nimitystä ali-VI (*SubVI*).



Kuva 12. Erään LabVIEW-sovelluksen etupaneeli (näkyvissä on myös työkaluvalikko sekä ohjeikkuna (context help)); etupaneeli koostuu pääasiassa kytkimistä, merkkivaloista ja muista graafisista elementeistä, kuten esimerkiksi keskellä ikkunaa olevasta Waveform Chart -objektista, joka piirtää mittaustulokset graafisena käyränä

LabVIEW-sovellukset ohjelmoidaan LabVIEW:n ohjelmointikielellä, jota kutsutaan G-kieleksi. G-kieli on graafinen ohjelmointikieli. Lähdekoodi ei ole tekstimuotoista, vaan se koostuu lohkoista ja niiden välisistä johtimista. Graafisen ohjelmoinnin ansiosta LabVIEW:llä saa nopeasti ja helposti tehtyä sovelluksia. LabVIEW-sovellus koostuu etupaneelista (*front panel*) ja lohkokaaviosta (*block diagram*). Kuvassa 12 on esimerkkinä erään sovelluksen etupaneeli.

Etupaneeli toimii sovelluksen käyttöliittymänä. Etupaneeli voi sisältää kytkimiä, painonappeja, merkkivaloja sekä muita graafisia elementtejä ja Windowsin käyttöliittymäkomponentteja. Tulona toimivaa objektia, jonka arvoa voidaan muuttaa, kutsutaan kontrolliksi (*control*). Lähtönä toimivaa objektia, joka esittää tietoa, kutsutaan indikaattoriksi (*indicator*). Sovelluksen toiminnallisuus ohjelmoidaan lohkokaavioon (kuva 13).



Kuva 13. Erään LabVIEW-sovelluksen lohkokaavio (näkyvissä on myös työkaluvalikko sekä ohjeikkuna); lohkokaavio sisältää sovelluksen lähdekoodin, joka koostuu funktiolohkoista ja niiden välisistä johdotuksista sekä ohjelman suoritusta ohjaavista rakenteista

Lohkokaavio muistuttaa piirikaaviota. Lohkot eli funktiot ovat yhdistetty viivoilla eli johtimilla toisiinsa. Lohkojen yhdistämistä kutsutaan johdottamiseksi (*wiring*). Yhdessä lohkossa voi olla monta tulo- ja lähtöliitäntää.

Lohkot suoritetaan perinteisistä ohjelmointikielistä poiketen virtauskaavioperiaatteella [11, s. 10] (*dataflow*). Lohko suoritetaan, kun sen kaikkien tulojen tieto on saatavilla. Perinteisissä tekstipohjaisissa ohjelmointikielissä ohjelman suoritus tapahtuu koodiriveittäin ylhäältä alas. LabVIEW-sovelluksessa lohkojen sijainti ei vaikuta millään tavalla suoritusjärjestykseen, vaan siihen vaikuttaa vain lohkojen välinen johdotus. Lisäksi ohjelman suoritukseen vaikuttavat rakenteet (*structure*). Eniten käytettyjä rakenteita ovat muun muassa sekvenssirakenne (*sequence structure*), valintarakenne (*case structure*) ja silmukkarakenne (*for loop* ja *while loop*).

Graafisen luonteensa ansiosta LabVIEW-ohjelman lähdekoodia on helppo lukea. Tosin laajojen ohjelmien lähdekoodi ei mahdu kerralla ruudulle vaan ruutua joudutaan vierittämään, ja näkyvissä voi olla kerrallaan vain osa lähdekoodista LabVIEW:n käyttämien rakenteiden takia.

LabVIEW:llä ohjelman kehitys on erittäin nopeaa verrattuna perinteisiin ohjelmointikieliin. Vaikka LabVIEW on graafinen ohjelmointikieli, se on kuitenkin täysiverinen ohjelmointikieli. LabVIEW:llä voi tehdä kaikki samat asiat kuin muillakin ohjelmointikielillä, ja monesti nopeammin ja helpommin. Funktioita on paljon ja kaikkiin tarkoituksiin.

Yhdessä VI:ssä voi olla yhteensä enintään 28 kappaletta tulo- ja lähtöliitäntöjä eli portteja. Jos portteja tarvitaan enemmän, rajoitus voidaan kiertää käyttämällä ryhmiä (*cluster*). Ryhmiä käyttämällä yhden portin läpi voidaan viedä monta eri parametriä. Näin VI:lle voidaan syöttää lähes rajattomasti parametrejä.

Tietokoneessa, jossa halutaan suorittaa LabVIEW:n virtuaali-instrumentteja, on oltava LabVIEW Run-Time -suoritusympäristö asennettuna. Virtuaali-instrumenttien muokkaamiseen tarvitaan lisäksi LabVIEW-kehitysympäristö.

3.2 TestStand-testausohjelmisto

Testausympäristön käyttöliittymänä toimii National Instrumentsin TestStand. Testejä luodaan ja muokataan TestStandin sekvenssieditorilla. TestStandin operaattorin käyttöliittymässä valitaan testit ja käynnistetään niiden suoritus. Suorituksen päätyttyä kaikista suoritetuista testeistä esitetään raportti, josta selviää muun muassa, menivätkö testit läpi vai ei.

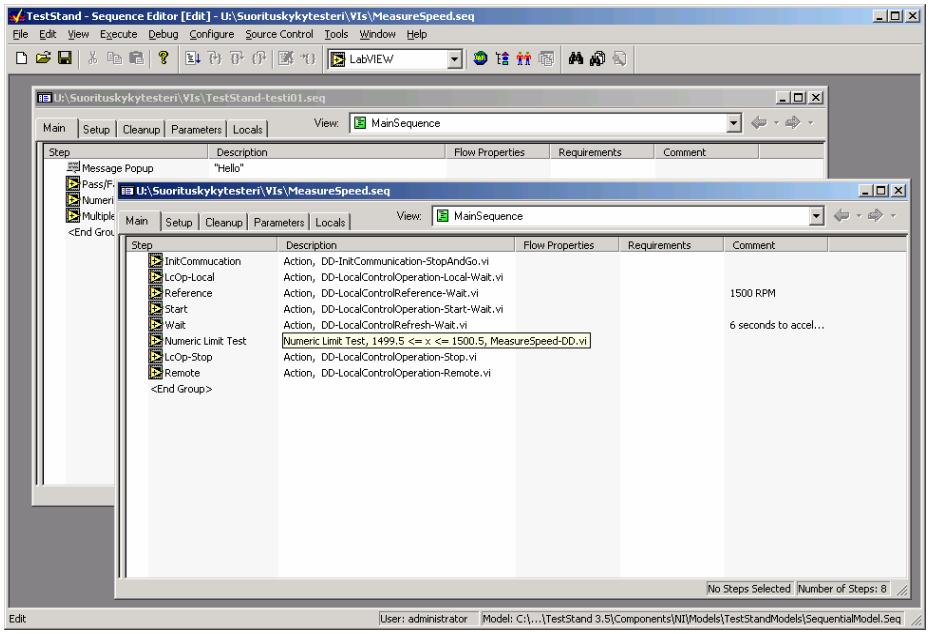
Testit muodostuvat sekvensseistä (*sequence*). Sekvenssi on sarja askelia (*step*), jotka määritellään suoritettavaksi tietyssä järjestyksessä. Askeleen suoritus voi riippua edellisten askelten tuloksista. Askel voi kutsua koodimoduuleita (*code module*) tai suorittaa muita toimintoja. Koodimoduuli on ohjelmamoduuli, joka sisältää yhden tai useamman funktion, joka suorittaa tietyn testin tai toiminnon.

Alisekvenssi (*subsequence*) on toisen sekvenssin kutsuma sekvenssi. Alisekvenssikutsu määritellään askeleeksi kutsuvaan sekvenssiin. Tiedostoa, joka sisältää yhden tai useamman sekvenssin määrittelyn, kutsutaan sekvenssitiedostoksi. Sen tiedostonimen pääte on SEQ.

TestStandin ydin on *test executive* -moottori. Se koostuu useasta moduulista, jotka tarjoavat ohjelmointikäyttöliittymän (*Application Programming Interface*, API) sekvenssien luomiseen, editoimiseen, suorittamiseen ja virheiden etsimiseen ja korjaukseen. Sekvenssieditori ja operaattorin käyttöliittymä käyttävät *test executive* -moottorin palveluita. [12.]

3.2.1 Sekvenssieditori

Sekvenssit määrittelevät suoritettavat testit ja muut toiminnot sekä niiden suoritusjärjestyksen. Sekvenssejä luodaan ja muokataan TestStandin sekvenssieditorilla (kuva 14).



Kuva 14. TestStandin sekvenssieditori; sekvenssieditorissa on auki kaksi sekvenssiä kukin omassa ikkunassaan, joissa sekvenssien askeleet näkyvät omina riveinään

Sekvenssit koostuvat askelista. Sekvenssin askeleita tarkasteltaessa askel näkyy yhtenä rivinä sekvenssieditorin ikkunassa. Askeleita käytetään pääasiassa koodimoduulien kutsumiseen. Jokaisessa sekvenssissä on oletusarvoisesti kolme askelryhmää (*step group*): Main, Setup ja Cleanup. Setup-askelryhmän askeleet suoritetaan aina ennen Main-ryhmän askelia. Se on tarkoitettu mittalaitteiden alustukseen tai vastaaviin toimintoihin. Cleanup-ryhmän askeleet suoritetaan aina Main-askelryhmän jälkeen. Se on tarkoitettu mittalaitteiden ja tietokoneen välisen yhteyden sulkemiseen tai vastaaviin toimintoihin. Cleanup-askelryhmä suoritetaan vaikka testisekvenssin suorittamisen aikana tapahtuisi virhe.

Testausympäristön suorituskykytestien sekvensseistä kutsutaan aina LabVIEW:llä tehtyjä koodimoduuleita. TestStandissa voi kutsua myös muita kuin LabVIEW:llä tehtyjä koodimoduuleita, kuten C- ja C++-ohjelmointikielillä tehtyjen DLL-kirjastojen (*Dynamic Link Library*) funktioita.

3.2.2 Operaattorin käyttöliittymä

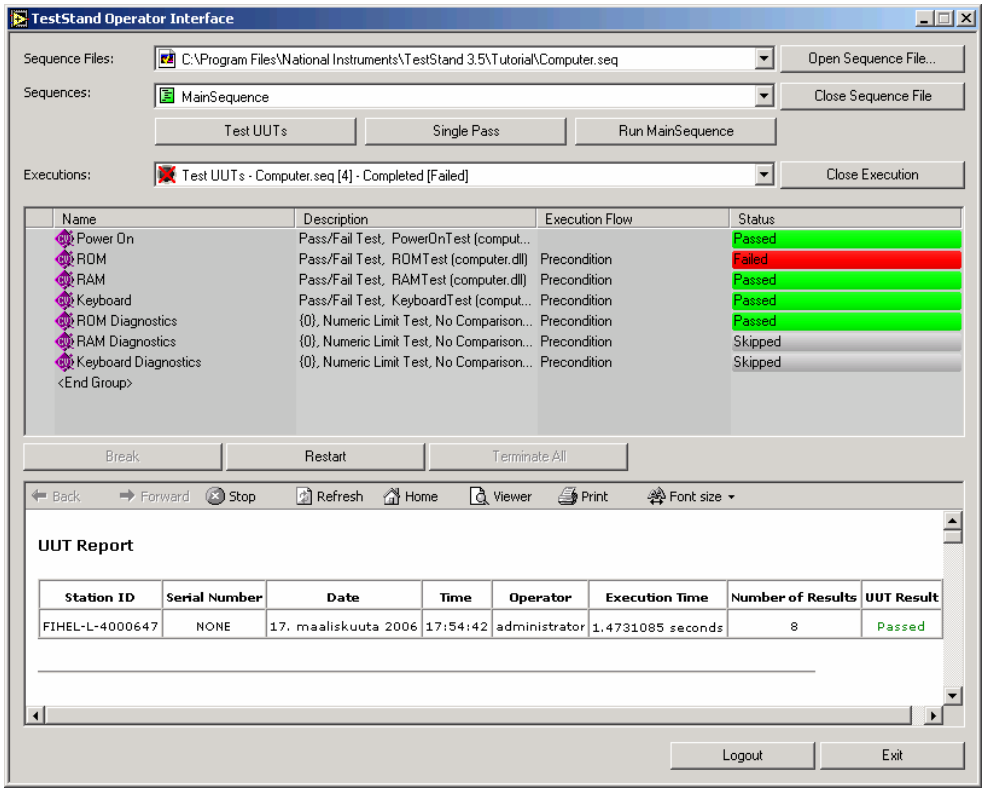
TestStand sisältää valmiita operaattorin käyttöliittymiä. Jokainen operaattorin käyttöliittymä on erillinen sovellusohjelma. Nämä käyttöliittymät ovat täysin muokattavissa.

TestStandin sekvenssieditorin tapaan operaattorin käyttöliittymät mahdollistavat monta samanaikaista suoritusta, keskeytyspisteiden (*breakpoint*) asettamisen ja yhden askeleen kerrallaan suorittamisen. Operaattorin käyttöliittymät eivät kuitenkaan salli sekvenssien muokkausta, eivätkä ne näytä sekvenssin muuttujia (*variables*) tai parametrejä (*parameters*), askeleen parametrejä tai muita yksityiskohtaisia tietoja. Operaattorin käyttöliittymät ovat tarkoitettu käytettäviksi valmiissa testijärjestelmissä. [13.]

Yksinkertaisimmillaan operaattorin käyttöliittymä on esimerkiksi sellainen ikkuna, jossa pyydetään laitteen sarjanumero ja painetaan OK-nappia. Tämän jälkeen TestStand suorittaa etukäteen ohjelmoidut testit ja ilmoittaa operaattorille lopputuloksena ilmoituksen hyväksymisestä tai hylkäyksestä. Tällaisessa käyttöliittymässä käyttäjän ei tarvitse tietää testistä mitään teknisiä ominaisuuksia. Tämäntyyppisiä käyttöliittymiä käytetään muun muassa tuotannon kokoonpanolinjoilla, joissa testit ovat kaikille laitteille suurinpiirtein samat, eikä työntekijöiden tarvitse muokata testejä.

TestStandin mukana tulee kaksi valmista operaattorin käyttöliittymää. Toinen on yksinkertainen ja toinen sisältää enemmän TestStandin ominaisuuksia, mutta siinäkään ei voi muokata sekvenssejä. Sekä yksinkertainen että laajempi operaattorin käyttöliittymä vaativat kirjautumisen eli käyttäjätunnuksen ja salasanan. Käyttäjätunnus määrittelee, mitkä toiminnallisuudet ovat käytettävissä.

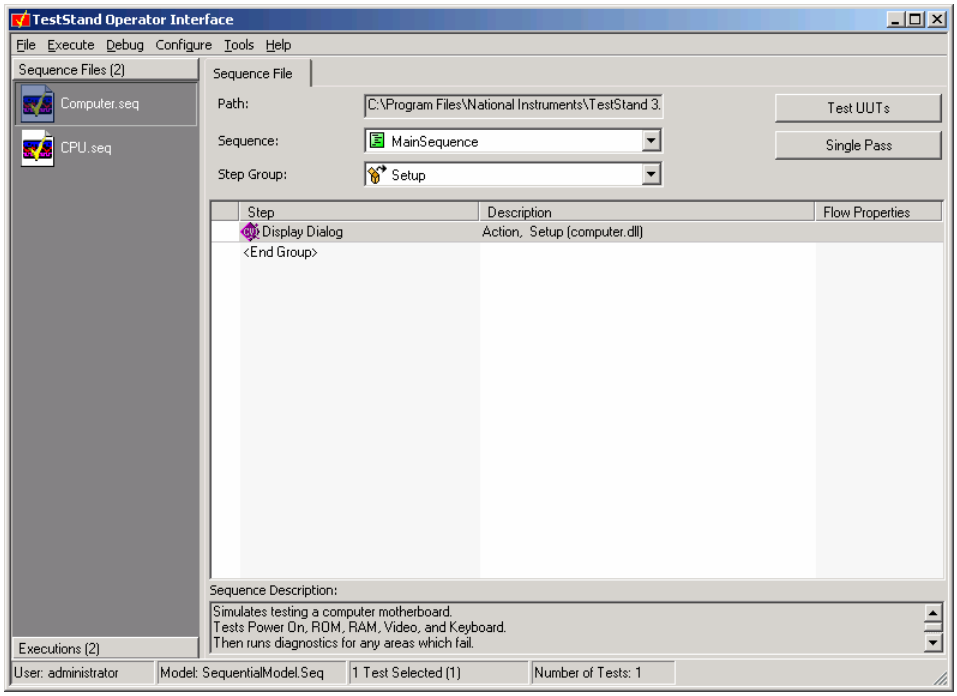
Yksinkertaisessa operaattorin käyttöliittymässä (kuva 15) voidaan avata yksi sekvenssitiedosto kerrallaan. Sekvenssitiedoston avaamisen jälkeen sen sisältämät sekvenssit voidaan ajaa joko monelle testattavalle laitteelle (*Test UUTs, Units Under Test*), yhdelle laitteelle yhden kerran (*Single Pass*) tai suorittaa vain pääsekvenssi (*MainSequence*). Sekvenssin suoritus näkyy askel kerrallaan. Lopuksi näytetään testiraportti, joka voidaan myös tulostaa.



Kuva 15. TestStandin mukana tuleva yksinkertainen operaattorin käyttöliittymä; sekvenssitiedosto ja suoritettava sekvenssi valitaan ikkunan ylälaidasta, askeleet näkyvät ikkunan keskellä, ja raportti suoritetusta sekvenssistä tulostuu ikkunan alalaitaan

Laajemmassa operaattorin käyttöliittymässä (kuva 16) on mahdollista avata monta sekvenssitiedostoa auki. Sekvenssin askelryhmiä voidaan tarkastella erikseen. Vain halutut askeleet voidaan valita suoritettaviksi. Askelia ei kuitenkaan voida muuttaa. Ikkunan ylälaidan alasvetovalikot sisältävät paljon

TestStandin ominaisuuksia. Help-valikon About-kohtaan on mahdollista lisätä yhtiön nimi ja logo.



Kuva 16. TestStandin mukana tuleva laajempi operaattorin käyttöliittymä; vasemmassa laidassa näkyviä sekvenssitiedostoja voi olla useampia auki kerrallaan, suoritettava sekvenssi ja tarkasteltava askelryhmä valitaan ylälaidasta, ja valitun askelryhmän askeleet näkyvät ikkunan keskellä

4 SUORITUSKYKYTESTIT

Taajuusmuuttajan suorituskykytestit ovat tyyppitestejä. IEC:n (*International Electrotechnical Commission*) standardi 61800-4 [14, s. 37] määrittelee käsitteen tyyppitesti seuraavasti:

> Yhden tai useamman tietynmallisen laitteen testi, jolla osoitetaan mallin täyttävän tietyt vaatimukset.

Lisäksi suorituskykytestit ovat järjestelmätestejä. Järjestelmätestauksessa koko järjestelmää eli taajuusmuuttajaa testataan kokonaisuutena.

Tässä työssä suorituskykytestien määrittelyn apuna on käytetty IEC:n standardia 61800-4, vaikka se on keskijänniteluokan (1 - 35 kV) laitteille tarkoitettu. Tässä työssä suorituskykytestien kohteena ovat alle 1 000 V:n taajuusmuuttajat. IEC:n standardi 61800-2 [15] on pienjänniteluokan (alle 1 kV) laitteille, mutta siinä suorituskykyvaatimuksia ei ole määritelty yhtä tarkasti kuin 61800-4-standardissa.

Koska taajuusmuuttajan tärkeimmät säätösuureet ovat moottorin pyörimisnopeus ja vääntömomentti, nopeus- ja momenttisäätäjien suorituskyky ovat keskeisessä asemassa taajuusmuuttajan suorituskykyä mitattaessa. Säädön suorituskyky määritellään siten, miten hyvin säätö seuraa ohjearvoa eli referenssiä. Tässä työssä keskityttiin taajuusmuuttajan nopeus- ja momenttisäätäjien suorituskykyyn.

Säädön ja ohjauksen ero on takaisinkytkentä: ohjauksessa ei käytetä takaisinkytkentää ja säädössä käytetään. Takaisinkytkentä voidaan korvata tarkalla moottorinmallinnuksella, jolloin myös voidaan puhua säädöstä, kuten menetellään momenttisäädöstä puhuttaessa.

Moottorin vääntömomenttia ohjataan eikä säädetä, koska sähkömoottori voidaan matemaattisesti mallintaa melko tarkasti, ja vääntömomenttia ei käytännön sovelluksissa mitata. Ohjauksessa käytetään tyypillisesti kahta mallia, joista ensimmäisen avulla määritetään magneettinen vuo ja vääntömomentti, ja toisen avulla lasketaan optimaalinen ohjaus, jolla nykyisestä tilasta päästään haluttuun tilaan mahdollisimman lyhyessä ajassa. Malleista voidaan saada myös estimaatti eli arvio koneen nopeudelle, jota voidaan käyttää nopeuden oloarvona.

Taajuusmuuttajan säädöistä nopeussäätö on kaikista vaikein. Sen on vaativissa sovelluksissa oltava puhdas säätö eikä ohjaus, eli on käytettävä pyörimisnopeuden takaisinkytkentää. DTC:n avulla saavutetaan moneen sovellukseen riittävä suorituskyky myös ilman pyörimisnopeuden takaisinkytkentää. [2, s. 69 - 70.]

Alla on esitelty yleisiä taajuusmuuttajan nopeus- ja momenttisäätöön liittyviä suorituskykytestejä. Testien kuvauksissa käytetyt pyörimisnopeus- ja momenttiarvot ovat aina testattavan käytön moottoriin verrattavia suhteellisia arvoja. Pyörimisnopeus ilmoitetaan prosentteina testattavan käytön moottorin nimellisnopeuteen nähden. Momentti ilmoitetaan prosentteina testattavan käytön moottorin nimellismomenttiin nähden.

4.1 Nopeussäädön suorituskyky

Nopeudella tarkoitetaan moottorin pyörimisnopeutta. Nopeus voidaan ilmaista taajuutena hertseissä (Hz), kierroksina minuutissa (RPM) tai kierroksina sekunnissa (s^{-1}).

Nopeussäätö tarkoittaa, että moottorin pyörimisnopeutta säädetään. Tällöin moottorin pyörimisnopeus pyritään pitämään mahdollisimman tarkasti ohjearvossa (referenssissä).

Nopeussäädön suorituskykytestien aikana testattavan käytön tulee olla nopeussäädetty ja kuormakäytön momenttisäädetty. Testattavan laitteen nopeussäätäjän on oltava viritetty, koska sillä on suuri merkitys testituloksiin. Testit tehdään takometrin kanssa ja ilman.

4.1.1 Nopeussäädön staattinen tarkkuus

Nopeussäädön staattinen tarkkuus eli vakaan tilan tarkkuus tarkoittaa sitä, miten tarkasti nopeussäätö pystyy tuottamaan nopeusohjetta vastaavan vakaan pyörimisnopeuden. Testi tehdään eri pyörimisnopeuksilla ja eri kuormituksilla.

IEC:n standardi 61800-4 määrittelee vakaan tilan suorituskyvyn (*steady state performance*) sähkömoottorikäytön pyörimisnopeudelle ja momentille poikkeamakaistan (*deviation band*) avulla. Poikkeamakaista tarkoittaa pyörimisnopeuden tai momentin kokonaispoikkeamaa vakaassa tilassa ympäristöolosuhteiden pysyessä määriteltyjen rajojen sisällä. Poikkeamakaista ilmaistaan prosentteina pyörimisnopeuden tai momentin ideaalisesta maksimiarvosta.

Testikäytölle annetaan eri nopeusohjeita ja kuormakäytölle eri momenttiohjeita. Tilan odotetaan vakiintuvan ohjearvojen antamisen jälkeen, minkä jälkeen kerätään mittausdata. IEC 61800-4 -standardin mukaan säätöjärjestelmä on vakaassa tilassa, kun ohjearvo ja toimintamuuttujat (*operating variables*) ovat olleet vakiona pidempään kuin kolme kertaa säätöjärjestelmän asettumisajan eivätkä ympäristöolosuhteet ole muuttuneet. [14, s. 73.]

Mitattavia suureita ovat ainakin akselin pyörimisnopeus ja vääntömomentti. Akselin todellinen pyörimisnopeus mitataan kuormakäytön takometrin kautta ja vääntömomentti suoraan momenttianturilta. Jos mahdollista, DUT:ltä luetaan estimoitu nopeus.

Nopeussäädön staattisen tarkkuuden periaatteellinen testisekvenssi on seuraava:

- DUT: Aseta nopeusreferenssi
- DUT: ”Start”
- ACS800: Aseta momenttireferenssi
- ACS800: “Start”
- ACS800: Lue SPEED MEASURED -parametria
- ACS800: “Stop”
- DUT: “Stop”.

Tuloksissa esitetään nopeusohje, akselin todellinen pyörimisnopeus ja vääntömomentti, sekä mahdollisesti DUT:n estimoima nopeus.

4.1.2 Nopeussäädön dynaaminen tarkkuus

Nopeussäädön dynaaminen tarkkuus eli muuttuvan tilan tarkkuus tarkoittaa sitä, miten tarkasti nopeussäätö pystyy pitämään pyörimisnopeuden ohjearvossa kuormituksen muuttuessa tai miten hyvin nopeusohjeen muuttuessa nopeussäätö kykenee säätämään pyörimisnopeuden nopeusohjetta vastaavaksi.

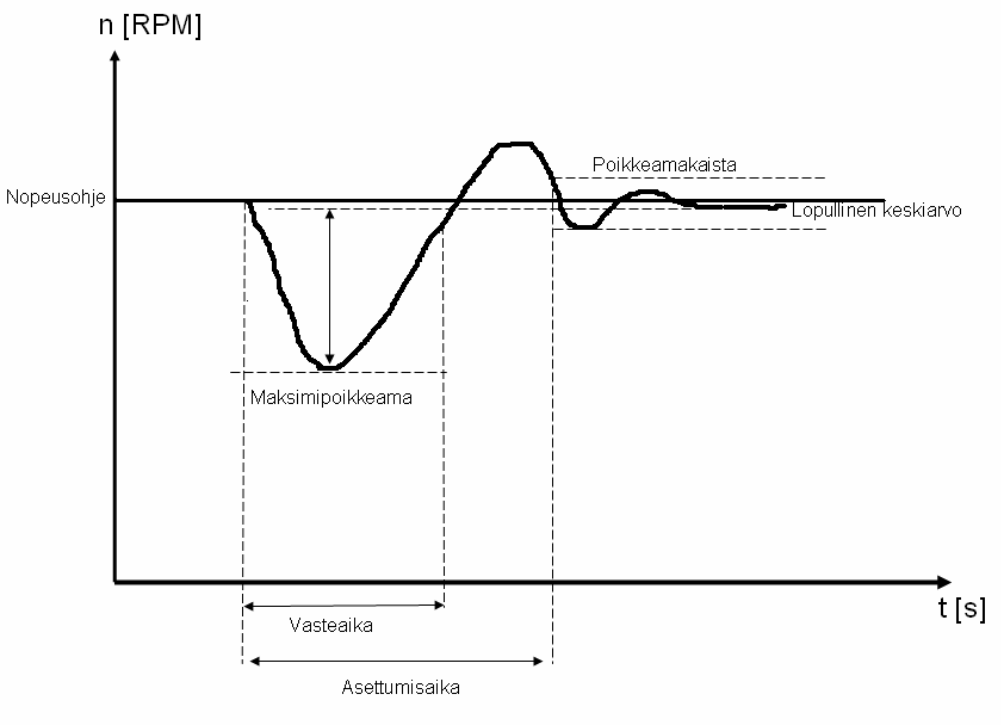
Nopeussäädön dynaaminen tarkkuus määritetään perinteisen askelvastekokeen avulla. Askelvastekoe tehdään erikseen ohjearvoaskelvastekokeena ja häiriösuureen askelvastekokeena. Häiriösuureena on moottorin kuormitus eli vääntömomentti. Häiriösuureen askelvastekokeessa kuormitusta muutetaan askelmaisesti. Ohjearvoaskelvastekokeessa nopeusreferenssiä muutetaan askelmaisesti.

Dynaamisen suorituskyvyn testeissä mittaukset on tehtävä muutoksen kestoajan lisäksi myös muutosta ennen ja jälkeen, jotta muuttuvan tilan mittauksia voidaan verrata muuttumattomaan tilaan. Nopeussäädön dynaamisen tarkkuuden testiä varten kuormakäytön nopeusrajat tulee asettaa 1,5-kertaisiksi testimoottorin nimellisnopeuteen nähden ja momenttirampit nollaan sekuntiin. [16, s. 5.]

Kuorma-askel

Testikäytön tulisi pitää pyörimisnopeus vakiona kuorman äkillisesti muuttuessa eli kuormaiskun (*impact load*) aikana.

IEC 61800-4 -standardin mukaan kuormaiskun nopeuspoikkeaman pinta-alan (*load impact speed deviation area*) perusteella voidaan arvioida nopeussäädön vastetta vääntömomentin muuttuessa äkillisesti (kuva 17). Se lasketaan kertomalla vasteaika (*response time*) ja hetkellisen poikkeaman maksimiarvo (*maximum transient deviation*) keskenään ja jakamalla tulo kahdella. Vasteaika ilmoitetaan sekunneissa ja hetkellisen poikkeaman maksimiarvo prosenteissa pyörimisnopeuden lopulliseen keskiarvoon nähden. Näin ollen kuormaiskun nopeuspoikkeaman pinta-alan yksikkö on prosenttisekunti, %s. [14, s. 79.]



Kuva 17. Kuormaiskun nopeuspoikkeaman pinta-alan laskemiseen käytetyt tunnusluvut; kuormaiskun nopeuspoikkeaman pinta-ala (%s) on maksimipoikkeaman (%) ja vasteajan (s) tulo jaettuna kahdella (maksimipoikkeama lasketaan IEC 61800-4 -standardin mukaan lopullisen pyörimisnopeuden keskiarvon perusteella)

Edellä esitetty kuormaiskun nopeuspoikkeaman pinta-alan laskutapa ”kolmiomenetelmällä” on likimääräinen, koska nopeuden muutokset eivät ole lineaarisia, ja pinta-alan tarkka määrittämän mittaustuloksista on hankalaa. Arviosta aiheutuu hieman virhettä.

Vasteaika tarkoittaa aikaa kuormaiskun alusta siihen asti kunnes pyörimisnopeus saavuttaa arvon, jonka suuruus on pyörimisnopeuden lopullinen keskiarvo lisättynä 10 %:lla hetkellisen poikkeaman maksimiarvosta.

Asettumisaika (*settling time*) tarkoittaa aikaa kuormaiskun alusta siihen asti, kunnes pyörimisnopeus on lopullisen keskiarvon ympärillä olevan poikkeamakaistan sisällä. Kyseisen poikkeamakaistan suuruus on ±5 % hetkellisen poikkeaman maksimiarvosta. [14, s. 77 - 83.]

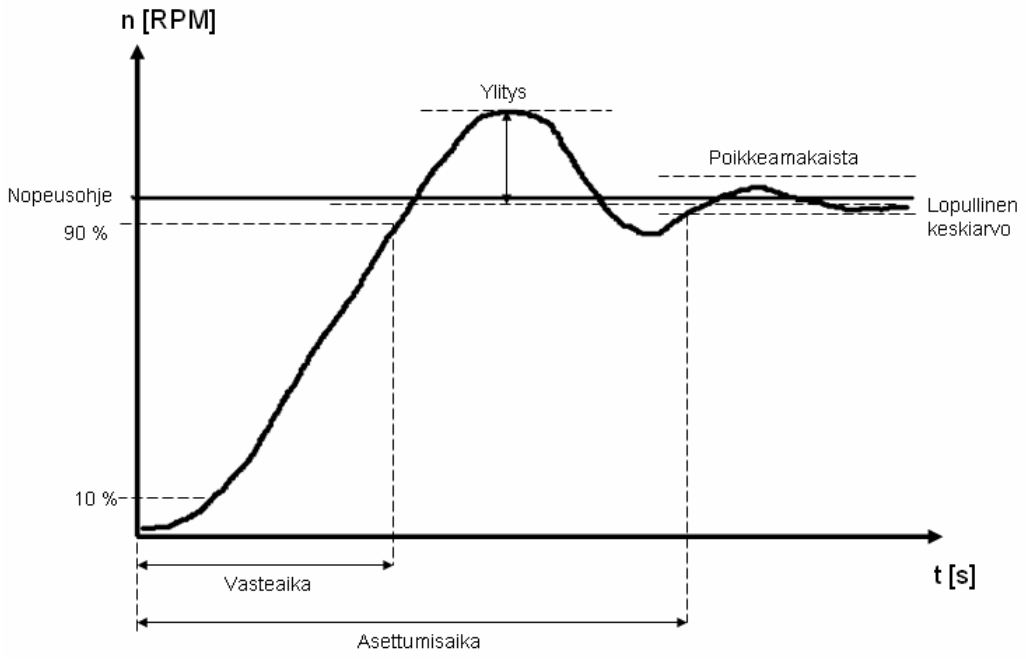
Testikäyttöä ajetaan vakionopeusohjeella ja momenttia muutetaan askelmaisesti kuormakäytöltä. Todellinen momentti mitataan ja pyörimisnopeus luetaan kuormakäytön kautta. Mittausten perusteella lasketaan vasteaika (s), suurin nopeuden virhe (%) ja nopeuspoikkeaman pinta-ala (%s). Lopullisen nopeuden tulee olla staattisen nopeuden tarkkuuden määrittämissä rajoissa. Testi toistetaan useilla eri nopeusohjeilla sekä positiivisilla ja negatiivisilla kuorman muutoksilla. [16, s. 4 - 6.]

Nopeussäädön kuorma-askeleen dynaamisen tarkkuuden periaatteellinen testisekvenssi on seuraava:

- DUT: Aseta nopeusreferenssi
- DUT: ”Start”
- ACS800: Aseta momenttireferenssi
- ACS800: “Start”
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttiantoria
- ACS800: Muuta momenttireferenssi
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttiantoria
- ACS800: “Stop”
- DUT: “Stop”.

Nopeusaskel

Kun nopeusaskel eli ohjearvoaskel annetaan, pyörimisnopeuden pitäisi seurata tätä muutosta parhaalla mahdollisella tahdilla eikä ylittää lopullista ohjearvoa liian paljon (kuva 18).



Kuva 18. Ohjearvoaskelvasteeseen liittyviä tunnuslukuja; ylitys lasketaan lopullisen keskiarvon perusteella, vasteaika on aika askeleen alusta siihen asti kunnes pyörimisnopeus on 90 % nopeusohjeen muutoksen arvosta, ja asettumisaika on aika askeleen alusta siihen asti kunnes pyörimisnopeus on poikkeamakaistan sisällä (poikkeamakaistan suuruus on ±2 % ohjearvosta)

Vasteaika tarkoittaa aikaa nopeusaskeleen alusta siihen asti kunnes pyörimisnopeus on 90 % nopeusohjeen muutoksen arvosta. Ylitys (*transient overshoot*) saa olla enintään 10 % lopullisesta keskiarvosta. Asettumisaika tarkoittaa aikaa nopeusaskeleen alusta siihen asti kunnes pyörimisnopeus on lopullisen keskiarvon ympärillä olevan poikkeamakaistan sisällä. Poikkeamakaistan suuruus on ±2 % ohjearvosta. [14, s. 77 - 81.]

Nopeussäätäjä viritetään parhaaseen mahdolliseen suorituskykyyn nimellisarvoilla (nimellismomentilla nopeuden muuttuessa nollasta nimellisnopeuteen). Testikäyttöä ajetaan vakionopeudella ja sille annetaan nopeuden ohjearvon askel. Nopeuden ohjearvo ja todellinen pyörimisnopeus tallennetaan ja niiden perusteella lasketaan asettumisaika ja ylitys. Testi toistetaan useilla eri nopeuksilla ja kuormilla. [16, s. 2 - 7.]

Nopeussäädön nopeusaskeleen dynaamisen tarkkuuden periaatteellinen testisekvenssi on seuraava:

- DUT: Aseta nopeusreferenssi
- DUT: ”Start”
- ACS800: Aseta momenttireferenssi

- ACS800: “Start”
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttianturia
- DUT: Muuta nopeusreferenssi
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttianturia
- ACS800: “Stop”
- DUT: “Stop”.

4.2 Momenttisäädön suorituskyky

Momenttisäätö tarkoittaa, että moottorin vääntömomenttia säädetään. Moottorin momentti pyritään pitämään mahdollisimman tarkasti ohjearvossa. Tällöin moottorin akselilla on aina oltava kuormaa. Muuten akselille ei synny vääntövaikutusta.

Momenttisäädön suorituskykytestien aikana testikäytön tulee olla momenttisäädetty ja kuormakäytön nopeussäädetty. Ennen mittauksia momenttianturi täytyy kalibroida akselin ollessa paikoillaan ja kuormittamattona.

Momenttisäädön suorituskykytesteissä mitataan aina akselin todellinen vääntömomentti momenttianturin kautta. Myös testattavalta laitteelta voidaan lukea sen näkemystä moottorin momentista, mikäli tämä on mahdollista. Lisäksi monissa testeissä on tarpeen lukea akselin todellista pyörimisnopeutta ACS800-kuormakäytön SPEED MEASURED -parametrin kautta.

4.2.1 Momenttireferenssin lineaarisuus

Momenttireferenssin lineaarisuudella tarkoitetaan taajuusmuuttajan momenttisäädön suorituskykyä momenttiohjeen lineaarisesti muuttuessa (ramppina) tuottaa mahdollisimman tarkasti lineaarinen vääntömomentti akselille.

Momenttireferenssi muutetaan ramppina -150 %:sta 150 %:iin tietyssä ajassa (esimerkiksi 20 s). Testi toistetaan eri pyörimisnopeuksilla. Moottorin nimellisnopeutta suuremmilla pyörimisnopeuksilla momenttireferenssiä on alennettava. Nollanopeudella akselin on oltava lukittuna. [17, s. 3.]

Momenttireferenssin ramppi on manuaalisesti asetettava testattavaan laitteeseen ennen testin aloittamista. Ramppi asetetaan syöttämällä ajat momentin nousulle ja laskulle. Mitatut momenttiarvot esitetään graafisesti ja

niihin sovitetaan suora, jonka perusteella määritetään momenttireferenssin lineaarisuuden virhe.

Momenttireferenssin lineaarisuuden periaatteellinen testisekvenssi on seuraava:

- ACS800: Aseta nopeusreferenssi
- ACS800: ”Start”
- DUT: Aseta momenttireferenssi
- DUT: ”Start”
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttiantoria
- DUT: Muuta momenttireferenssi
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttiantoria
- DUT: ”Stop”
- ACS800: “Stop”.

4.2.2 Momentin lineaarisuus

Momentin lineaarisuus tarkoittaa taajuusmuuttajan momenttisäädön suorituskykyä tuottaa momenttiohjetta vastaava vääntömomentti eri pyörimisnopeuksilla. Moottorin vääntömomentin on vastattava momenttireferenssiä mahdollisimman hyvin huolimatta moottorin eri pyörimisnopeuksista.

Testikäytölle annetaan vakiomomenttireferenssi. Kuormakäyttö muuttaa akselin pyörimisnopeutta ramppina -100 %:sta 100 %:iin tietyssä ajassa (esimerkiksi 50 s). Testi toistetaan eri momenttireferensseillä mahdollisimman suureen momenttiin asti (esimerkiksi 150 %). Momentti mitataan myös nopeusramppia ennen ja jälkeen (esimerkiksi 5 s:n ajalta). [17, s. 3.]

Mitatuista momenttiarvoista lasketaan suhteellinen virhe momenttireferenssiin nähden ja ilmoitetaan virheen keskiarvo, maksimi ja minimi. Mitatut arvot esitetään graafisesti.

Periaatteellinen testisekvenssi momentin lineaarisuudelle on seuraava:

- ACS800: Aseta nopeusreferenssi
- ACS800: ”Start”
- DUT: Aseta momenttireferenssi

- DUT: ”Start”
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttianturia
- ACS800: Muuta nopeusreferenssi
- ACS800+DAQmx: Lue SPEED MEASURED -parametria ja momenttianturia
- DUT: ”Stop”
- ACS800: “Stop”.

4.2.3 Momentin toistettavuus

Momentin toistettavuudella tarkoitetaan momenttisäädön kykyä tuottaa momenttireferenssiä vastaava vääntömomentti toistuvasti samanlaisena.

Momenttireferenssi muutetaan nollasta tiettyyn referenssiin. Tämä tehdään monta kertaa (esimerkiksi 10 kertaa). Jokaisella kerralla momentti mitataan lyhyesti (esimerkiksi 10 näytettä) asettumisviiveen (esimerkiksi 5 - 10 s) jälkeen. Momentin keskiarvo määritellään mittauksen perusteella. Testi toistetaan eri momenttireferenssin arvoilla 150 %:iin asti ja eri pyörimisnopeuksilla (kumpaankin suuntaan). Suurilla pyörimisnopeuksilla momenttireferenssiä on pienennettävä. [17, s. 5.]

Tulokset esitetään taulukoituina eri momenttireferensseillä ja pyörimisnopeuksilla. Mitattua momenttia verrataan momenttireferenssiin, minkä perusteella määritetään suhteellisen virheen keski- ja maksimiarvot sekä varianssi. Momentin suhteelliset virheet esitetään graafisesti, vaaka-akselilla nopeus ja pystyakselilla suhteellinen virhe.

Momentin toistettavuuden periaatteellinen testisekvenssi on seuraava:

- ACS800: Aseta nopeusreferenssi
- ACS800: ”Start”
- DUT: Aseta momenttireferenssi
- DUT: ”Start”
- DAQmx: Lue momenttianturia
- DUT: Muuta momenttireferenssi
- DAQmx: Lue momenttianturia
- DUT: ”Stop”
- ACS800: “Stop”.

4.2.4 Momentin staattinen tarkkuus

Momenttisäädön staattinen tarkkuus tarkoittaa sitä, miten tarkasti momenttisäätö pystyy tuottamaan momenttiohjetta vastaavan vakaan vääntömomentin moottorin akselille.

Testikäytölle annetaan vakiomomenttireferenssi ja momentti mitataan momenttianturilta. Testi toistetaan eri momenttireferensseillä ja pyörimisnopeuksilla (kumpaankin suuntaan).

Mitattua momenttia verrataan momenttireferenssiin, minkä perusteella lasketaan suhteellinen virhe ja varianssi. Tulokset esitetään taulukoituina eri momenteilla ja pyörimisnopeuksilla, sekä graafisesti siten, että vaaka-akselilla on pyörimisnopeus ja pystyakselilla suhteellinen virhe.

Momenttisäädön staattisen tarkkuuden periaatteellinen testisekvenssi on seuraava:

- ACS800: Aseta nopeusreferenssi
- ACS800: ”Start”
- DUT: Aseta momenttireferenssi
- DUT: ”Start”
- DAQmx: Lue momenttianturia
- DUT: ”Stop”
- ACS800: “Stop”.

5 TAAJUUSMUUTTAJIEN OHJELMALLINEN OHJAUS

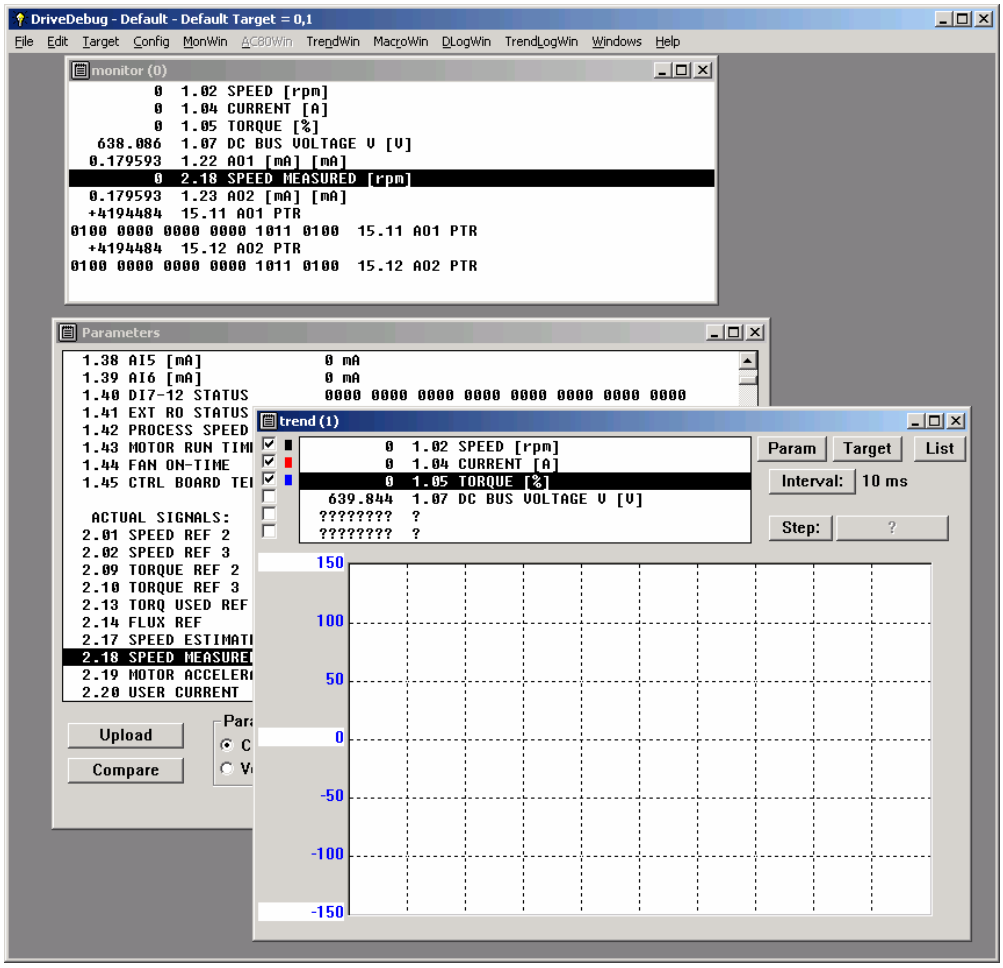
Tietokoneen ja testattavan taajuusmuuttajan välinen yhteys sekä tietokoneen ja ACS800-kuormakäytön välinen yhteys toteutettiin ensisijaisesti DriveDebug- ja DTool-kirjastoilla. Ohjausta varten kehitettiin LabVIEW:n virtuaali-instrumentteja. Niissä hyödynnettiin jo olemassa olevia taajuusmuuttajien ohjelmallisia ohjausmahdollisuuksia.

5.1 DriveDebug-funktiot

ABB:n DriveDebug-ohjelma (kuva 19) on DDCS-protokollaa käyttävien laitteiden ohjaukseen tarkoitettu Windows-sovellus. Ohjelman mukana tulee AMCTVB.DLL-kirjasto, jonka sisältämillä funktioilla voidaan muun muassa

- siirtää parametrilista tietokoneelle

- lukea ja kirjoittaa parametrien arvoja
- siirtää dataloggerin näytteet tietokoneelle ja ohjata dataloggeria
- siirtää vikalista tietokoneelle ja tyhjentää se
- suorittaa paikallisohjausoperaatioita (*local control*). [18, s. 1.]



Kuva 19. DriveDebugin käyttöliittymä; tarkasteltavat parametrit ovat omassa monitorointi-ikkunassaan, ja haluttujen parametrien arvot voidaan esittää graafisesti trendi-ikkunassa

Liitteessä 1 on lista DriveDebugin funktioista ja niiden C-kielisistä prototyypeistä. DriveDebugia ja sovellusta, joka käyttää DriveDebugin funktiokirjastoa, ei voida suorittaa samanaikaisesti. Vain yksi DriveDebug-funktio voidaan suorittaa kerrallaan.

Ensimmäinen suoritettava DriveDebug-funktio tulee olla InitCommunication. Se täytyy suorittaa ennen muiden DriveDebug-funktioiden kutsumista. Mikäli funktio palauttaa muun arvon kuin nolla, DriveDebug-funktiot eivät ole käytettävissä. Viimeinen suoritettava DriveDebug-funktio tulee olla StopCommunication.

Jokaisen DriveDebug-funktiokutsun suorittaminen kestää useita millisekunteja. Poikkeuksena tähän ovat nopeat parametrifunktiot, joita on mahdollista käyttää jopa 1 kHz:n taajuudella (1 ms:n välein) parametrin lukemiseen.

DriveDebug-funktioilla ohjataan ACS800-sarjan taajuusmuuttajaa, joka toimii testausympäristön kuormakäyttönä. Ohjauksessa voidaan käyttää paikallisohjausta tietokoneelta tai paneelilta.

5.1.1 Paikallisohjaus

Ohjattaessa ACS800-taajuusmuuttajaa tietokoneelta paikallisohjaustilassa LocalControlRefresh-funktiota on kutsuttava jatkuvasti vähintään viiden sekunnin välein. Muuten tulee paneelivirheilmoitus (*panel loss*), ja paikallisohjaus menee pois päältä. Tämän jälkeen laite on resetoitava.

Jotta LocalControlRefresh-funktiota voidaan kutsua, virtuaali-instrumentin on oltava jatkuvasti käynnissä. Tämä voi kuitenkin muodostua ongelmaksi joissakin testeissä. Silloin täytyy käyttää paikallisohjausta paneelin kautta.

5.1.2 Paneeliohjaus

Paneeliohjausta käytettäessä ACS800:ssa on oltava ohjauspaneeli kytkettynä. ACS800 luulee, että sitä ohjataan paneelilta, mutta todellisuudessa sitä ohjataan kirjoittamalla tietokoneelta paikallisohjausryhmän (LOCAL CONTROL) parametreihin. Paikallisohjaustila on tällöin paneelilla, ja paneeli hoitaa automaattisesti päivityspyynnöt, jolloin niitä ei tarvitse lähettää tietokoneelta eikä myöskään paneelivirhettä tule.

Paneeliohjausta käytettäessä paneelin on oltava kiinni eikä sitä saa irrottaa missään vaiheessa testin suorituksen aikana. Paneelia ei myöskään saa käyttää testin aikana. Tietokoneella ajettava testiohjelmisto huolehtii ohjauskomentojen antamisesta paneelin kautta ACS800:lle.

Paneelin paikallisohjausta ei kuitenkaan ole mahdollista asettaa tietokoneelta päälle. Tätä varten testaajan on testin aluksi manuaalisesti painettava paneelin LOC/REM-nappia, joka asettaa paneelin paikallisohjaustilaan. Paneelin ollessa paikallisohjaustilassa paneelin näytön ylälaidassa näkyy L-kirjain.

Paikallisohjausryhmän parametreihin kirjoittamalla voi laitteelle antaa kaikki tavalliset paikallisohjauskäskyt, paitsi paikallisohjaustilan asettamisen päälle ja pois. Ohjaamiseen käytetään seuraavia parametrejä:

- LOCAL CONTROL WORD
- LOCAL STATUS WORD
- LOCAL REFERENCE 1 [RPM]
- LOCAL REFERENCE 2 [%].

Komennot annetaan kirjoittamalla parametriin Local Control Word (LCW). Laitteen tila luetaan parametrista Local Status Word (LSW). Nopeusreferenssi annetaan kirjoittamalla parametriin Local Reference 1 [RPM]. Momenttireferenssi annetaan kirjoittamalla parametriin Local Reference 2 [%].

Parametreihin kirjoittaminen ei onnistu DriveDebugin tavallisilla parametrifunktioilla, vaan on käytettävä nopeita parametrifunktioita. Nopeita parametrifunktioita käytettäessä kaikki muut kuin REAL-tyyppiset parametrit ovat INT-tyyppisiä. LCW-parametri on BITSTRING-tyyppinen, joten siihen kirjoitettaessa on käytettävä WriteToIntParam-funktiota.

5.2 DTool-funktiot

ABB:n DTool on tuotekehityskäyttöön tarkoitettu Windows-sovellus ACS550-sarjan ja muille samaa teknologiaa käyttäville taajuusmuuttajille. DToolin käyttöliittymä vastaa ulkoasultaan DriveDebugia. ACS550- ja muiden vastaavien taajuusmuuttajien ohjelmiston toiminnallisuudesta johtuen DToolin ominaisuudet ovat rajoittuneemmat kuin DriveDebugin.

DTool-funktioilla voidaan

- siirtää parametrilista tietokoneelle
- lukea ja kirjoittaa parametrien arvoja. [19, s. 1.]

Liitteessä 2 on lista DTool-funktioista ja niiden C-kielisistä prototyypeistä. DToolia ja DToolin funktiokirjastoa käyttävää sovellusta ei voi suorittaa samaan aikaan. Vain yksi DTool-funktio voidaan suorittaa kerrallaan. DToolin funktiokirjaston kanssa samasta hakemistosta tulee löytyä COMINFO.TXT-tiedosto, joka on kaksirivinen tekstitiedosto. Ensimmäisellä rivillä lukee tietokoneen sarjaportti, johon ACS550-pohjainen taajuusmuuttaja on kytketty (esimerkiksi COM1 tai COM2). Toisella rivillä lukee yhteyden baudinopeus, joka normaalikäytössä on 9 600 baudia.

DTool käyttää tietokoneen sarjaporttia. Normaalisti tietokone ja ACS550-pohjainen taajuusmuuttaja ovat liitetty yhteen tavallisella Ethernet-

verkkokaapelilla ja RJ-45/RS-232-adapterilla. Verkkokaapeli kytketään suoraan taajuusmuuttajan paneeliliitäntään. Tämä RS-232-sarjaporttiyhteys on syynä DTool-funktioiden hitaaseen suoritusnopeuteen DriveDebug-funktioihin verrattuna. DriveDebugin käyttämän DDCS:n yhteysnopeus voi olla useita megabittejä sekunnissa, kun taas RS-232-yhteyden nopeus on yleensä vain joitain kilobittejä sekunnissa.

Ennen muiden DTool-funktioiden käyttöä täytyy suorittaa InitCommunication- ja ModbusConnect-funktiokutsut tässä järjestyksessä. InitCommunication-funktion suoritus kestää useita sekunteja. Viimeinen suoritettava funktio pitää olla StopCommunication.

DTool-funktioiden joukossa ei ole DriveDebug-funktioiden tyyppisiä paikallisohjausfunktioita. DTool-funktioilla voidaan vain lukea ja kirjoittaa parametrejä. Ohjaus voidaan kuitenkin toteuttaa kirjoittamalla ”I/O-ryhmän” parametreihin. I/O-ryhmässä on digitaalisia ja analogisia tuloja vastaavat parametrit, jotka ovat

- DigitalForceMask (digitaalisen I/O:n pakotusvalitsin)
- DigitalForceData (digitaalisen I/O:n pakotusdata)
- AnalogForceMask (analogisen I/O:n pakotusvalitsin)
- tAI1+2 (analogisen tulon 1 pakotusdata)
- tAI2+2 (analogisen tulon 2 pakotusdata).

Parametrin DigitalForceMask bitit 0 - 5 vastaavat digitaalisia tuloja 1 - 6. Tulon tila voidaan pakottaa asettamalla tuloa vastaava bitti päälle. Tila, johon tulo pakotetaan (0 tai 1), asetetaan parametrin DigitalForceData kautta. Parametrin DigitalForceData bitit ovat samat kuin parametrilla DigitalForceMask.

Parametrin AnalogForceMask bitit 0 ja 1 vastaavat analogisia tuloja 1 ja 2. Tulon arvo voidaan pakottaa asettamalla tuloa vastaava bitti päälle. Pakotettava arvo kirjoitetaan parametriin tAI1+2 tai tAI2+2 riippuen siitä, kumpaan tuloon arvo halutaan pakottaa. Arvo voi olla väliltä 0 - 1 000, mikä vastaa tulon arvoa väliltä 0 - 100 %.

5.3 I/O-ohjaus

I/O-ohjauksella tarkoitetaan laitteen ohjaamista sen analogisten ja digitaalisten tulojen kautta, sekä laitteen tilan mahdollista lukemista vastaavien lähtöjen kautta. Testattaessa seuraavan sukupolven taajuusmuuttajaa I/O-ohjaus on ensisijainen ohjausvaihtoehto, koska testilaitteessa ei välttämättä ole muita ohjausvaihtoehtoja käytettävissä tai ne eivät ole yhteensopivia testausympäristön muiden ohjausmahdollisuuksien kanssa.

I/O-ohjaukseen käytetään USB-väyläisiä I/O-moduuleita, joita ohjataan LabVIEW:n DAQmx-funktioilla, ja ACS800-kuormakäytön analogisia lähtöjä, joita ohjataan DriveDebug-funktioilla. Analogiset ja digitaaliset lähdöt kytketään taajuusmuuttajan analogisiin ja digitaalisiin tuloihin. Digitaalisilla lähdöillä annetaan DUT:lle komentoja ja analogisilla lähdöillä ohjearvoja. Testattavan laitteen digitaalisten signaalien taso on oltava 24 V ja analogisten tulojen -10 - +10 V tai 0(4) - 20 mA.

I/O-ohjausta käytettäessä tietokoneen puolelta katsottuna analoginen lähtö toimii referenssin asettamiseksi, analoginen tulo numeerisen tilatiedon (esimerkiksi virta, nopeus tai teho) saamiseksi DUT:lta, digitaalinen lähtö käynnistys-, pysäytys- ja nollauskäskyjen antamiseksi, ja digitaalinen tulo binäärisen tilatiedon (*ready*, *running*, *fault*) lukemiseksi DUT:lta.

5.3.1 ACS800-taajuusmuuttajan I/O-liitännät

ACS800-taajuusmuuttajan RMIO-ohjauskortin I/O-liitännät ovat seuraavat:

- Kolme analogista tuloa: AI1, AI2 ja AI3. AI1 on 0(2) - 10 V jännitetulo, jonka resoluutio on 0,025 % (12 bittiä). AI2 ja AI3 ovat 0(4) - 20 mA virtatuloja, joiden resoluutio on 0,05 % (11 bittiä). Tulojen epätarkkuus on ±0,5 %.
- Kaksi analogista lähtöä: AO1 ja AO2. Ne molemmat ovat 0(4) - 20 mA virtalähtöjä. Lähtöjen resoluutio on 0,1 % (10 bittiä) ja epätarkkuus ±1 %.
- Kuusi digitaalista tuloa: DI1, DI2, DI3, DI4, DI5 ja DI6. Niiden logiikan jännitetaso on 24 V.
- Kolme digitaalista relelähtöä: RO1, RO2 ja RO3. Niihin voidaan RMIO:lta johdottaa 24 voltin signaali. [20, s. 85; 21, s. 73 - 74.]

Analogiatuloa AI1 käytetään nopeusreferenssin asettamiseen ja analogiatuloa AI2 käytetään momenttireferenssin asettamiseen. Molemmat analogiset lähdöt AO1 ja AO2 saadaan parametrien kautta näyttämään haluttua arvoa

tai muuttujaa, kuten moottorin pyörimisnopeutta tai lähtövirtaa. Perusasetuksilla digitaalista tuloa DI1 käytetään moottorin käynnistämiseen ja pysäyttämiseen, ja digitaalista tuloa DI2 pyörimisnopeuden suunnan vaihtoon. Digitaalinen relelähtö RO1 ilmoittaa onko laite valmiustilassa (*Ready*), RO2 ilmoittaa pyöriikö moottori (*Running*) ja RO3 ilmoittaa vikatilan (*Fault*).

ACS800-kuormakäytön analogisten lähtöjen ohjaaminen

ACS800-kuormakäytön analogisia lähtöjä ohjataan DriveDebug-funktioiden kautta. Seuraavia parametrejä käytetään ohjaamaan analogisten lähtöjen AO1 ja AO2 toimintaa:

- ANALOGUE OUTPUT1
- ANALOGUE OUTPUT2
- AO1 PTR
- AO2 PTR.

Parametrille Analogue Output1 on annettava arvoksi 17, ja parametrille Analogue Output2 on annettava arvoksi 16, jotta lähtöjen virran suuruutta voidaan muuttaa suoraan parametreillä AO1 PTR ja AO2 PTR. Lähtöjen virtaa voi muuttaa välillä 0 - 20 mA ja tätä virtaa vastaa luku väliltä 0 - 20 000. Lukuun lisätään 4 194 304 ja summa kirjoitetaan joko parametriin AO1 PTR tai AO2 PTR, riippuen siitä kumman lähdön virran suuruutta halutaan muuttaa.

5.3.2 ACS550-taajuusmuuttajan I/O-liitännät

ACS550-taajuusmuuttajan I/O-liitännät ovat seuraavat:

- Kaksi analogista tuloa: AI1 ja AI2. Molemmat voidaan erikseen asettaa toimimaan joko 0 - 10 voltin jännitetulona tai 0 - 20 mA:n virtatulona. Tulojen resoluutio on 0,1 % ja tarkkuus on ±1 %.
- Kaksi analogista lähtöä: AO1 ja AO2. Molemmat ovat 0 - 20 mA:n virtalähtöjä. Kuormaa saa olla enintään 500 Ω yhtä lähtöä kohden.
- Kuusi digitaalista tuloa: DI1, DI2, DI3, DI4, DI5 ja DI6. Niiden logiikan jännitetaso on 24 V.
- Kolme digitaalista relelähtöä: RO1, RO2 ja RO3. Niiden suurin sallittu kuorma on 2 A. [22, s. 17.]

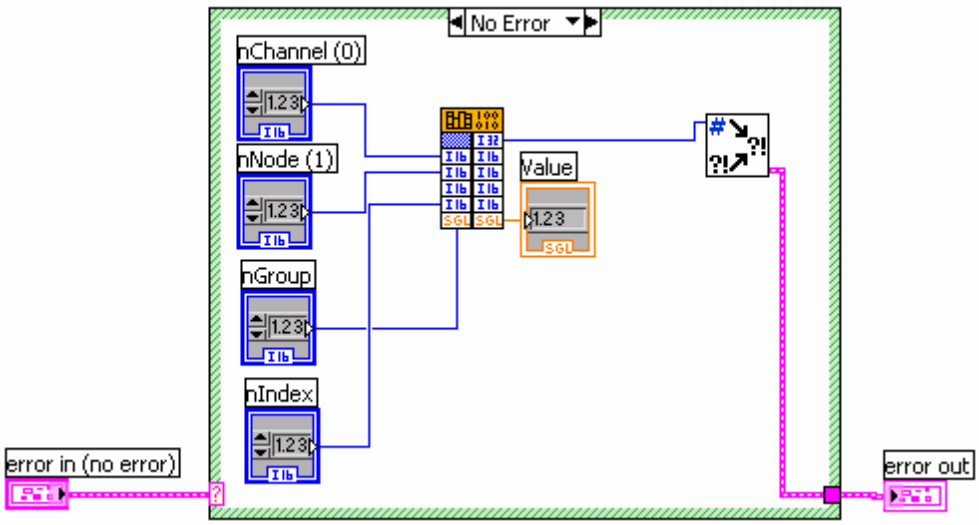
ACS550:n I/O-liitäntöjen toiminta perustilassa on samanlainen kuin ACS800:lla. (Ks. 5.3.1 ACS800-taajuusmuuttajan I/O-liitännät.)

6 TESTAUSYMPÄRISTÖN VIRTUAALI-INSTRUMENTIT

Testit suoritetaan kutsumalla TestStandista LabVIEW:llä tehtyjä virtuaali-instrumentteja. Virtuaali-instrumenteista on tehty mahdollisimman yksinkertaisia ja suurin osa niistä toteuttaa vain yhden funktion.

Testausympäristön virtuaali-instrumenttien pohjana käytettiin SubVI with error handling -VI-mallia (*template*). Tämä VI-malli sisältää jo valmiiksi error in- ja error out -muuttujaryhmät, sekä valintarakenteen. Tämän mallin avulla voidaan tehdä virtuaali-instrumentti, jonka sisältö suoritetaan vain, jos error in -ryhmä on tyhjä, eli virhettä ei ole tapahtunut. Käyttämällä error in - ja error out -ryhmiä virtuaali-instrumenttien ketjutukseen voidaan tehokkaasti hyödyntää virtauskaavio-ohjelmointiperiaatetta. Mikäli jossain ketjun ali-VI:ssä tapahtuu virhe, muut virtuaali-instrumentit eivät suorita funktioitaan, ja tieto virheestä kuljetetaan samantien koko ketjun läpi.

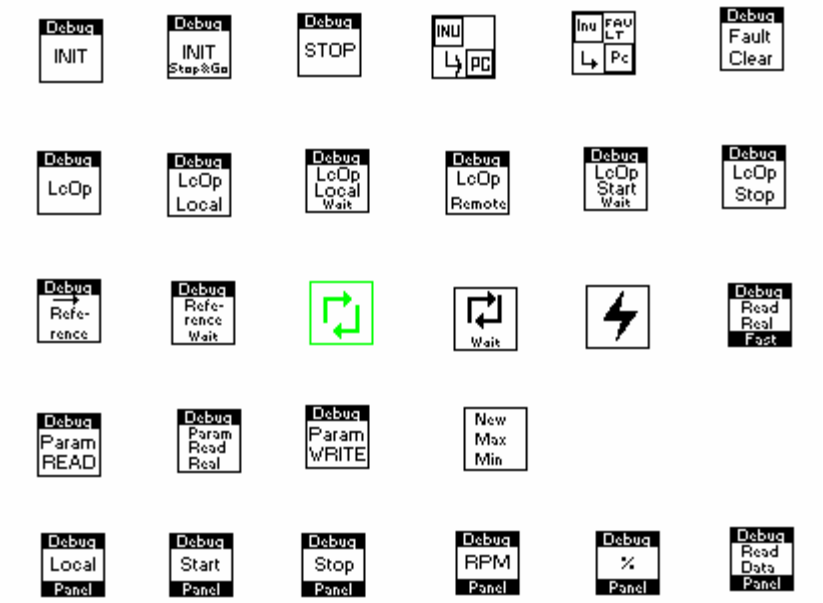
Jokaisesta käytetystä DriveDebug- ja DTool-funktiosta tehtiin oma virtuaali-instrumentti (kuva 20). Näiden virtuaali-instrumenttien rakenne pyrittiin pitämään mahdollisimman yksinkertaisena. Funktiokutsut tehtiin Call Library Function -lohkolla. Funktion palauttama arvo johdotettiin Error Cluster from Error Code -ali-VI:n kautta error out -ryhmään. Virtuaali-instrumentti siis palauttaa suoraan funktiokutsun arvon. Kaikki muut funktion parametrit johdotettiin kontrolleille ja indikaattoreille.



Kuva 20. DriveDebugin ReadFromRealParam-funktiokutsun suorittavan virtuaali-instrumentin lähdekoodi; funktiokutsu tehdään kuvan keskellä näkyvällä, monta tuloa ja lähtöä sisältävällä Call Library Function -lohkolla, ja funktion palauttama arvo on johdotettu Error Cluster from Error Code -ali-VI:n kautta error out -ryhmään

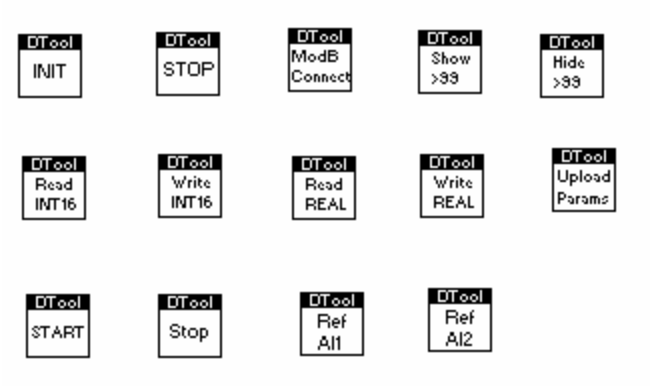
Kaikille virtuaali-instrumenteille piirrettiin yhdenmukaiset ikonit. Ikonista selviää, minkä kirjaston funktioita virtuaali-instrumentti käyttää ja mitä funktiota se kutsuu. Jokaisella ikonilla on yhden pikselin levyinen reunus.

DriveDebugin kirjastofunktioita käyttäviä virtuaali-instrumentteja (kuva 21) on muun muassa erikseen käynnistys- ja pysäytyskomennoille, referenssin asettamiselle, paikallis- ja etäohjaustilan asettamiseen sekä parametrien lukemiselle ja kirjoittamiselle. Näitä virtuaali-instrumentteja käytettäessä on ensin suoritettava DD-InitCommunication.VI, ja viimeiseksi DD-StopCommunication.VI.



Kuva 21. DriveDebugin funktioita käyttävien virtuaali-instrumenttien kuvakkeet; alimmalla rivillä ovat paneeliohjauksessa käytettävät virtuaali-instrumentit

DToolin kirjastofunktioita käyttäviä virtuaali-instrumentteja (kuva 22) on muun muassa parametrien luvulle ja kirjoitukselle. Näitä virtuaali-instrumentteja käytettäessä on ensin suoritettava DT-InitCommunication.VI ja seuraavaksi DT-ModBusConnect.VI. Viimeiseksi on suoritettava DT-StopCommunication.VI.



Kuva 22. DToolin funktioita käyttävien virtuaali-instrumenttien kuvakkeet

I/O-ohjaukseen tarkoitettujen virtuaali-instrumenttien kuvakkeet ovat esitetty kuvassa 23. IO-Start.VI:llä voidaan asettaa mikä tahansa digitaalinen lähtö päälle. IO-Stop.VI:llä vastaavasti voidaan nollata mikä tahansa digitaalinen lähtö. Referenssin asetukseen tarkoitetut virtuaali-instrumentit IO-Reference-AO1.VI ja IO-Reference-AO2.VI käyttävät DriveDebug-funktioita ACS800-taajuusmuuttajan analogisten lähtöjen ohjaamiseen.

Kuva 23. I/O-ohjaukseen tarkoitettujen virtuaali-instrumenttien kuvakkeet

LabVIEW-lähdekoodin luonne mahdollistaa sen, että testeissä käytettäviä valmiita virtuaali-instrumentteja voidaan helposti muokata, mikäli jokin testi sitä vaatii. Liitteessä 3 on kaikkien testausympäristön virtuaali-instrumenttien lähdekoodi. Lähdekoodi on tulostettu LabVIEW:n File/Print-valikon kautta. Print Contents -kohdasta valittiin VI documentation. Tulostettavaksi valittiin kuvake ja liitinpaneeli (*icon and connector pane*), kuvaus (*description*), lohkokaavio (*block diagram*) ja piilotetut ruudut (*hidden frames*). Etupaneeleita ei valittu tulostettavaksi, koska virtuaali-instrumentteja käytetään Test-Standin kautta ilman etupaneelin näyttämistä. Destination-kohdasta valittiin Rich Text Format (RTF) file, jolloin lähdekoodi tallentui tiedostoon.

Lähdekoodin yhteydessä on kaikkien virtuaali-instrumenttien tulojen ja lähtöjen kuvaus sekä lyhyt kuvaus toiminnallisuudesta. Virtuaali-instrumenttien kuvaus on syötetty File/VI Properties -valikon Documentation-ikkunan Description-kenttään. Selkeyden vuoksi liitteen 3 lähdekoodista poistettiin kuvat, joissa esiintyi vain virheen läpivienti ruudun läpi.

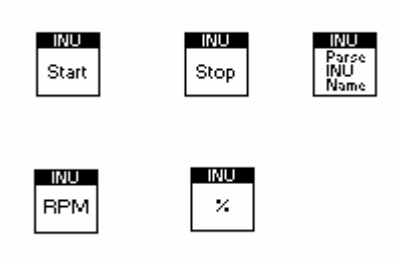
6.1 NI-VISA-tyyliset virtuaali-instrumentit taajuusmuuttajien ohjaukseen

LabVIEW:ssä on VISA-lohkoja (*Virtual Instrument Software Architecture*), jotka ovat tarkoitettu kommunikointiin laitteiden kanssa, joiden yhteys tietokoneeseen voi vaihdella. Yhteys voi olla esimerkiksi GPIB (*General Purpose Interface Bus*), RS-232 tai joku muu. Kuitenkin samat VISA-lohkot toimivat riippumatta laitteen ja tietokoneen välisestä liitännästä. Tätä samaa periaatetta voidaan hyödyntää myös tässä työssä taajuusmuuttajan ja tietokoneen välistä kommunikointia vaativassa LabVIEW-sovelluksessa tai TestStand-testisekvenssissä.

Ongelmana on se, että eri taajuusmuuttajat sisältävät erilaiset kommunikointimahdollisuudet. Esimerkiksi ACS800-sarjan taajuusmuuttajiin voi liittää RDCO-moduulin, jolla saadaan DDCS-valokuituyhteys laitteen ja tietokoneen välille. Tätä DDCS-yhteysmahdollisuutta ei kuitenkaan ole ACS550-sarjan taajuusmuuttajissa, vaan jos niitä halutaan ohjata tietokoneelta, on käytettävä RS-232-pohjaista yhteyttä.

LabVIEW:ssä on muun muassa VISA Read- ja VISA Write-lohkot. Näihin lohkoihin syötetään laitteen osoite, joka sisältää tiedon siitä, miten laite on kytketty tietokoneeseen. VISA-lohkoissa tämän osoitteen nimi on VISA resource name. Lisäksi ilmoitetaan mitä tietoa halutaan lukea tai kirjoittaa. Näiden lohkojen käyttäjän eli ohjelmoijan ei tarvitse miettiä laitteen ja tietokoneen välisen yhteyden liitäntätapaa sen enempää, vaan hän voi keskittyä olennaiseen eli laitteen lukemiseen tai laitteelle kirjoittamiseen. Kaikki muu on piilotettu VISA-lohkojen sisälle.

Tätä samaa periaatetta voidaan soveltaa taajuusmuuttajan ohjaukseen. Virtuaali-instrumentit ovat nimeltään INU-Start.VI, INU-Stop.VI, INU-Reference-RPM.VI ja INU-Reference-Torque.VI (kuva 24). Lisäksi on INU-ParseResourceName.VI, jota muut INU-VI:t käyttävät osoitekentän sisältämän tiedon purkamiseen. Jokaiseen INU-virtuaali-instrumenttiin syötetään INUn (*INverter Unit*) eli taajuusmuuttajan ”osoite”. Osoite sisältää tiedon siitä, mikä yhteystapa on kyseessä. Lisäksi se voi sisältää kanavan (*channel*) ja solmunumeron (*node*) tai digitaalisen linjan osoitteen (I/O-ohjausta käytettäessä). Näin kanavalle ja solmunumerolla ei tarvita omia tuloja, vaan kuten VISA-lohkoissa, kaikki tiedot yhteyden muodostamiseen taajuusmuuttajan ja tietokoneen välille sisältyvät tähän ”osoitteeseen”.



Kuva 24. NI-VISA-tyyliset virtuaali-instrumentit taajuusmuuttajien ohjaukseen

INU-virtuaali-instrumenttien käyttämän osoitekentän nimi on INU resource name. Taulukossa 1 on esitetty kaikki vaihtoehdot osoitekentän muodolle.

Taulukko 1. INU Resource Name -osoitekentän muotosäännöt eri yhteystavoilla; vapaavalintaiset parametrit ovat hakasulkeiden sisällä

Yhteystapa	Muotosääntö
DDCS	DDCS::channel::node[::PANEL]
Modbus (RS-232)	MODBUS::node
I/O (käynnistys/pysäytys)	IO::line
I/O (referenssin asetus)	IO::channel::node
Manuaalinen ohjaus	MANUAL

DDCS-yhteyttä käytettäessä ohjaus voidaan toteuttaa joko paikallisohjauksella tai paneeliohjauksella. Paneeliohjausta käytettäessä osoitteen perään on lisättävä PANEL-määre.

I/O-ohjausta käytettäessä osoitekentällä voi olla kaksi muotoa. Käynnistettäessä tai pysäytettäessä moottori on ilmoitettava digitaalinen lähtö (*line*), joka joko asetetaan päälle tai nollataan. Referenssin asetusta varten on ilmoitettava ACS800-kuormakäytön kanava- ja solmunumero, koska referenssi asetetaan ACS800-taajuusmuuttajan analogisten lähtöjen kautta.

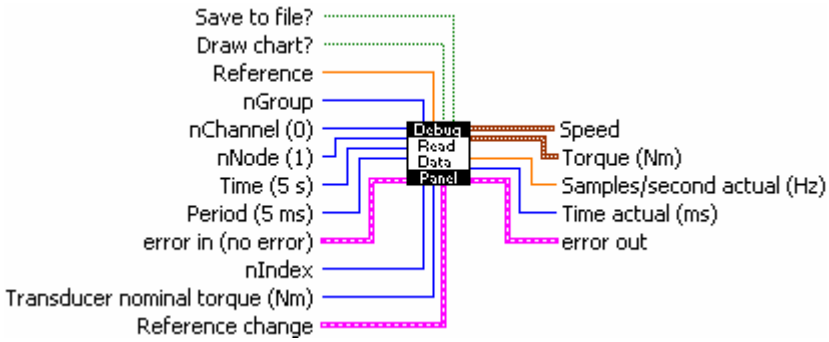
Koska kaikista DriveDebug-, DTool- ja I/O-funktioista on omat virtuaali-instrumentit, INU-VI:t kutsuvat niitä laitteen liitäntätavasta riippuen. Näin saavutetaan sama toiminnallisuus kuin VISA-lohkoilla. Ohjelmoijan tai

testaajan tarvitsee vain ilmoittaa millä tavalla laite on liitetty tietokoneeseen ja INU-VI:t kutsuvat sen perusteella oikeita virtuaali-instrumentteja.

Kun testisekvenssit tehdään käyttäen yhteystapariippumattomia INU-virtuaali-instrumentteja, ne voidaan helposti päivittää käyttämään toista yhteystapaa vain muuttamalla testattavan laitteen osoite. Näin testisekvenssit säilyvät mahdollisimman riippumattomina tiedonsiirtotavasta testausympäristön ja testattavan laitteen välillä.

6.2 Mittaustiedon keräys

Mittaukset tehdään omassa virtuaali-instrumentissa (kuva 25), joka pääosin koostuu ajastetusta silmukasta (*timed loop*). Ajastettu silmukka on LabVIEW:n rakenne, jolle voidaan suoraan syöttää sen suoritustaajuus jaksonaikana (*period*). Windowsin rajoituksista johtuen ajastettu silmukka toimii millisekuntitasolla ja pienin mahdollinen jaksonaika on 1 ms. Tämä tarkoittaa, että silmukka pyritään suorittamaan tuhat kertaa sekunnissa.



Kuva 25. DD-Panel-ReadData-RealParam-Torque-Reference.VI:n tulo- ja lähtöliitännät; virtuaali-instrumentti lukee mitattua pyörimisnopeutta ACS800-sarjan taajuusmuuttajan kautta DriveDebug-funktiota käyttäen ja momenttianturia DAQmx-funktioilla ja palauttaa muun muassa nopeuden ja momentin keski-, maksimi- ja minimiarvot

Silmukan sisällä tapahtuu nopeusparametrin luku ACS800-kuormakäytöltä ja momenttianturin lukeminen. Yleisin luettava parametri on 2.18 SPEED MEASURED. Se näyttää moottorin akselilta mitatun pyörimisnopeuden. Pyörimisnopeus mitataan pulssianturilla, joka on kytketty RTAC-01- tai RTAC-03-moduulin kautta ACS800-taajuusmuuttajaan. Parametrin lukemiseen käytetään nopeaa ReadFromRealParam-funktiota.

Momenttianturia luetaan NI USB-9215 -analogisen tulomoduulin kanavan numero 0 kautta. Lukemista varten luodaan DAQmx Create Channel

-lohkolla AI Voltage -tehtävä (*task*). Näytteenottotaajuus asetetaan DAQmx Timing -lohkolla. Mittaus aloitetaan DAQmx Start Task -lohkolla ennen ajastetun silmukan suorittamista. Näytteet luetaan silmukan sisällä yksi näyte kerrallaan DAQmx Read (Analog DBL 1Chan 1Samp) -lohkolla. Näyte on momenttianturin ulostulojännitteen suuruus, joka muutetaan momenttianturin nimellismomentin perusteella momenttiarvoksi. Ajastetun silmukan loppumisen jälkeen DAQmx-tehtävä lopetetaan DAQmx Clear Task -lohkolla.

Virtuaali-instrumentin etupaneelissa ovat Waveform Chart -objektit mittaustietojen reaaliaikaista esittämistä varten. Ajastettu silmukka on mahdollista suorittaa 1 ms:n välein, mutta näin suurella suoritusnopeudella ei ole samanaikaisesti mahdollista piirtää luettuja arvoja näytölle. Kun näytteenottoväliksi valitaan 1 ms, Waveform Chart -objektit kytketään automaattisesti pois päältä, eivätkä luetut arvot tällöin piirry näytölle.

Ajastetun silmukan suorituksen aikana on mahdollista muuttaa halutun taajuusmuuttajan nopeus- tai momenttireferenssiä. Referenssin muutosta varten VI:ssä on Reference change -ryhmätulo. Ryhmä sisältää neljä muuttujaa: Reference time (s), INU resource name, Speed/torque reference (T/F) ja Reference. Ensimmäinen muuttuja (Reference time) ilmoittaa, kuinka monta sekuntia ajastetun silmukan aloittamisen jälkeen referenssin muutos suoritetaan. Mikäli muuttujan arvo on nolla, referenssin muutosta ei suoriteta. Referenssin muutoskomento annetaan INU resource name -osoitekentän määrittelemälle taajuusmuuttajalle. Mikäli Speed/torque reference -muuttuja on tosi (*true*), muutetaan nopeusreferenssiä. Muussa tapauksessa, eli muuttujan ollessa epätosi (*false*), muutetaan momenttireferenssiä. Nopeus- tai momenttireferenssin arvo syötetään Reference-muuttujaan.

Kun silmukan suoritus on päättynyt, luetuista arvoista piirretään käyrät Waveform Graph -objekteihin, ja mittaustiedot kirjoitetaan tekstitiedostoon. Kirjoitukseen käytetään Write To Measurement File -lohkoa. Tiedosto on CSV-päätteinen (*Comma Separated Values*) ja se voidaan avata vaikka Exceliin.

Virtuaali-instrumentti palauttaa lähtöjensä kautta pyörimisnopeuden ja momentin keski-, maksimi- ja minimiarvot. Nopeuden arvot annetaan Speed-ryhmässä ja momentin arvot Torque (Nm) -ryhmässä. Lisäksi palautetaan todellinen toteutunut näytteenottotaajuus hertseissä ja näytteiden ottamiseen kulunut aika millisekunneissa.

7 TESTAUSYMPÄRISTÖN KÄYTTÖLIITTYMÄ

Testisekvenssit luodaan TestStandin sekvenssieditorilla luvussa 4 kuvattujen suorituskykytestien ja testisekvenssien perusteella. TestStandin testisekvensseissä käytetään LabVIEW:llä tehtyjä virtuaali-instrumentteja, joilla ohjataan testattavaa taajuusmuuttajaa ja ACS800-kuormakäyttöä.

Testisekvenssit voidaan suorittaa joko sekvenssieditorissa tai operaattorin käyttöliittymässä. Testisekvenssin luomisen aikana on hyvä suorittaa se suoraan sekvenssieditorista. Kun sekvenssi on saatu valmiiksi ja todettu toimivaksi, sitä voidaan käyttää operaattorin käyttöliittymästä, jossa ei ole mahdollista muokata testisekvenssiä. Operaattorin käyttöliittymä voi sijaita vaikka testauslaboratorion tietokoneella.

TestStand kirjoittaa jokaisen sekvenssin suorittamisen jälkeen XML-muotoisen (*EXtensible Markup Language*) raportin testeistä. Raportista näkyy jokaisen askeleen suorituksen tulos.

7.1 Askelryhmien käyttäminen testisekvensseissä

Askelryhmät näkyvät sekvenssieditorissa erillisinä välilehtinä. Askelryhmiä voidaan tarkastella myös operaattorin käyttöliittymässä. Jokaisessa testisekvenssissä on oletusarvoisesti Main-, Setup- ja Cleanup-askelryhmät.

Setup-askelryhmän askeleet suoritetaan aina testisekvenssin aluksi. Mikäli testisekvenssissä ohjataan taajuusmuuttajia DriveDebug- tai DTool-funktioiden kautta, on niiden InitCommunication-funktiot suoritettava mieluiten tässä askelryhmässä. Lisäksi DTool-funktioita käytettäessä on suoritettava myös ModbusConnect-funktio.

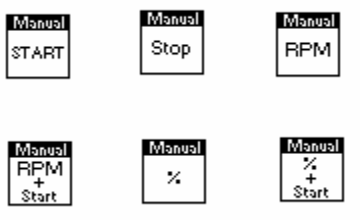
Cleanup-askelryhmän askeleet suoritetaan aina testisekvenssin lopuksi, vaikka testisekvenssin suorituksessa olisi tapahtunut virhe. Tähän askelryhmään tulee sijoittaa DriveDebugin ja DToolin StopCommunication-funktiot. Lisäksi ryhmään voidaan varmuuden vuoksi sijoittaa myös moottorin pysäyttävät *stop*-komennot. Tämä edesauttaa sitä, että moottori ei jää pyörimään missään tilanteessa, vaikka testisekvenssin suorituksen aikana tapahtuisikin virhe. Pysäytyskomennon sijoittaminen Cleanup-ryhmään ei kuitenkaan takaa moottorin pysähtymistä, koska testisekvenssin suorituksen aikana tapahtuva virhe voi kokonaan estää kommunikaation taajuusmuuttajien ja

tietokoneen välillä. Tällöin testaajan on oltava paikalla pysäyttämässä moottori joko taajuusmuuttajan paneelin kautta tai kytkemällä sähköt pois esimerkiksi Hätäseis-painikkeella.

7.2 Testattavan laitteen manuaalinen ohjaus

Testausohjelmisto osaa ohjata automaattisesti laitteita DriveDebug-funktioiden DDCS-yhteyden, DTool-funktioiden ModBus-yhteyden ja USB-väyläisten I/O-moduulien kautta. On myös mahdollista, että testausjärjestelmä ei itse ohjaa testattavaa laitetta. Tähän voi olla syynä se, että testattava taajuusmuuttaja (DUT) on vielä niin varhaisessa kehitysvaiheessa, että mitään testausjärjestelmän tukemaa kommunikointikeinoa ei ole mahdollista käyttää. Voi myös olla, että testausjärjestelmän kommunikointi ei ole yhteensopivaa DUT:n kanssa, ja testausjärjestelmän virtuaali-instrumentteja ei ole mahdollista tai vielä ehditty päivittää tukemaan uutta kommunikointiprotokollaa. Tällöin testaajan on ohjattava DUT:tä manuaalisesti.

Testausohjelmisto ilmoittaa testaajalle, kun testaajan on käynnistettävä tai pysäytettävä testikäyttö tai annettava sille testausjärjestelmän ilmoittama nopeus- tai momenttireferenssi. Testaaja ilmoittaa OK-nappia painamalla, kun laitteelle on annettu tietty komento, jolloin testausohjelmisto tietää siirtyä testisekvenssin seuraavaan askeleeseen. Mikäli testaaja ei kykene suorittamaan testausohjelmiston ilmoittamaa toimintoa tai haluaa jostain muusta syystä keskeyttää testin, on mahdollista painaa Cancel-nappia. Cancel-nappi tuottaa virheen kyseisen virtuaali-instrumentin error out -ryhmään, jolloin testin suoritus keskeytyy.



Kuva 26. Manuaalisesti ohjattavien taajuusmuuttajien testaamiseen tarkoitettujen virtuaali-instrumenttien kuvakkeet; jokainen virtuaali-instrumentti avaa ikkunan, jossa kerrotaan, mikä komento testaajan on annettava testattavalle taajuusmuuttajalle

Kuvassa 26 on esitetty manuaaliseen ohjaukseen käytettyjen virtuaali-instrumenttien kuvakkeet. Kyseisiä virtuaali-instrumentteja tulee käyttää

testisekvensseissä, kun testattavaa taajuusmuuttajaa ei ole mahdollista ohjata automaattisesti testausohjelmiston tukemalla tavalla.

Manuaalista ohjausta käytettäessä ei voida enää puhua automaattisesta testauksesta. Tällöin testausjärjestelmä toimii vain mittalaitteena, joka kerää tietoja kuormakäyttönä toimivalta ACS800-taajuusmuuttajalta ja momenttianturilta. Koska mitään testausjärjestelmän tukemaa kommunikointikeinoa DUT:n ja testausjärjestelmän välillä ei voida käyttää, testausjärjestelmä ei voi myöskään kerätä tietoja DUT:ltä. Testaajan on oltava testin suorituksen aikana testipaikalla.

8 POHDINTAA

Tässä luvussa testausympäristöä verrataan toiseen testauslaitteistoon sekä pohditaan automaattista testausta yleisesti. Lisäksi arvioidaan mittaustietoa keräävää virtuaali-instrumenttia ja esitetään siihen jatkokehitysaiheita.

8.1 Suorituskyvyn testausympäristö verrattuna I/O-testeriin

ABB:n tuotekehityslaboratorioissa on käytetty niin sanottua I/O-testeriä ohjelmiston toiminnallisuuden testausta varten. Nimensä mukaisesti I/O-testeri testaa ACS800-taajuusmuuttajan ohjausta sen analogisten ja digitaalisten I/O-liityntöjen kautta.

I/O-testeriä varten kehitettiin oma skriptikieli Visual Basicillä. Testisekvenssit kirjoitetaan tällä skriptikielellä, joka muistuttaa Visual Basicia. Testit suoritetaan avaamalla skriptitiedosto testisovellukseen ja ajamalla se. Testien tekemisen helpottamiseksi on kehitetty myös Excel-pohjainen sovellus, joka käyttäjän tekemän taulukon perusteella tuottaa I/O-testerin käyttämää skriptikieltä. Näin testien kirjoitus saatiin helpommaksi, mutta Excel-sovellusta ei voida käyttää kaikkien testien luomiseen.

I/O-testerin suurimmat ongelmat ovat: testerin käyttämä skriptikieli ei ole mikään yleinen ohjelmointikieli vaan testien kirjoittamista varten täytyy opetella uusi skriptikieli, skriptejä on vaikea lukea ja saada selvää kuvaa testin toiminnasta, testien tuloksia on vaikea tallentaa eikä testien tuloksista voida aina olla varmoja, koska I/O-testeri saattaa kaatua. Kaatumisen syy voi olla testattavan taajuusmuuttajan ohjelmistossa, I/O-testerin laitteistossa tai sen ohjelmistossa (skriptissä).

I/O-testeristä poiketen tässä työssä kehitetty suorituskyvyn testausympäristö perustuu teollisuudessa yleisesti käytettyihin ohjelmistoihin, LabVIEW-ohjelmointiympäristöön ja TestStand-testausohjelmistoon. Sitä varten ei tarvitse opetella mitään uutta ohjelmointi- tai skriptikieltä, jos osaa jo ennestään ohjelmoida LabVIEW:llä.

Suorituskykytestien sekvenssit luodaan ja suoritetaan TestStandissa. Sekvenssin askeleet näkyvät selkeästi omina riveinään. Testisekvenssit ja testien toiminnallisuuden toteuttava ohjelma ovat erotettu toisistaan. Testien suoritus ja raportointi on hallittua. Kaikki mittaustulokset kirjoitetaan tekstitiedostoon. Raportti suoritetuista testeistä kirjoitetaan XML-tiedostoon.

8.2 Automaattisesta testauksesta

Kuten Antti Asikainen esittää diplomityössään [23, s. 61 - 62], automaattinen testaus ei välttämättä vähennä testeihin tarvittavaa työmäärää eikä testien automatisointi tule koskaan täysin korvaamaan manuaalista testausta. Testien ohjelmointi ja ylläpito voivat teettää enemmän työtä kuin manuaalisesti suoritetut testit. Automaattisten testien tulosten analysointi on työlästä.

Tämä suorituskyvyn testausympäristö analysoi automaattisesti vain staattisen suorituskyvyn testien tulokset. Ne on helppo analysoida keski-, maksimi- ja minimiarvojen perusteella. Dynaamisen suorituskyvyn testien tuloksia ei analysoida. Mitä enemmän dynaamisen suorituskyvyn testejä tehdään eri pyörimisnopeuksilla tai kuormituksilla, sitä enemmän mittaustuloksia jää manuaalisesti analysoitavaksi.

Asikainen esittää, että testien ylläpidettävyys on maksimoitava suunnittelemalla testit mahdollisimman riippumattomiksi muutoksista, jotta automaattisesta testauksesta saadaan mahdollisimman suuri hyöty. Tiedonsiirto testattavan taajuusmuuttajan ja testausympäristön välillä tulee toteuttaa erillisenä toiminnallisuutena. Testitapaukset eivät saa olla riippuvaisia tiedonsiirtotavasta.

Tämän testausympäristön TestStand-testisekvensseistä voidaan tehdä laitteistoriippumattomia käyttämällä INU-virtuaali-instrumentteja. Samaa testisekvenssiä voidaan käyttää eri taajuusmuuttajien kanssa muuttamalla INU resource name -osoitekenttää. Sekvenssin rakenne pysyy muuten samanlaisena.

8.3 Mittaustiedon keräävän virtuaali-instrumentin arviointia

Pyörimisnopeuden ja momenttianturin samanaikaista lukemista varten on yksi virtuaali-instrumentti, joka palauttaa muun muassa pyörimisnopeuden ja momentin keski-, maksimi- ja minimiarvot. Tätä virtuaali-instrumenttia voidaan suoraan hyödyntää nopeus- ja momenttisäätöjen staattisten tarkkuuksien testeissä.

Samalla virtuaali-instrumentilla on mahdollista muuttaa halutun taajuusmuuttajan nopeus- tai momenttireferenssiä mittaussilmukan aikana. Näin saadaan kerättyä tiedot pyörimisnopeudesta ja momentista myös dynaamisessa tilassa. Tällä virtuaali-instrumentilla ei kuitenkaan pystytä lukemaan samanaikaisesti muita asioita, kuten esimerkiksi testattavan taajuusmuuttajan estimoimaa pyörimisnopeutta. Tämä on mahdollista lisätä virtuaali-instrumenttiin myöhemmin, mutta silloin näytteenottotaajuus saattaa laskea.

Mahdollisimman hyvää näytteenottotaajuutta tavoiteltaessa mittauksien suorittamista varten tulisi kehittää erilaisia virtuaali-instrumentteja. Näytteenottotaajuuteen vaikuttaa muun muassa testattavan taajuusmuuttajan ja tietokoneen välinen yhteys. Esimerkiksi yhden parametrin luku DTool-funktioita käyttämällä kestää monta millisekuntia, kun taas DriveDebug-funktioita käyttämällä päästään alhaisimmillaan yhteen millisekuntiin.

Hitaat mittaukset eivät saisi hidastaa nopeita mittauksia, kuten momenttianturin lukua. Mittaukset tulisi tehdä parhaalla mahdollisella näytteenottotaajuudella. Parasta mahdollista näytteenottotaajuutta tavoiteltaessa mittaussilmukka tulisi ohjelmoida siten, että se lukisi eri asiat eri taajuudella. Tätä varten pitää etukäteen selvittää eri taajuusmuuttajan ja tietokoneen välisten yhteyksien nopeudet.

Mittaukset suorittava virtuaali-instrumentti ei analysoi tuloksia. Se kirjoittaa kaikki näytteet tekstitiedostoon. Dynaamisen suorituskyvyn testien tapauksessa mittauksia pitää analysoida ja selvittää esimerkiksi vasteaika ja maksimipoikkeama. Tämä jää vielä käsin tehtäväksi. LabVIEW sisältää paljon signaalien analysointiin tarkoitettuja funktioita. Niistä voi mahdollisesti olla hyötyä myös tämän testausympäristön mittauksien käsittelyssä.

9 YHTEENVETO

Tässä opinnäytetyössä kehitettiin taajuusmuuttajien suorituskyvyn automaattinen testausympäristö. Mittausten näytteenottoväliksi saatiin tavoitteeksi asetettu 5 ms luettaessa momenttianturia ja pyörimisnopeutta ACS800:n kautta samanaikaisesti. Näytteenottoväli voi olla pienimmillään 1 ms eli näytteenottotaajuus on tällöin 1 kHz.

Työn tuloksena syntyi virtuaali-instrumentteja, joilla voidaan muun muassa ohjata taajuusmuuttajia sekä lukea taajuusmuuttajien parametrejä ja momenttianturia. Virtuaali-instrumenteista koostetuilla TestStand-testisekvensseillä taajuusmuuttajien nopeus- ja momenttisäädön suorituskykytestit voidaan tehdä automaattisesti. Virtuaali-instrumentit sallivat testien tekemisen millä tahansa pyörimisnopeudella ja kuormituksella. TestStandin avulla testit voidaan automaattisesti suorittaa kattavasti koko testikäytön sallimalla pyörimisnopeus- ja kuormitusalueella.

Testausympäristöä voidaan hyödyntää sekä nykyisen että seuraavan sukupolven taajuusmuuttajien testauksessa. Ympäristö perustuu teollisuudessa yleisesti käytettyihin ohjelmistoihin ja tarjoaa hyvät mahdollisuudet jatkokehitykselle.

Tulosten raportointia olisi mahdollista kehittää huomattavasti. Nykyisessä muodossa TestStand-käyttöliittymä ilmoittaa mittaustulosten keskiarvon ja sen perusteella päättää menikö testi läpi vai ei. Lisäksi mittaukset suorittava virtuaali-instrumentti tallentaa mittaustulokset tekstitiedostoon. Testausohjelmisto voisi automaattisesti mittaustietojen perusteella piirtää graafiset käyrät tulosten analysointia varten. Tämä voitaisiin toteuttaa omana virtuaali-instrumenttina, joka lukisi mittaustulokset tekstitiedostosta ja piirtäisi niistä kuvat. Lisäksi mittaustulokset voitaisiin viedä suoraan tietokantaan. TestStandissa on mahdollista käsitellä muun muassa SQL-tietokantoja (*Structured Query Language*) suoraan testisekvensseistä.

Testausympäristöä varten kehitettyihin virtuaali-instrumentteihin ei ole kattavaa dokumentaatiota eikä versionhallintaa ole hyödynnetty. Virtuaali-instrumentit tulisi viedä johonkin versionhallintajärjestelmään, jotta olisi mahdollista seurata muutoksia ja jatkokehitystyötä. Tällä hetkellä virtuaali-instrumenttien dokumentaationa toimii liitteessä 3 esitetyt virtuaali-instrumenttien kuvaukset ja lähdekoodit.

Taajuusmuuttajia on mahdollista ohjata monilla eri kenttäväylillä. Kenttäväyläohjausta ei kuitenkaan ollut mahdollista toteuttaa tämän työn aikarajoissa. Testausympäristössä voitaisiin hyödyntää muun muassa kannettaviin tietokoneisiin saatavia kenttäväylälaajennuskortteja. Kenttäväyläohjausta varten pitäisi tehdä samanlaiset virtuaali-instrumentit kuin muita ohjaustapoja varten tehtiin.

Kattavampia testejä varten testausympäristössä tulisi olla myös tehonmittaus. Testattavasta taajuusmuuttajasta mitattaisiin välipiirin jännite ja moottorin vaihevirrat. Tehonmittaus voitaisiin suorittaa vaikka tietokoneeseen kytkettävällä oskilloskoopilla. Nykyiset kehittyneet oskilloskoopit sisältävät Windows-käyttöjärjestelmän ja muun muassa USB-väylän ja verkkoyhteyden. Mikäli oskilloskoopin käyttöjärjestelmään olisi mahdollista asentaa LabVIEW Run-Time -suoritusympäristö ja TestStand, testisekvenssit voitaisiin suorittaa oskilloskoopin sisällä, ja koko testausympäristö olisi integroituna oskilloskooppiin.

LÄHTEET

[1] *ACS800 Tuoteluettelo*. ABB Oy. 2003.

[2] Niiranen, Jouko, *Sähkömoottorikäytön digitaalinen ohjaus*. Helsinki: Otatieto. 2000.

[3] *T32FNA Torque transducers Mounting instructions*. HBM.

[4] *MC60 Amplifier plug-in unit Data sheet*. HBM. 1997.

[5] *MP60/MP07 Operating manual*. HBM. 2006.

[6] *Dataflex-vääntömomentin mittausakselit*. Tuote-esite. KTR Finland Oy.

[7] *Dataflex 22/… Torque Measuring Shaft Assembly-/Operating Instructions*. KTR. 2005.

[8] *USB-9215 Series User Guide and Specifications*. National Instruments Corporation. 2005.

[9] *USB-9421 User Guide and Specifications*. National Instruments Corporation. 2005.

[10] *USB-9472 User Guide and Specifications*. National Instruments Corporation. 2006.

[11] Wilenius, Aku, *Virtuaali-instrumentin ohjelmointi LabVIEW'lla*. Insinöörityö. Espoo-Vantaan teknillinen ammattikorkeakoulu. Tietotekniikan koulutusohjelma. 2001.

[12] *TestStand Reference Manual*. National Instruments Corporation. 2003.

[13] *Using TestStand*. National Instruments Corporation. 2003.

[14] *Rating specifications for a.c. power drive systems above 1 000 V a.c. and not exceeding 35 kV*. Referenssinumero IEC 61800-4:2002. Kansainvälinen standardi. Geneve: IEC:n pääkonttori. 2002.

[15] *Rating specifications for low voltage adjustable frequency a.c. power drive systems*. Referenssinumero IEC 61800-2:1998. Kansainvälinen standardi. Geneve: IEC:n pääkonttori. 1998.

[16] Lalu, Jarkko, *Speed control dynamic performance*. Testimäärittely. ABB Oy. 2002.

[17] Lalu, Jarkko, *Torque control linearity and repeatability test*. Testimäärittely. ABB Oy. 2002.

[18] Harjunen, Olli, *DriveDebug functions for Visual Basic Programs*. ABB Oy. 2000.

[19] Harjunen, Olli, *DTool Functions for VB and C Programs*. ABB Oy. 2005.

[20] *ACS800 Standard Application Program 7.x Firmware Manual*. ABB Oy. 2005.

[21] *ACS800-01 Drives (0,55 to 110 kW) Hardware Manual*. ABB Oy. 2005.

[22] *ACS550 User's Manual*. ABB Oy. 2004.

[23] Asikainen, Antti, *Taajuusmuuttajan ohjelmiston testaus*. Diplomityö. Lappeenrannan teknillinen yliopisto. Sähkötekniikan osasto. 2006.

Taulukko 2. DriveDebug-funktiot ja niiden C-kieliset prototyypit [18, s. 48 - 50]

InitCommunication	long _stdcall InitCommunication (short *pnNumberOfChannels);
StopCommunication	long _stdcall StopCommunication (void);
UploadParamDefinitions	long _stdcall UploadParamDefinitions (short nChannel, short nNode, short *pnParamCount, unsigned char *pbyParamDefs);
ParamReadBoolean	long _stdcall ParamReadBoolean (short nChannel, short nNode, short nGroup, short nIndex, short *pnValue);
ParamWriteBoolean	long _stdcall ParamWriteBoolean (short nChannel, short nNode, short nGroup, short nIndex, short nNewValue);
ParamReadInt16	long _stdcall ParamReadInt16 (short nChannel, short nNode, short nGroup, short nIndex, short *pnValue);
ParamWriteInt16	long _stdcall ParamWriteInt16 (short nChannel, short nNode, short nGroup, short nIndex, short nNewValue);
ParamReadBitString	long _stdcall ParamReadBitString (short nChannel, short nNode, short nGroup, short nIndex, long *plValue);
ParamWriteBitString	long _stdcall ParamWriteBitString (short nChannel, short nNode, short nGroup, short nIndex, long lNewValue);

ParamReadReal	long _stdcall ParamReadReal (short nChannel, short nNode, short nGroup, short nIndex, float *pfValue);
ParamWriteReal	long _stdcall ParamWriteReal (short nChannel, short nNode, short nGroup, short nIndex, float fNewValue);
ParamReadReal16	long _stdcall ParamReadReal16 (short nChannel, short nNode, short nGroup, short nIndex, short *pnValue);
ParamWriteReal16	long _stdcall ParamWriteReal16 (short nChannel, short nNode, short nGroup, short nIndex, short nNewValue);
ParamReadString	long _stdcall ParamReadString (short nChannel, short nNode, short nGroup, short nIndex, char *pszValue);
ParamWriteString	long _stdcall ParamWriteString (short nChannel, short nNode, short nGroup, short nIndex, char *pszNewValue);
FaultLoggerUpload	long _stdcall FaultLoggerUpload (short nChannel, short nNode, short *pnFaultCount, unsigned char pbyFaults);
FaultLoggerClear	long _stdcall FaultLoggerClear (short nChannel, short nNode);
LocalControlOperation	long _stdcall LocalControlOperation (short nChannel, short nNode, short nOperation);
LocalControlReference	long _stdcall LocalControlReference (short nChannel, short nNode, float fRefValue);

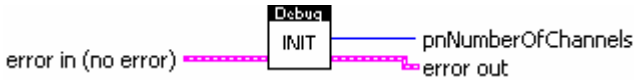
LocalControlRefresh	long _stdcall LocalControlRefresh (short nChannel, short nNode, long *plStatus, float *pfRefCurr, float *pfRefMax, float *pfRefMin, char *pszRefUnit);
ReadFromIntParam	long _stdcall ReadFromIntParam (short nChannel, short nNode, short nGroup, short nIndex, long *plValue);
WriteToIntParam	long _stdcall WriteToIntParam (short nChannel, short nNode, short nGroup, short nIndex, long lNewValue);
ReadFromRealParam	long _stdcall ReadFromRealParam (short nChannel, short nNode, short nGroup, short nIndex, float *pfValue);
WriteToRealParam	long _stdcall WriteToRealParam (short nChannel, short nNode, short nGroup, short nIndex, float fNewValue);

Taulukko 3. DTool-funktiot ja niiden C-kieliset prototyypit [19, s. 13]

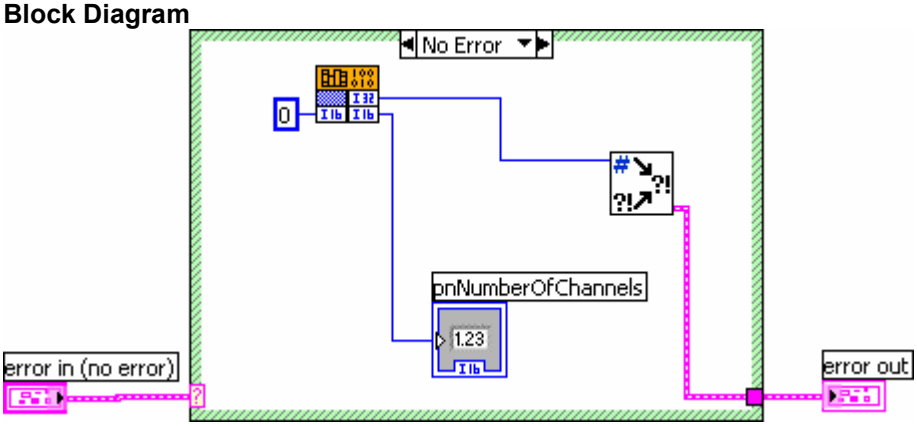
InitCommunication	long _stdcall InitCommunication (short *pnNumberOfChannels);
ModbusConnect	long _stdcall ModbusConnect (short nChannel, short nNode);
StopCommunication	long _stdcall StopCommunication (void);
UploadParamDefinitions	long _stdcall UploadParamDefinitions (short nChannel, short nNode, short *pnParamCount, unsigned char *pbyParamDefs);
ParamReadInt16	long _stdcall ParamReadInt16 (short nChannel, short nNode, short nGroup, short nIndex, short *pnValue);
ParamWriteInt16	long _stdcall ParamWriteInt16 (short nChannel, short nNode, short nGroup, short nIndex, short nNewValue);
ParamReadReal	long _stdcall ParamReadReal (short nChannel, short nNode, short nGroup, short nIndex, float *pfValue);
ParamWriteReal	long _stdcall ParamWriteReal (short nChannel, short nNode, short nGroup, short nIndex, float fNewValue);
ShowHiddenParameters	long _stdcall ShowHiddenParameters (short nChannel, short nNode);
HideHiddenParameters	long _stdcall HideHiddenParameters (short nChannel, short nNode);

DD-InitCommunication.vi
Initializes the communication software and hardware. This function must be called before the use of other DriveDebug functions. Returns the number of DDCS channels in the DDCS communication card of the computer.

Connector Pane



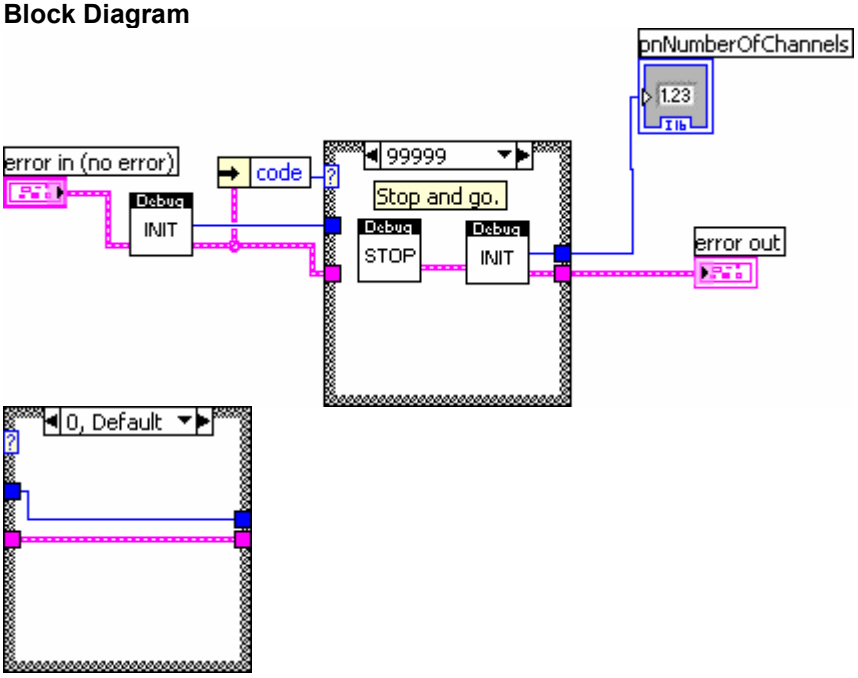
Block Diagram



DD-InitCommunication-StopAndGo.vi
If communication has already been initialized before using this VI, this VI stops the communication and initializes it again. Returns the number of DDCS channels in the DDCS communication card of the computer.

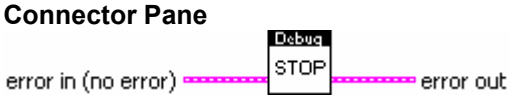
Connector Pane

Block Diagram

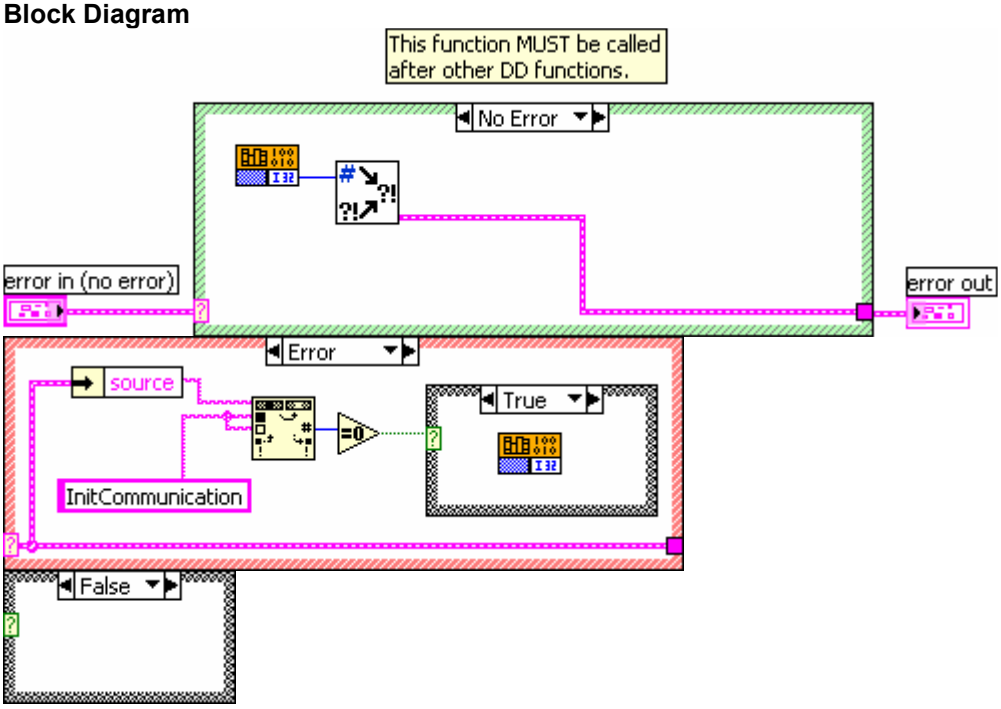


DD-StopCommunication.vi
Stops the use of the communication software and hardware. This function must be called after the use of other DriveDebug functions.

Connector Pane

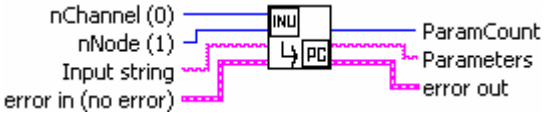


Block Diagram

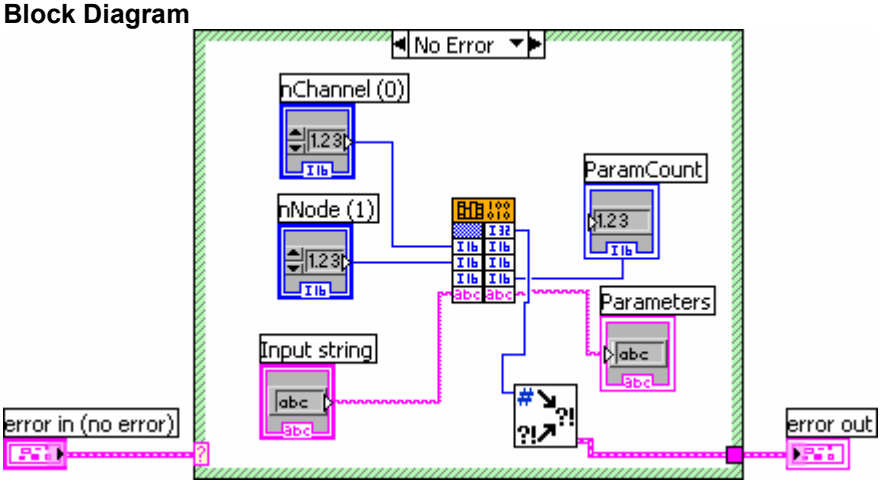


DD-UploadParameters.vi
Uploads the parameter definition list from the drive. Input string must be big enough to hold all parameters.

Connector Pane

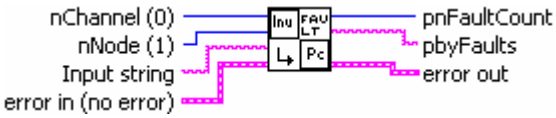


Block Diagram

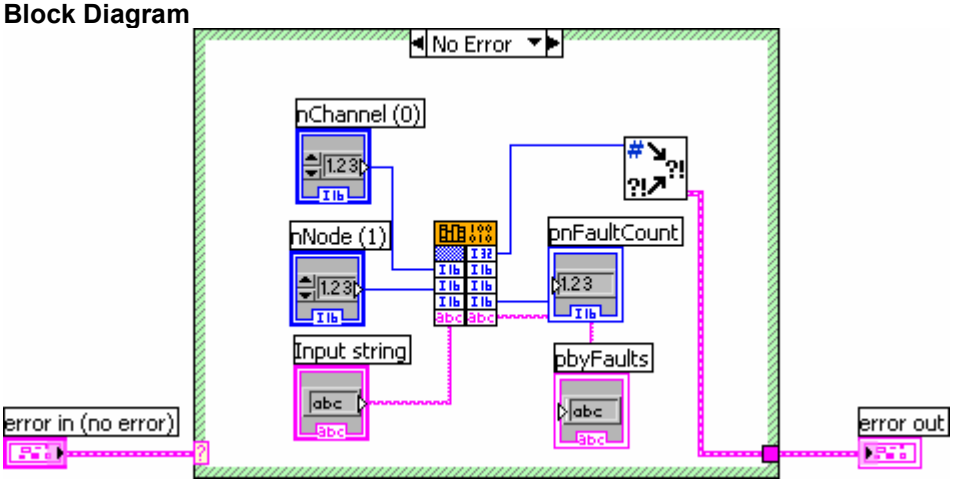


DD-FaultUpload.vi
Uploads the contents of fault logger. Input string must be big enough to hold the contents of fault logger.

Connector Pane

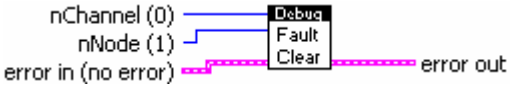


Block Diagram

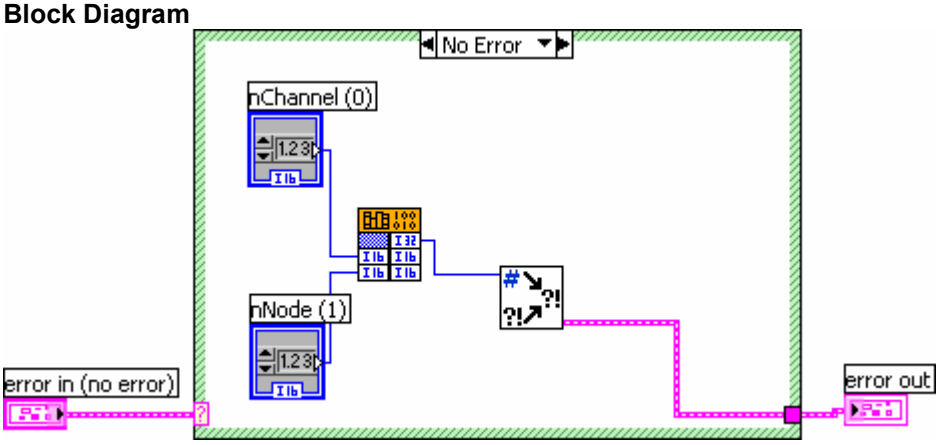


DD-FaultClear.vi
Clears the contents of fault logger.

Connector Pane

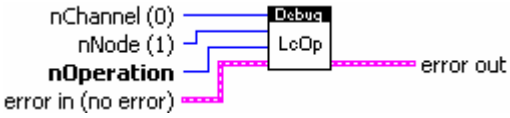


Block Diagram

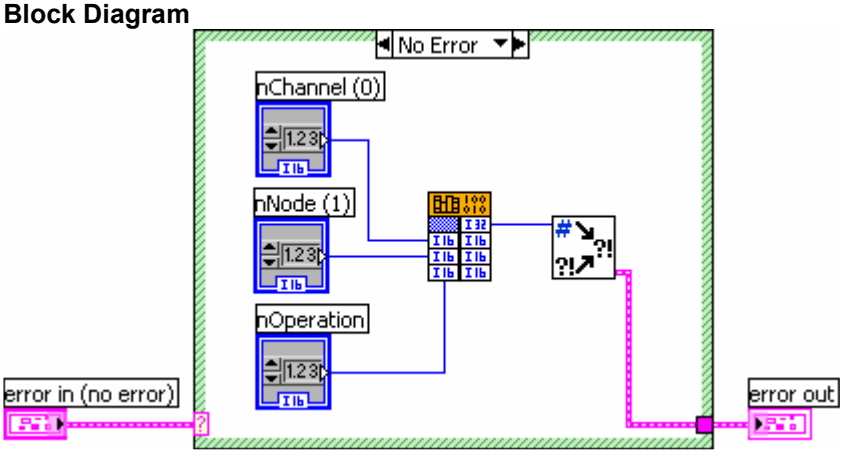


DD-LocalControlOperation.vi
Executes a local control operation.

Connector Pane



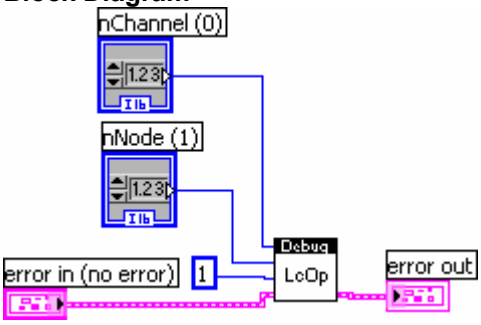
Block Diagram



DD-LocalControlOperation-Local.vi
Sets the drive to local control mode.

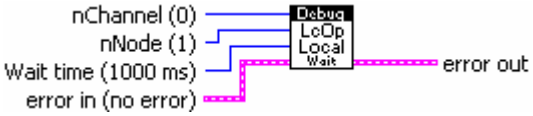
Connector Pane

Block Diagram

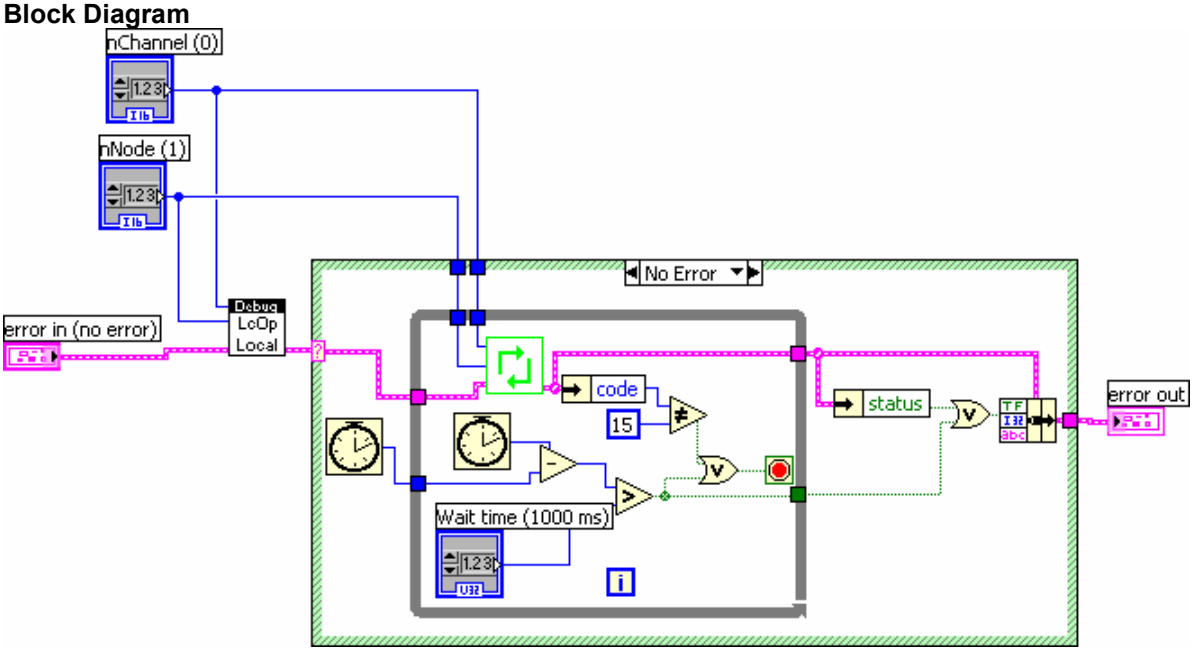


DD-LocalControlOperation-Local-Wait.vi
Sets the drive to local control mode and waits for local control mode to become active. Otherwise returns an error.

Connector Pane

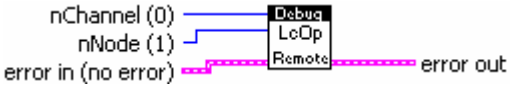


Block Diagram

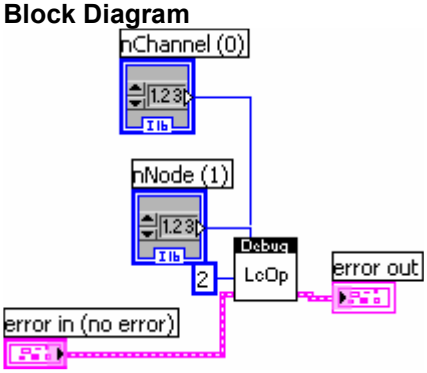


DD-LocalControlOperation-Remote.vi
Sets the drive to remote control mode.

Connector Pane

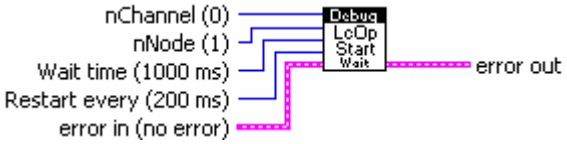


Block Diagram

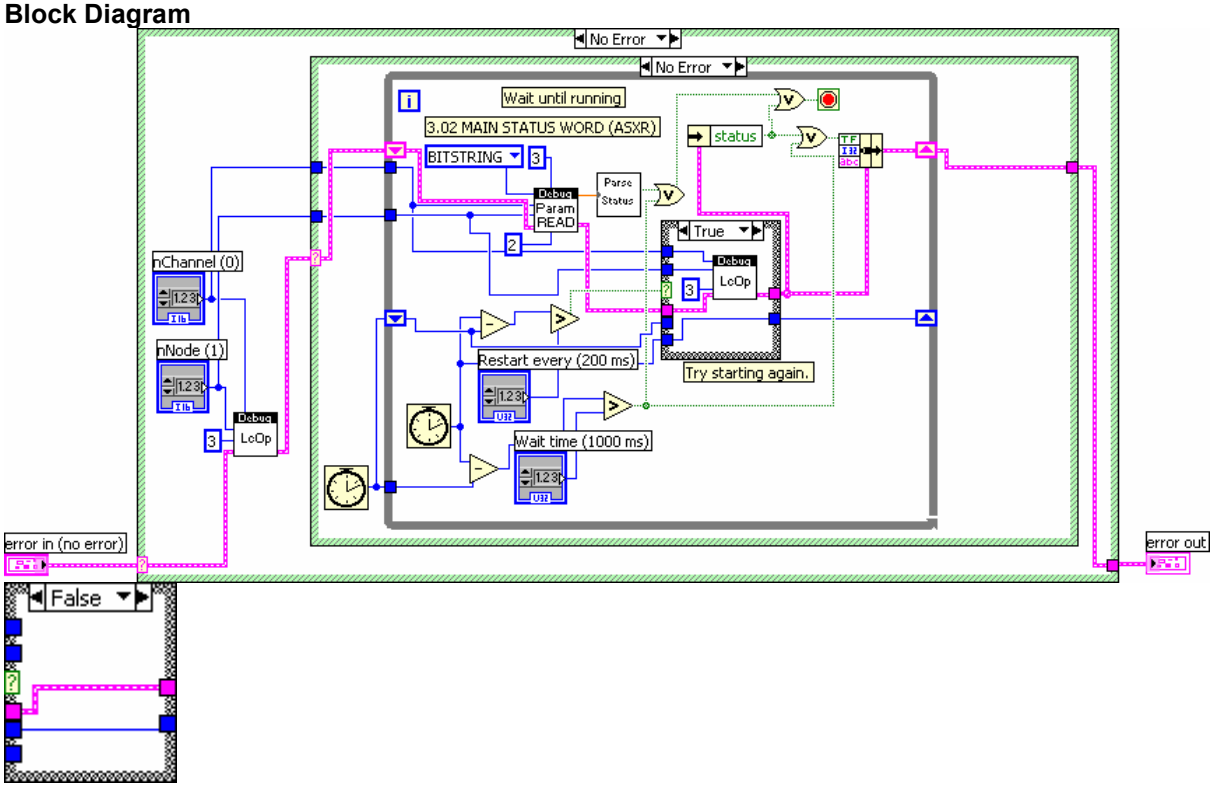


DD-LocalControlOperation-Start-Wait.vi
Starts the motor and waits for the motor to start running. Repeats the start command and returns an error if motor does not start.

Connector Pane

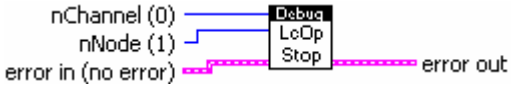


Block Diagram

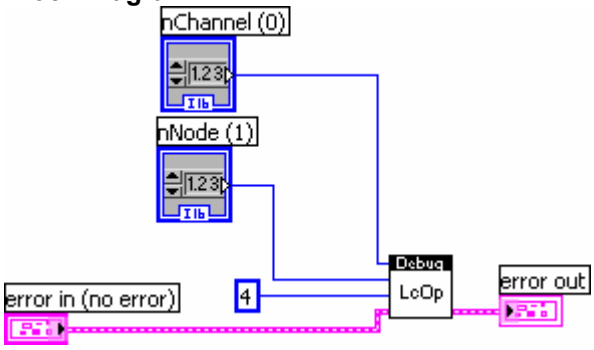


DD-LocalControlOperation-Stop.vi
Stops the motor.

Connector Pane

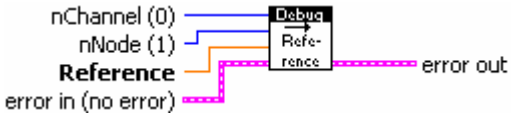


Block Diagram

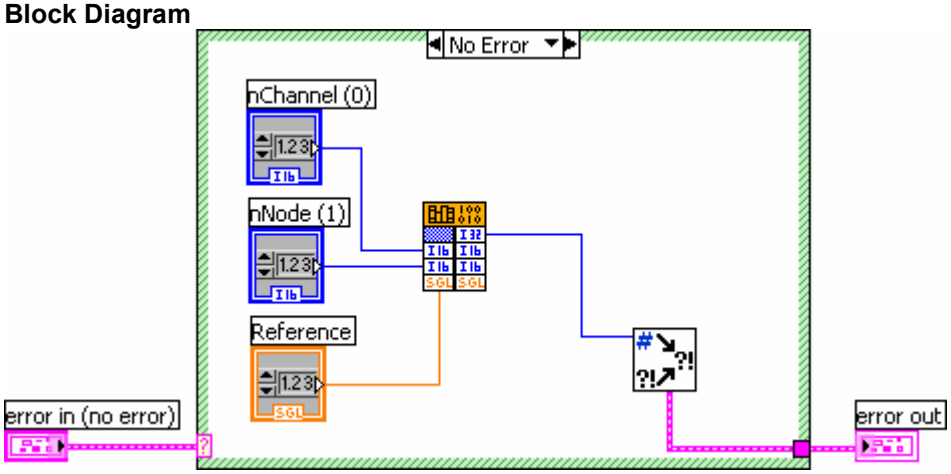


DD-LocalControlReference.vi
Sets a new local reference value.

Connector Pane

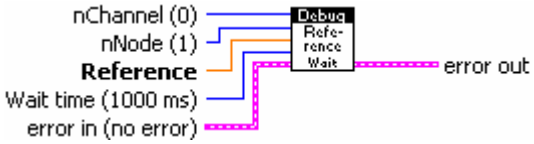


Block Diagram

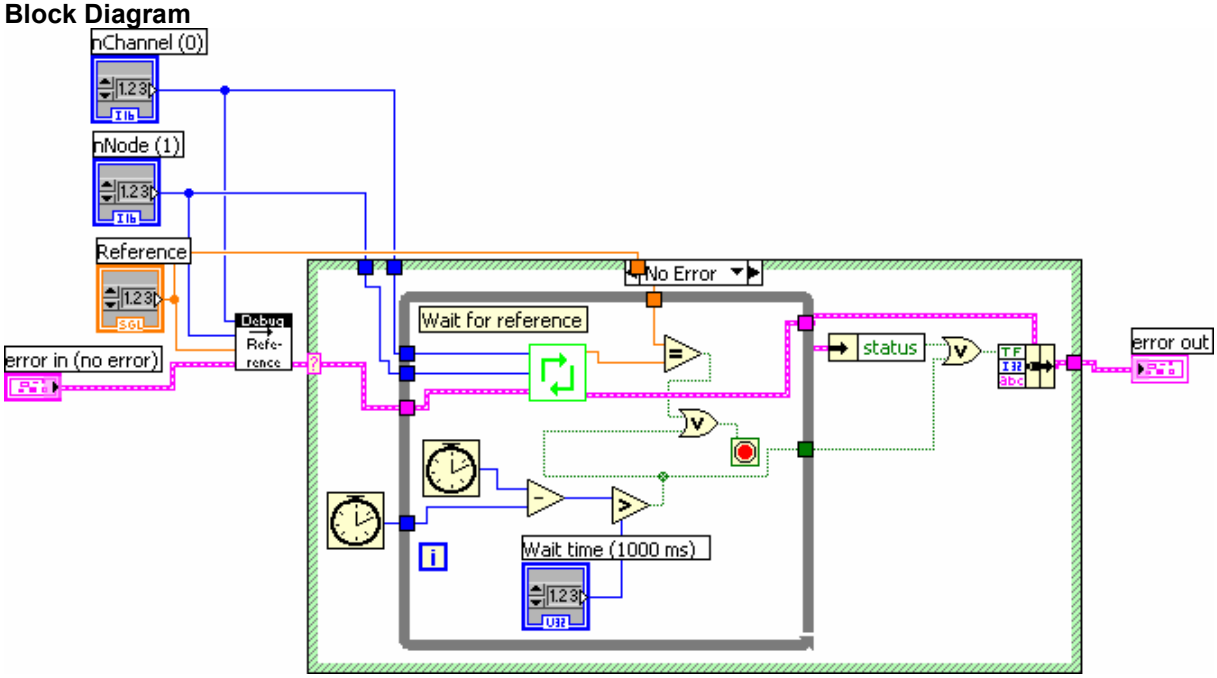


DD-LocalControlReference-Wait.vi
Sets a new local reference value and waits for the reference to become active. Otherwise returns an error.

Connector Pane

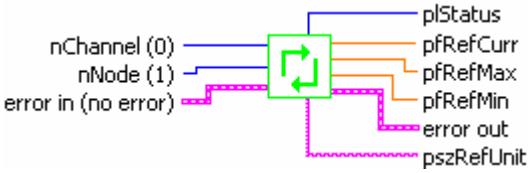


Block Diagram

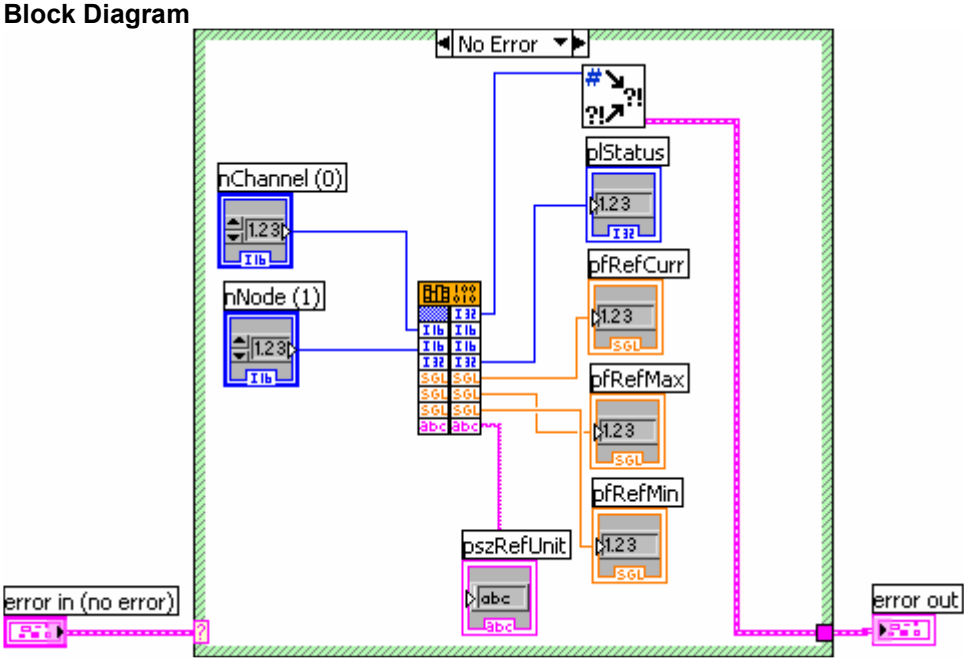


DD-LocalControlRefresh.vi
Refreshes the local control state and reads the current values of the local control data.

Connector Pane

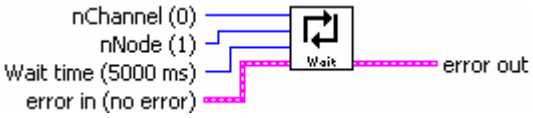


Block Diagram

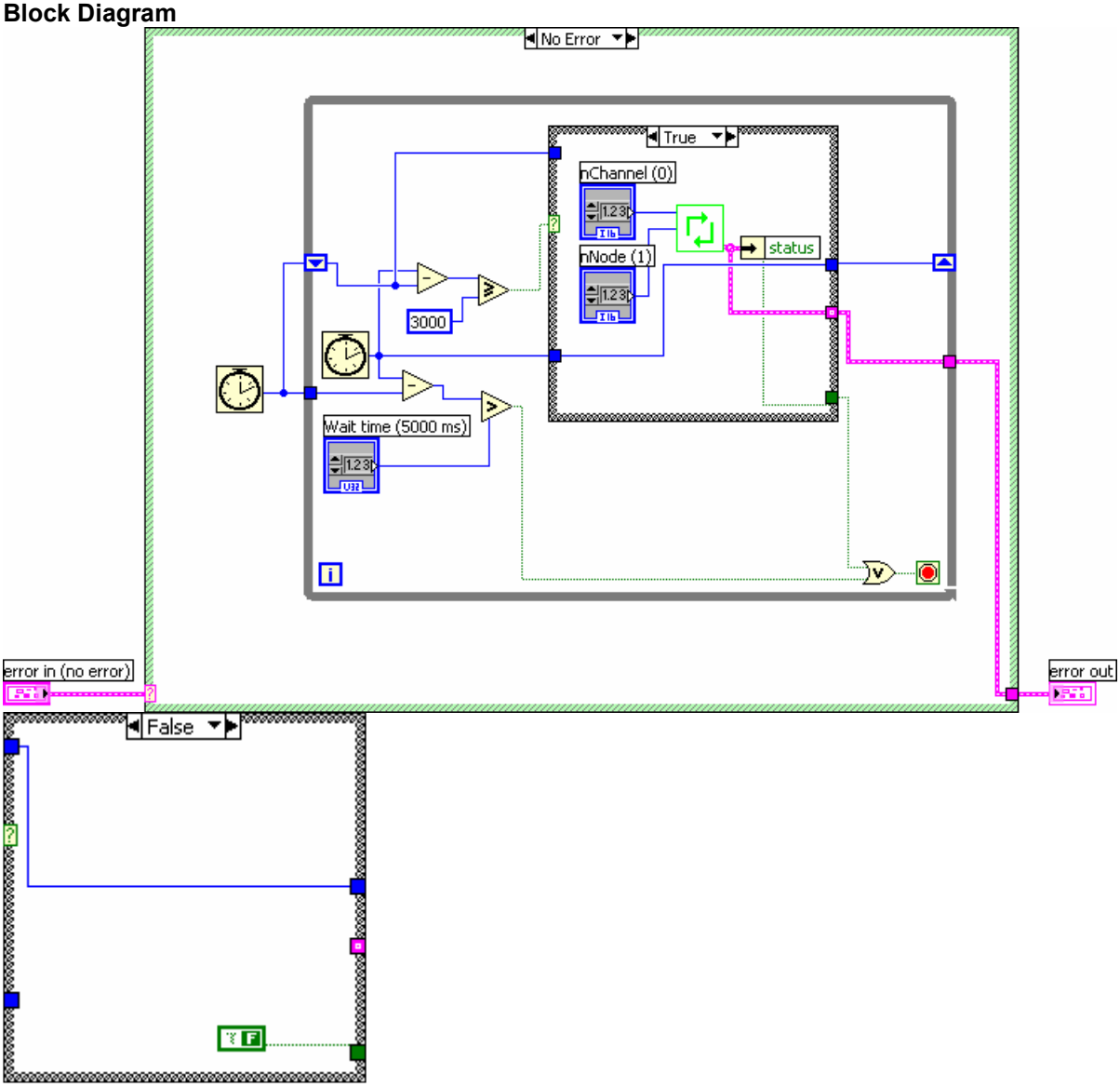


DD-LocalControlRefresh-Wait.vi
Waits a specified amount of time and refreshes the local control state.

Connector Pane

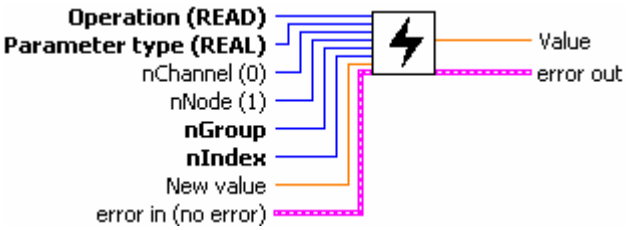


Block Diagram

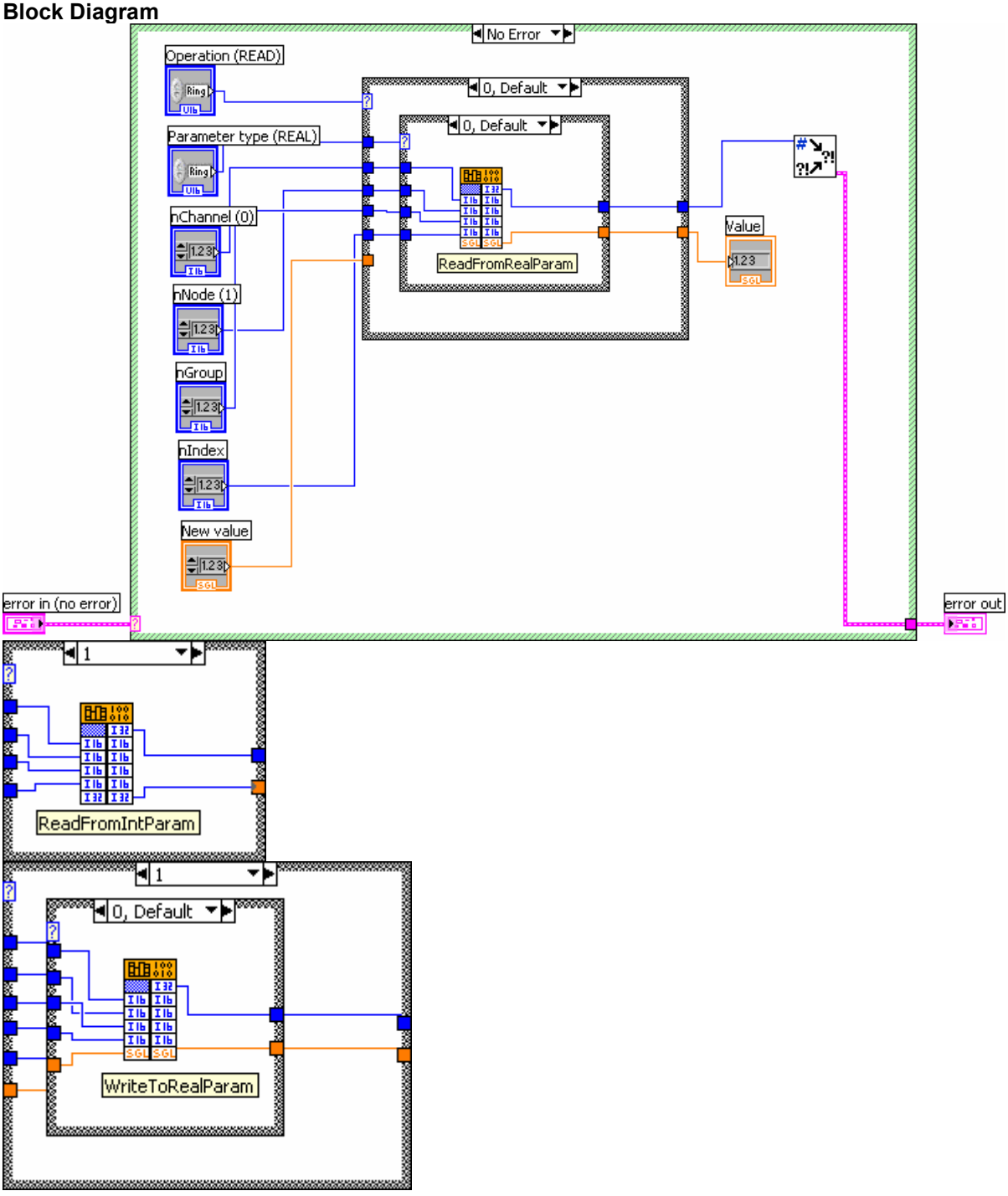


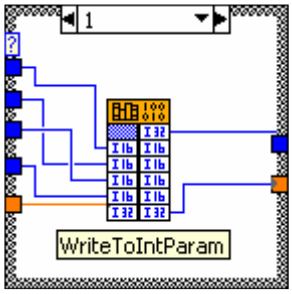
DD-FastParameterFunctions.vi
Reads a value of a parameter or writes a new value to a parameter by using the fast parameter functions.

Connector Pane



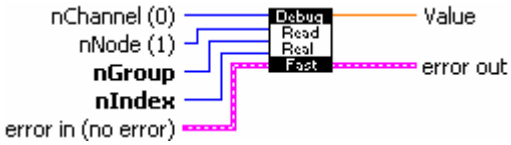
Block Diagram



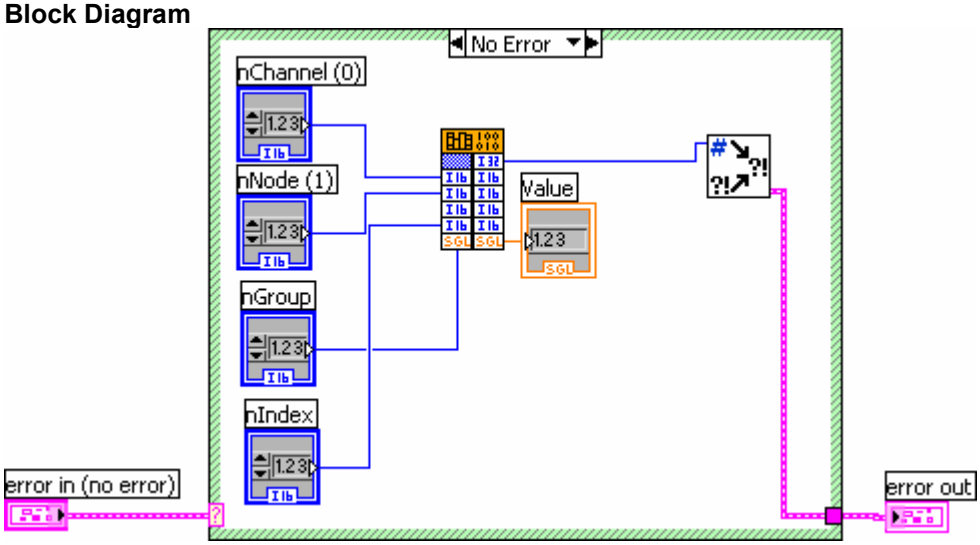


DD-FastParameterFunctions-ReadFromRealParam.vi
Reads a value of a REAL parameter by using a fast parameter function.

Connector Pane

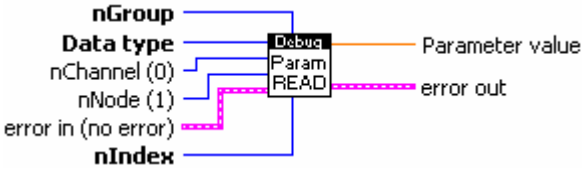


Block Diagram

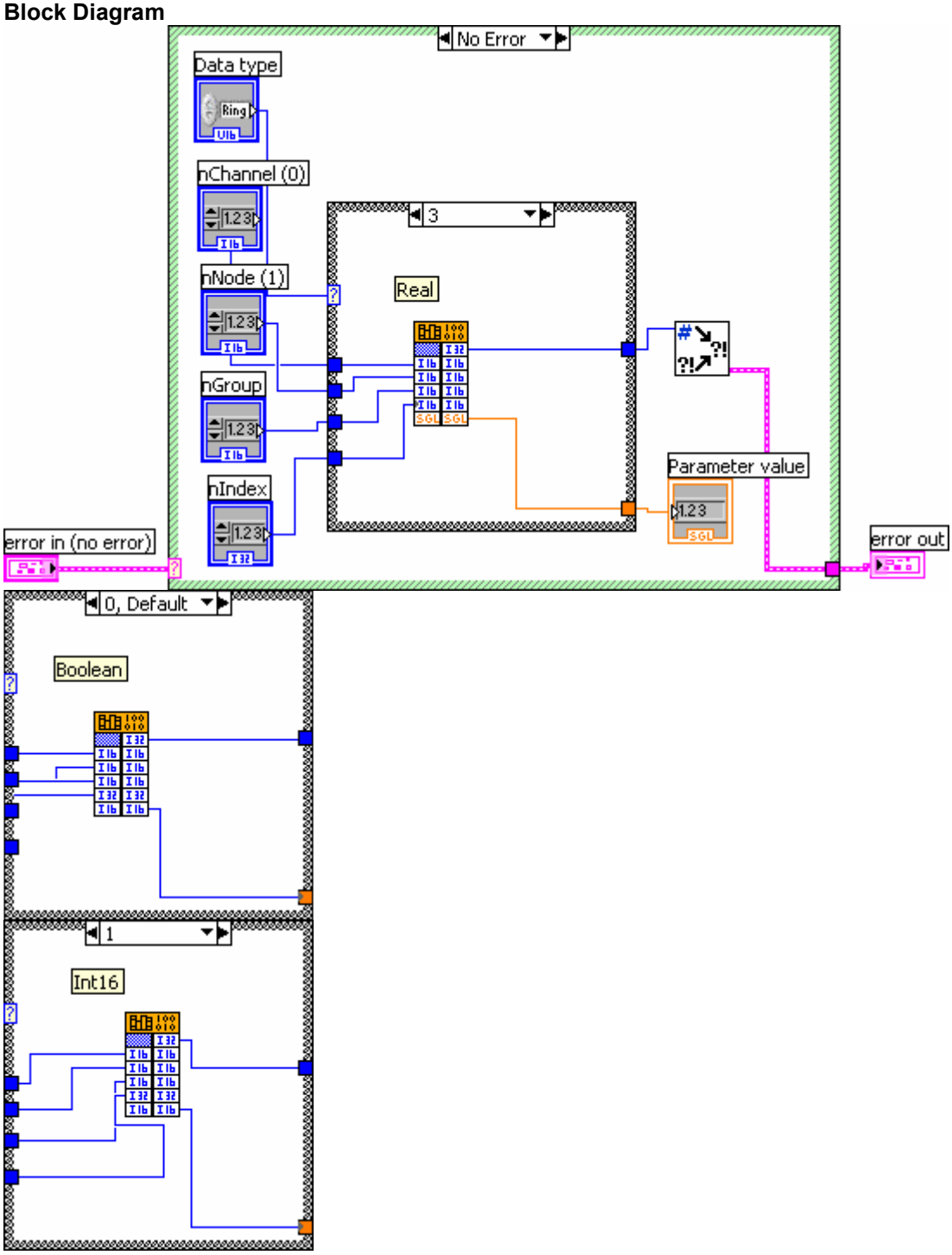


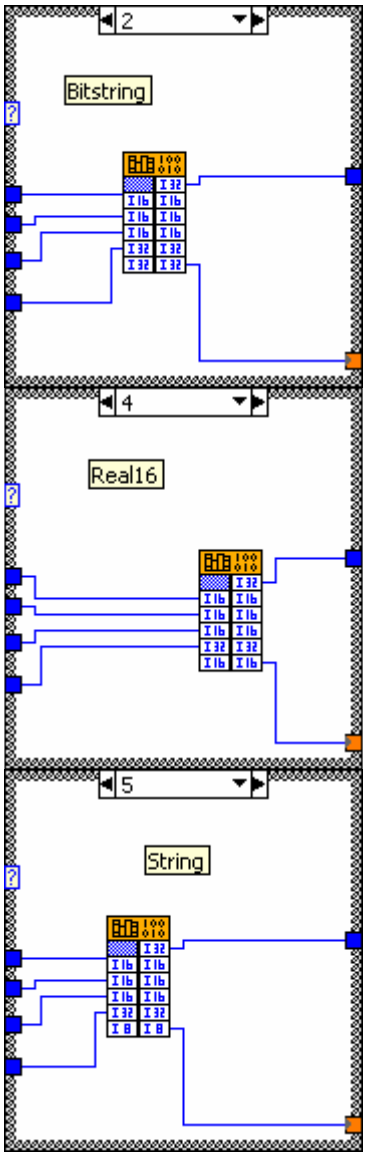
DD-ParamRead.vi
Reads a parameter.

Connector Pane



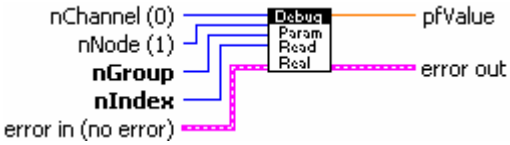
Block Diagram



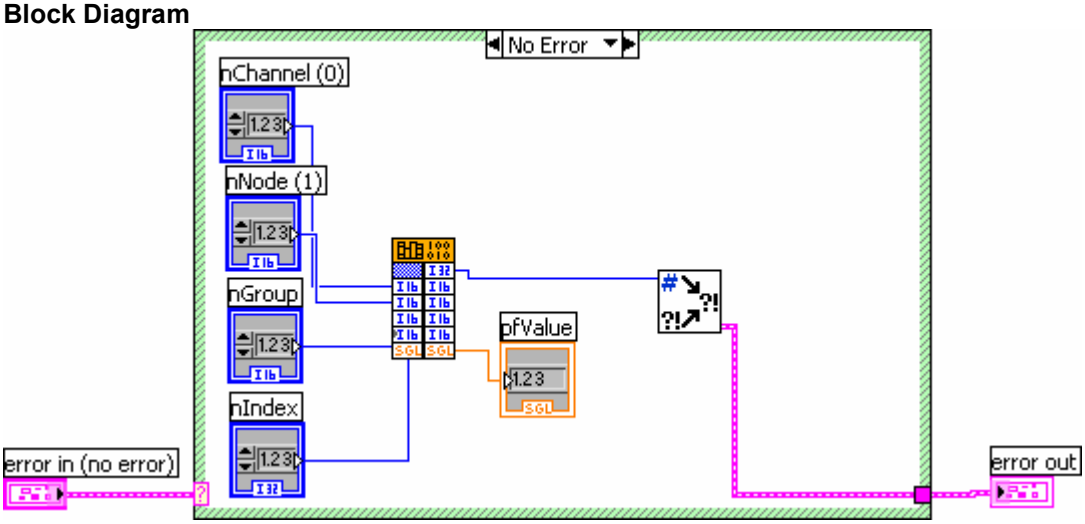

2
Bitstring
4
Real16
5
String

DD-ParamReadReal.vi
Reads a REAL parameter.

Connector Pane

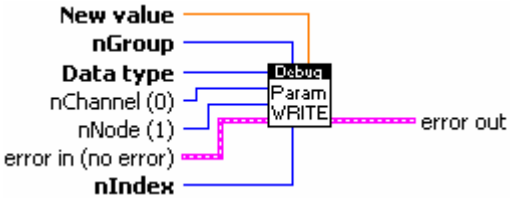


Block Diagram

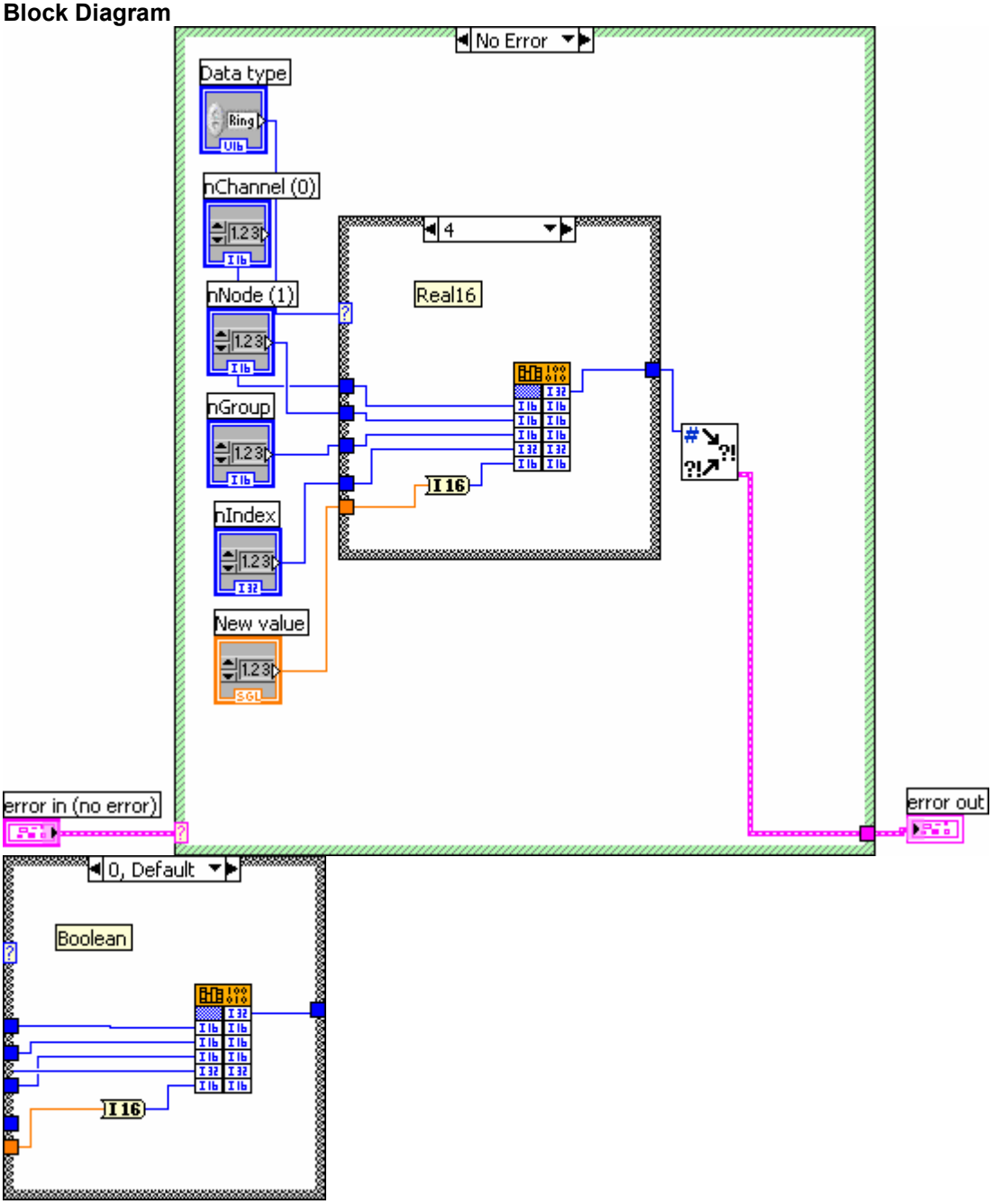


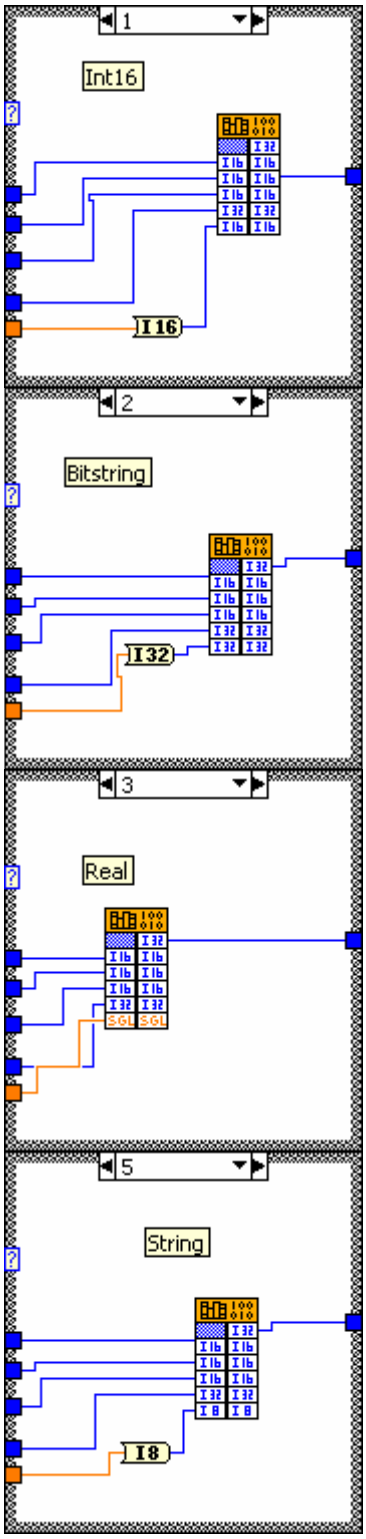
DD-ParamWrite.vi
Writes a new value to a parameter.

Connector Pane



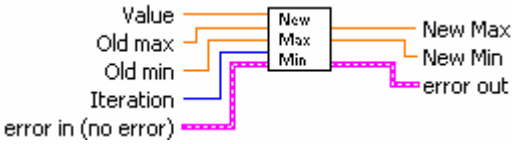
Block Diagram



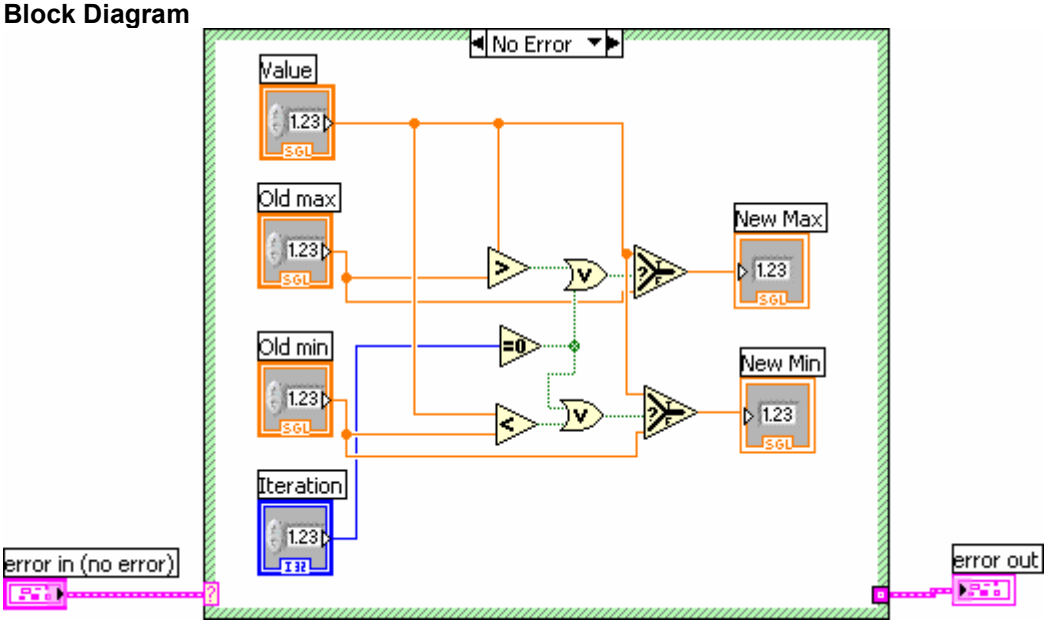

1
Int16
I16
2
Bitstring
I32
3
Real
5
String
I8

MaxMin-Old2New.vi
Compares value to old maximum and minimum value and returns new maximum and minimum values.

Connector Pane



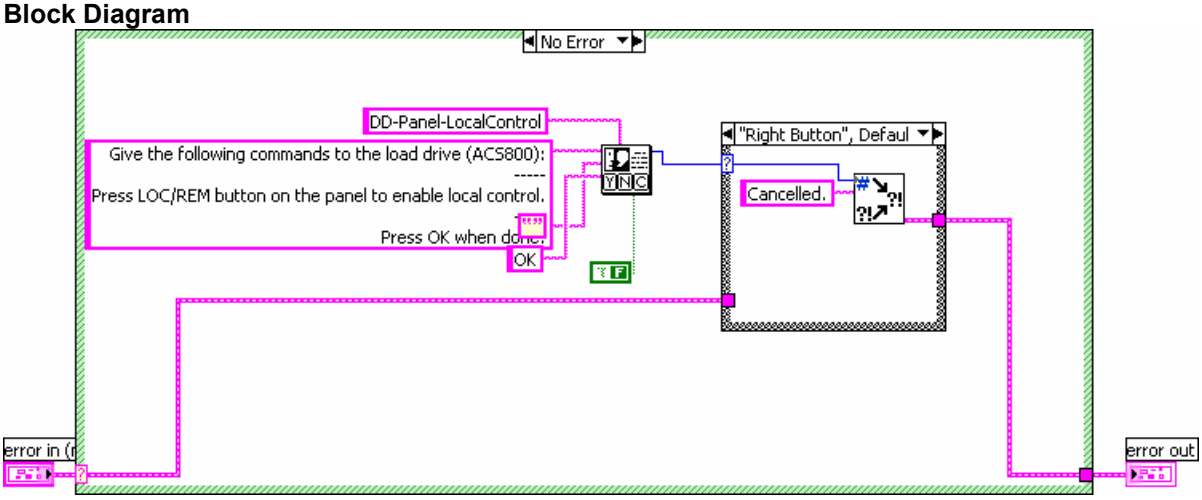
Block Diagram

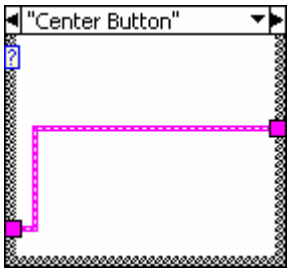


DD-Panel-LocalControl.vi
Displays a dialog to set the drive to local control mode through the control panel of the drive.

Connector Pane

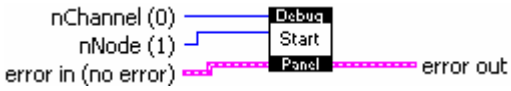
Block Diagram



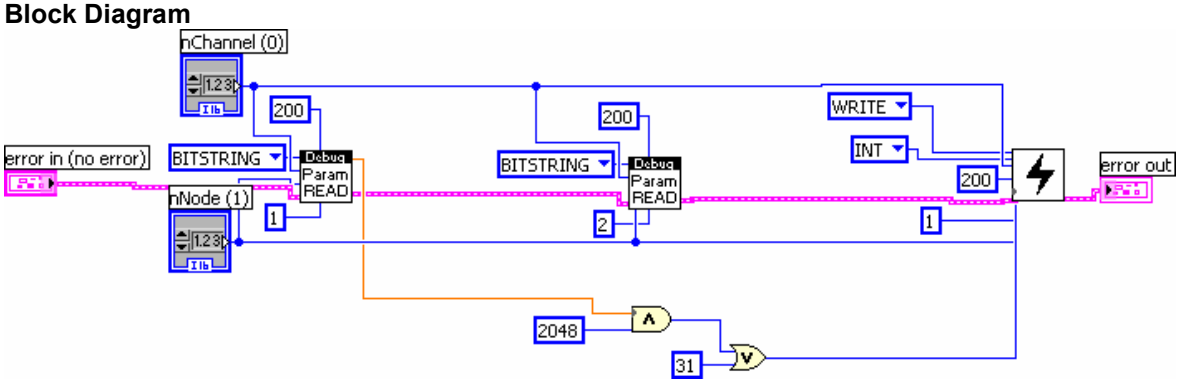


DD-Panel-Start.vi
Starts the motor through the control panel.

Connector Pane

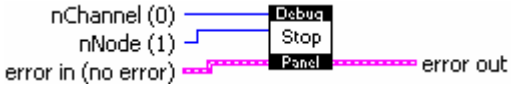


Block Diagram

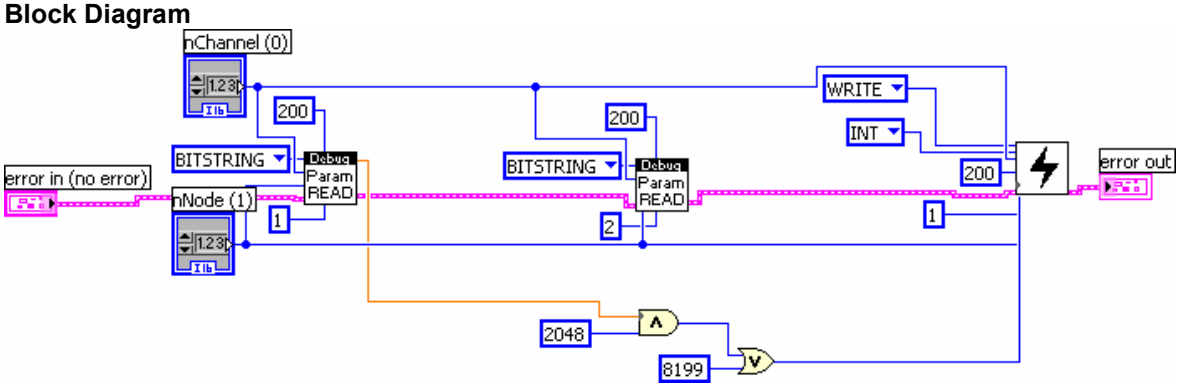


DD-Panel-Stop.vi
Stops the motor through the control panel.

Connector Pane

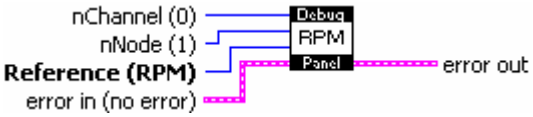


Block Diagram

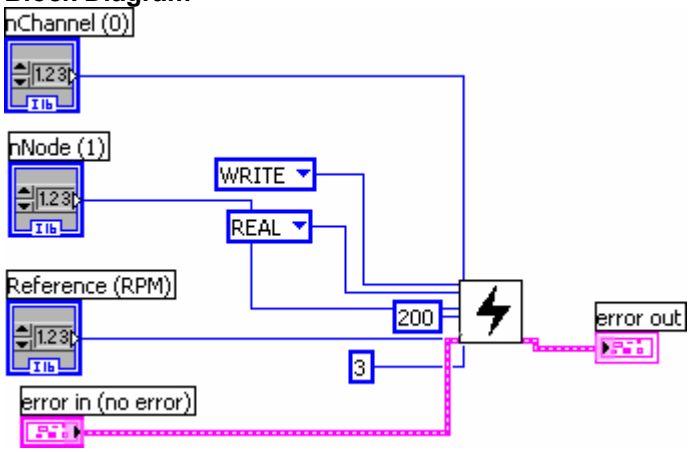


DD-Panel-Reference-RPM.vi
Sets a new speed reference through the control panel.

Connector Pane

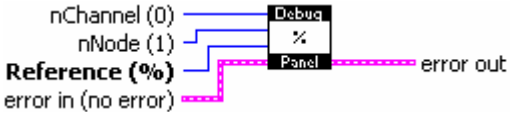


Block Diagram

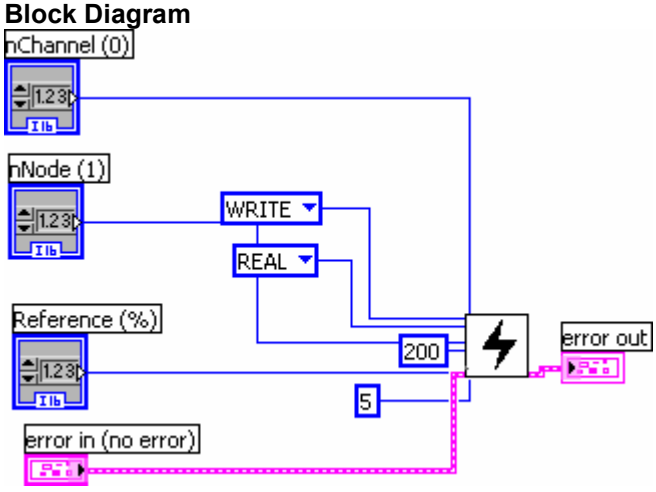


DD-Panel-Reference-Percent.vi
Sets a new torque reference through the control panel.

Connector Pane

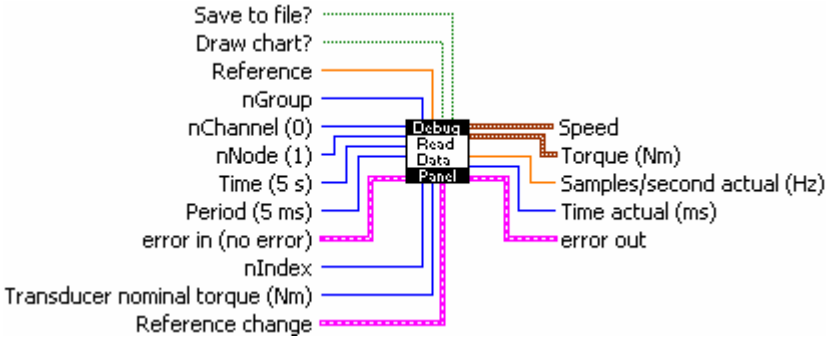


Block Diagram

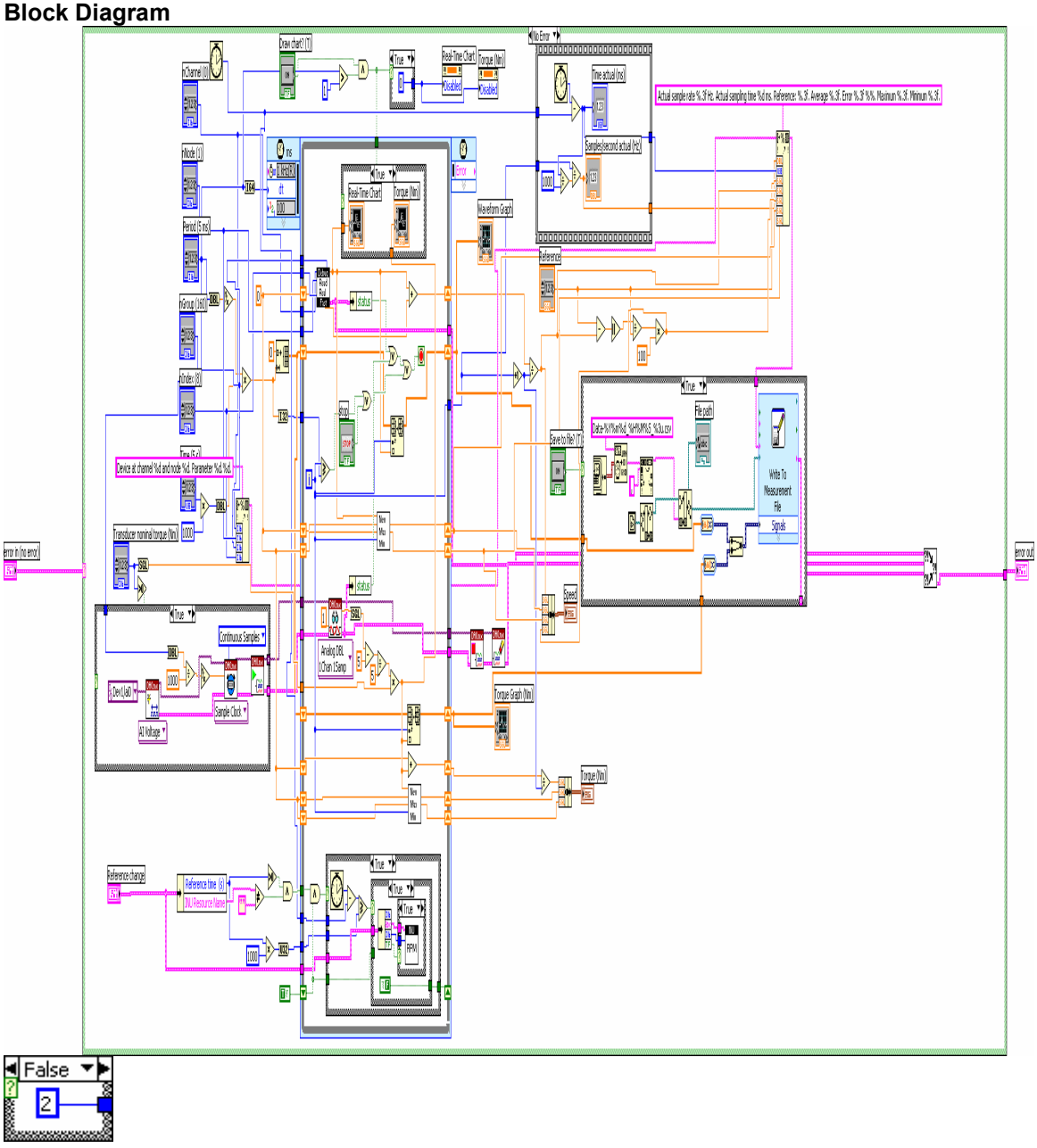


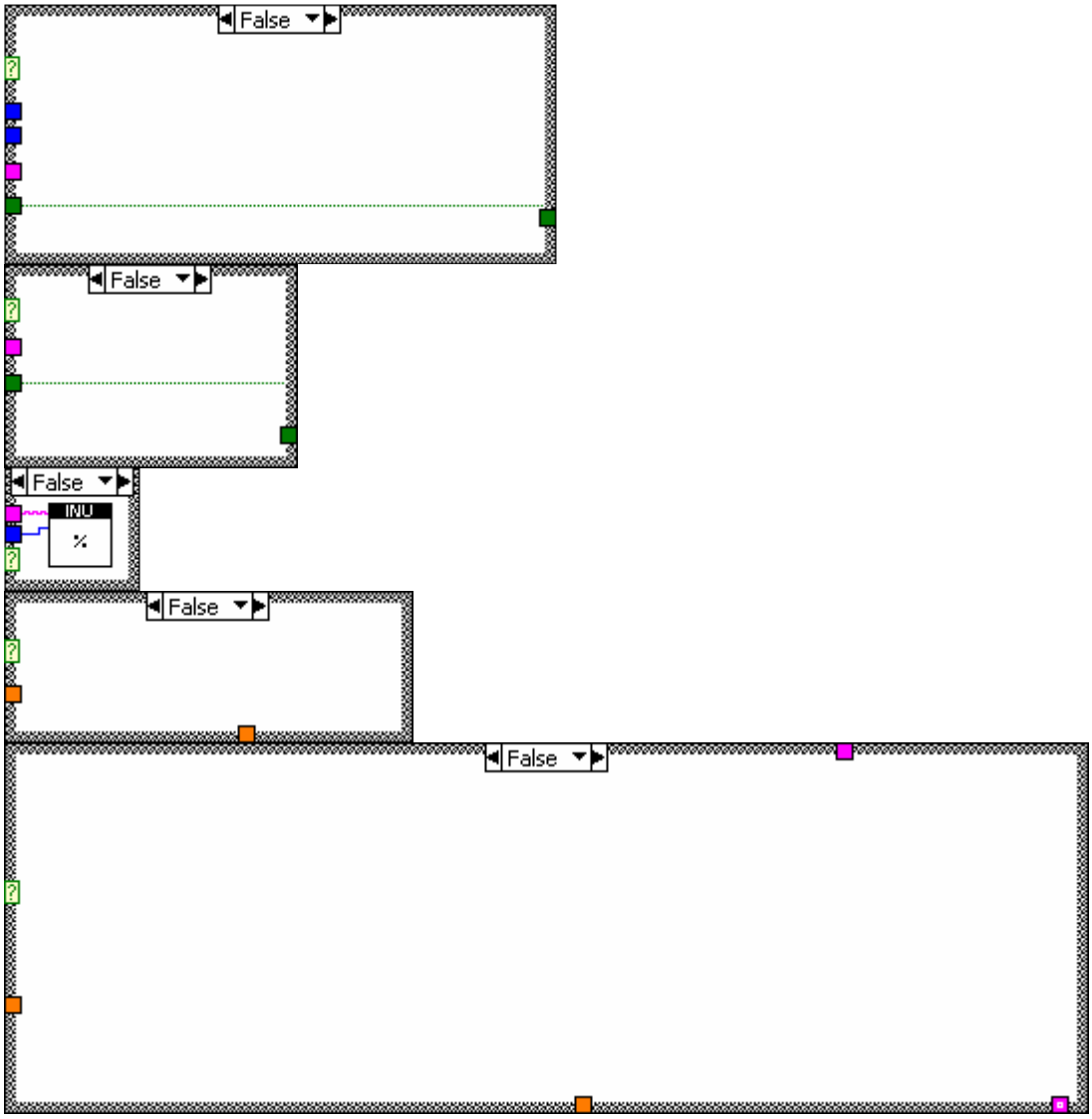
DD-Panel-ReadData-RealParam-Torque-Reference.vi
This VI reads the measured speed from ACS800 and torque from the torque transducer. It is also possible to change the speed or torque reference of INU specified by INU Resource Name through Reference change cluster.

Connector Pane



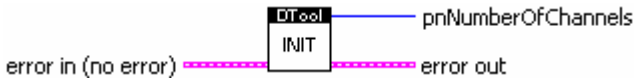
Block Diagram



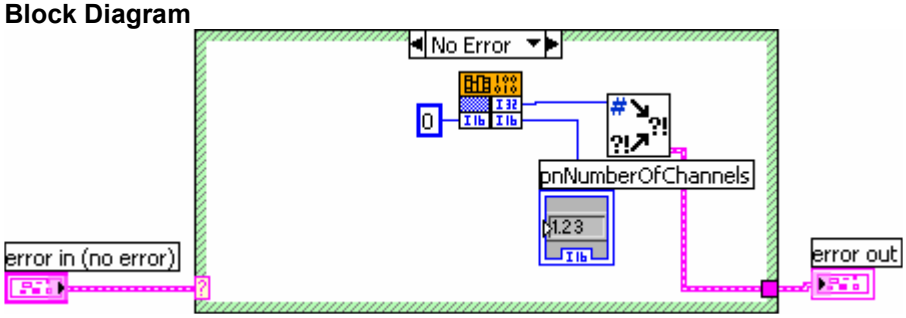


DT-InitCommunication.vi
Initializes the communication software and hardware. This function must be called before the use of other DTool functions. Returns the number of COM ports connected to drives (this value is always 1).

Connector Pane

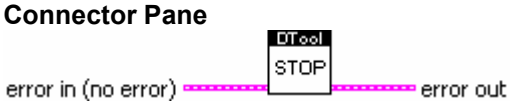


Block Diagram

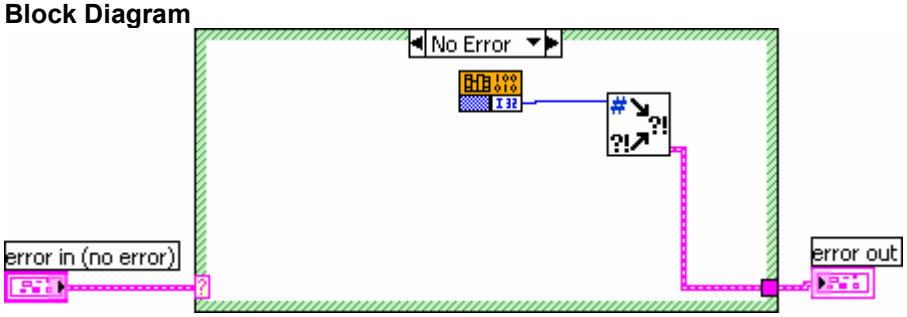


DT-StopCommunication.vi
Stops the use of the communication software and hardware. This function must be called after the use of other DTool functions.

Connector Pane

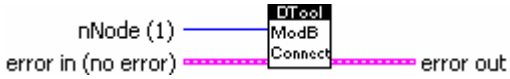


Block Diagram

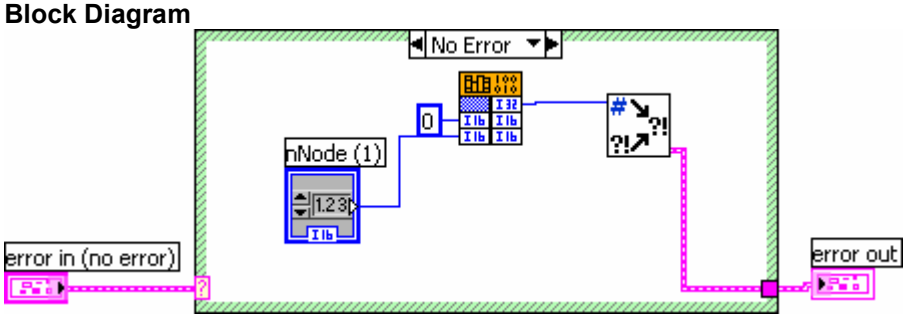


DT-ModBusConnect.vi
Initializes the Modbus communication with the drive. This function must be called before the use of all parameter functions.

Connector Pane

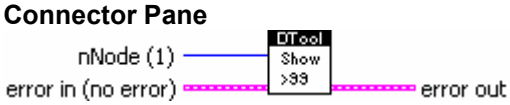


Block Diagram

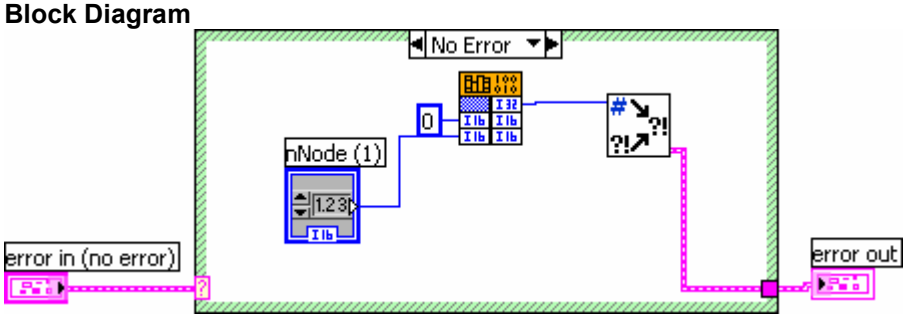


DT-ShowHiddenParameters.vi
Makes all hidden parameters visible.

Connector Pane

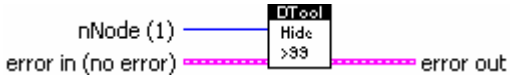


Block Diagram

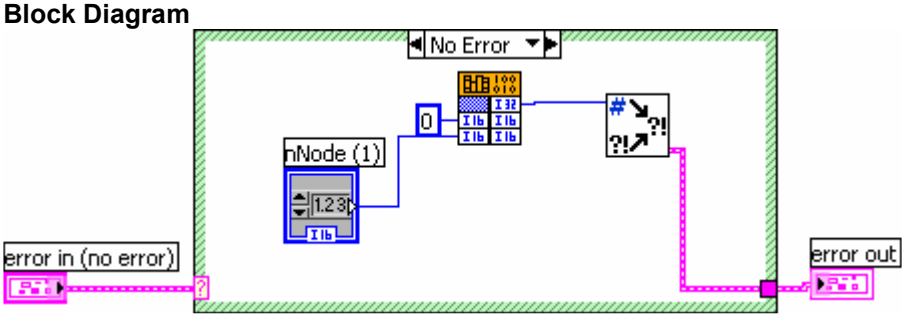


DT-HideHiddenParameters.vi
Hides all hidden parameters.

Connector Pane

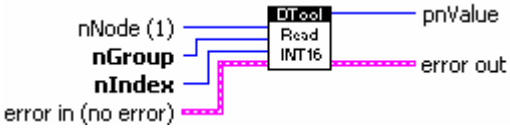


Block Diagram

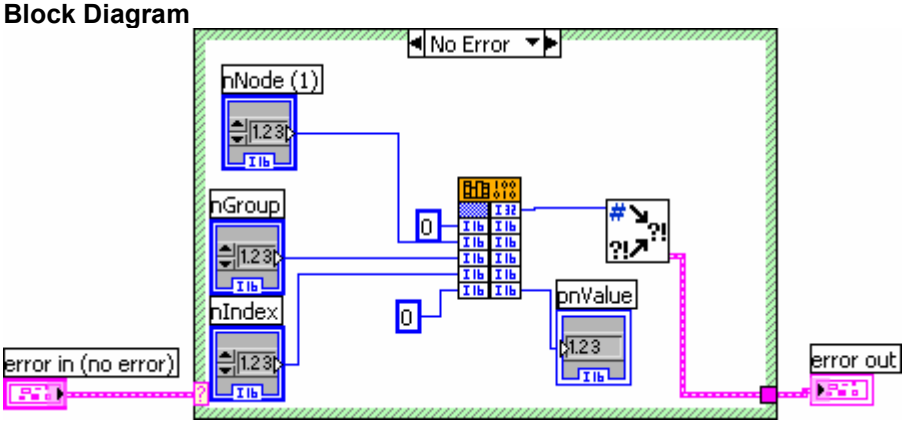


DT-ParamReadInt16.vi
Reads the value of an unscaled (signed or unsigned) 16-bit parameter.

Connector Pane

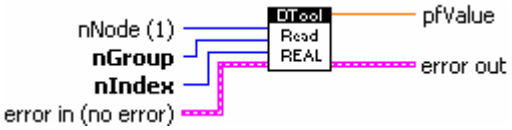


Block Diagram

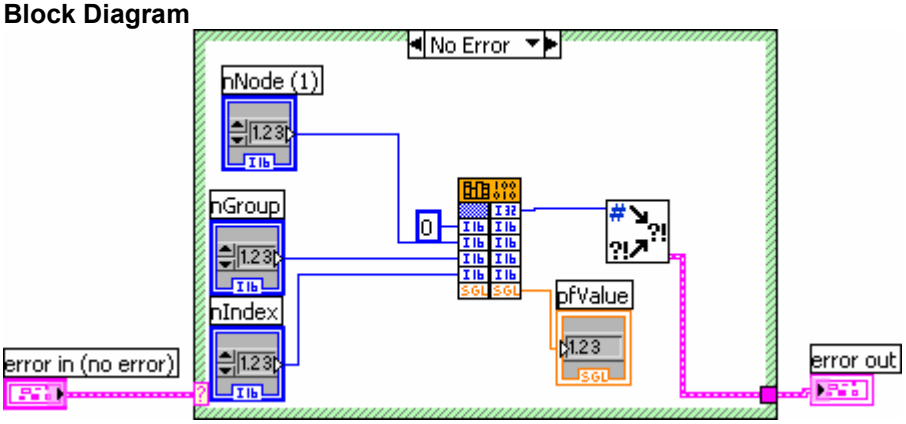


DT-ParamReadReal.vi
Reads the value (converted to real) of a scaled 16-bit parameter.

Connector Pane

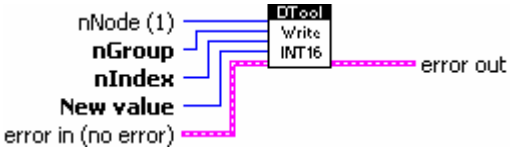


Block Diagram

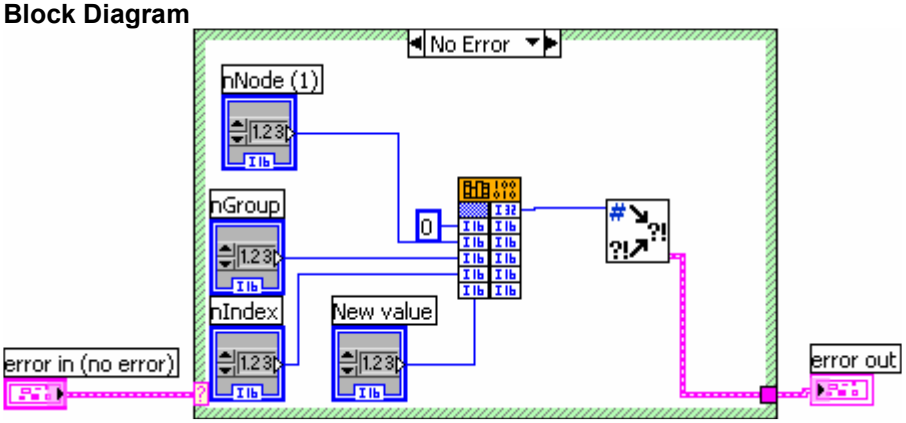


DT-ParamWriteInt16.vi
Writes the new value of an unscaled (signed or unsigned) 16-bit parameter.

Connector Pane

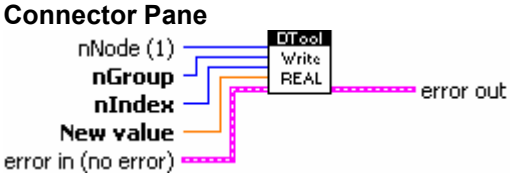


Block Diagram

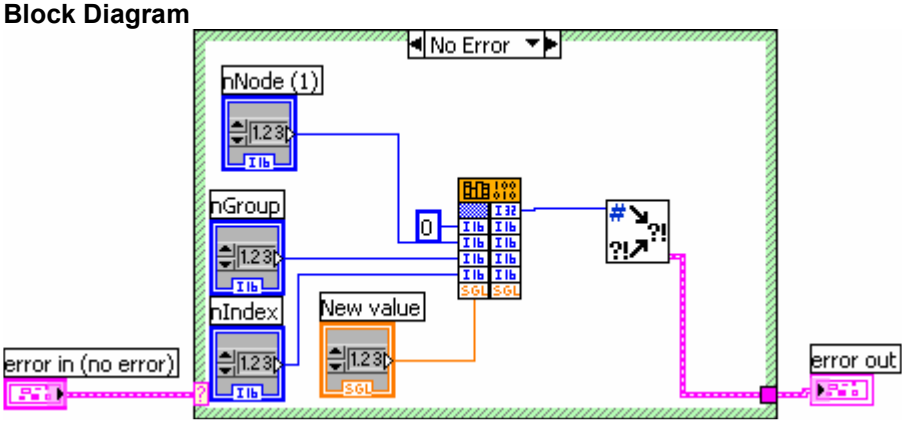


DT-ParamWriteReal.vi
Writes the new value (given as real) of a scaled 16-bit parameter.

Connector Pane

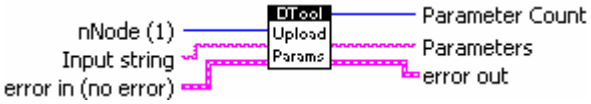


Block Diagram

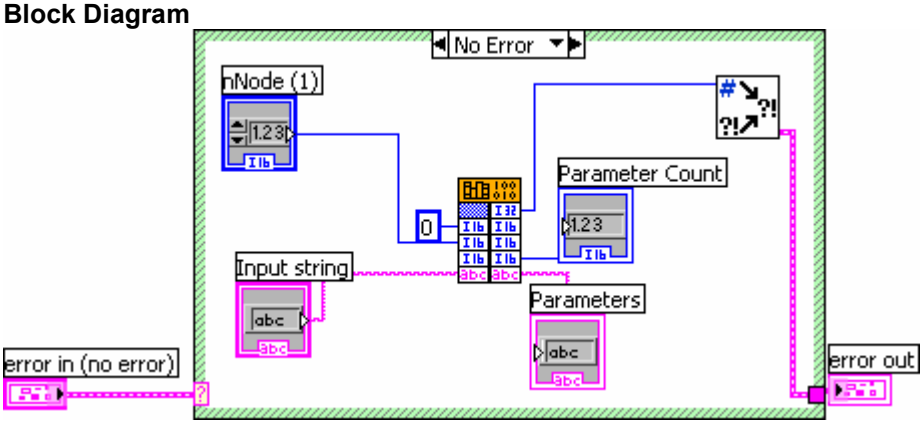


DT-UploadParamDefinitions.vi
Uploads the parameter definition list from the drive. Input string must be big enough to hold all parameters.

Connector Pane

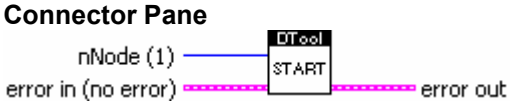


Block Diagram

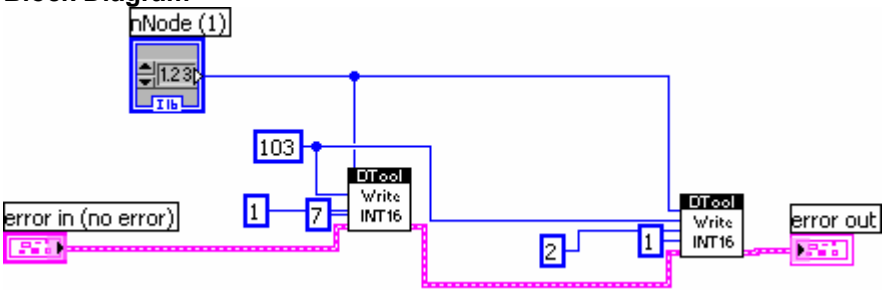


DT-Start.vi
Starts the motor.

Connector Pane

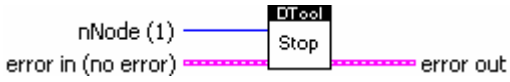


Block Diagram

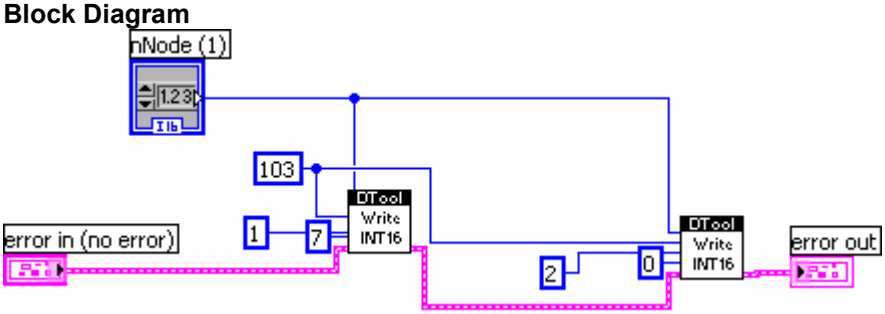


DT-Stop.vi
Stops the motor.

Connector Pane

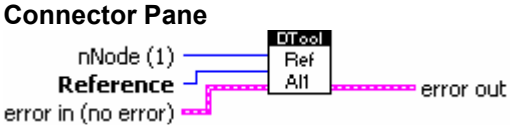


Block Diagram

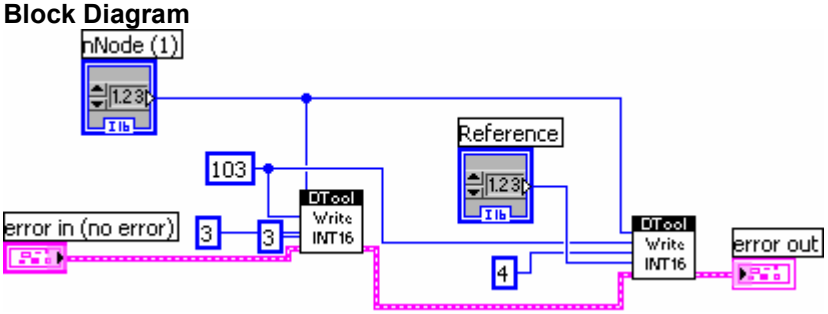


DT-Reference-AI1.vi
Sets a new speed reference through analog input 1.

Connector Pane

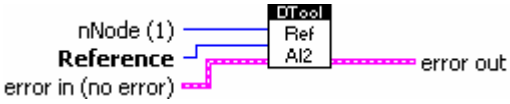


Block Diagram

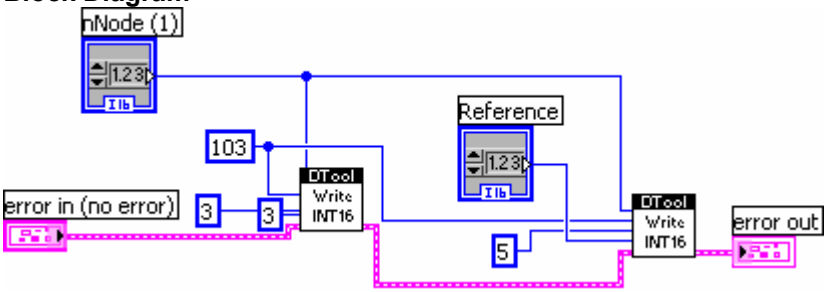


DT-Reference-AI2.vi
Sets a new torque reference through analog input 2.

Connector Pane



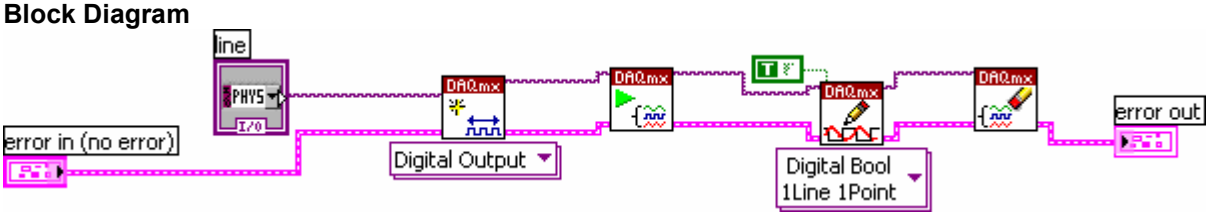
Block Diagram



IO-Start.vi
Sets the specified digital output line to TRUE state.

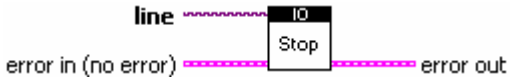
Connector Pane

Block Diagram

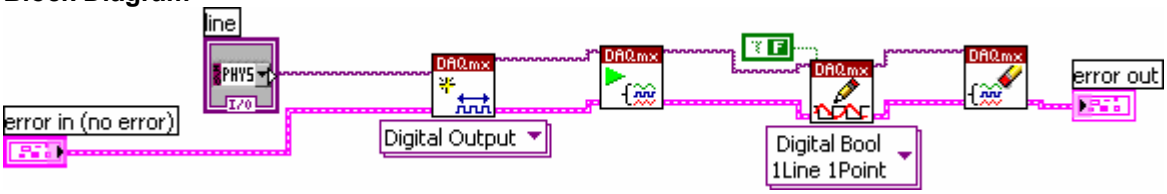


IO-Stop.vi
Sets the specified digital output line to FALSE state.

Connector Pane

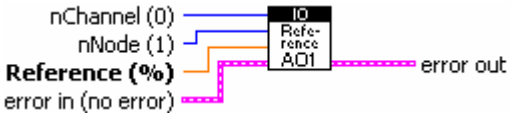


Block Diagram

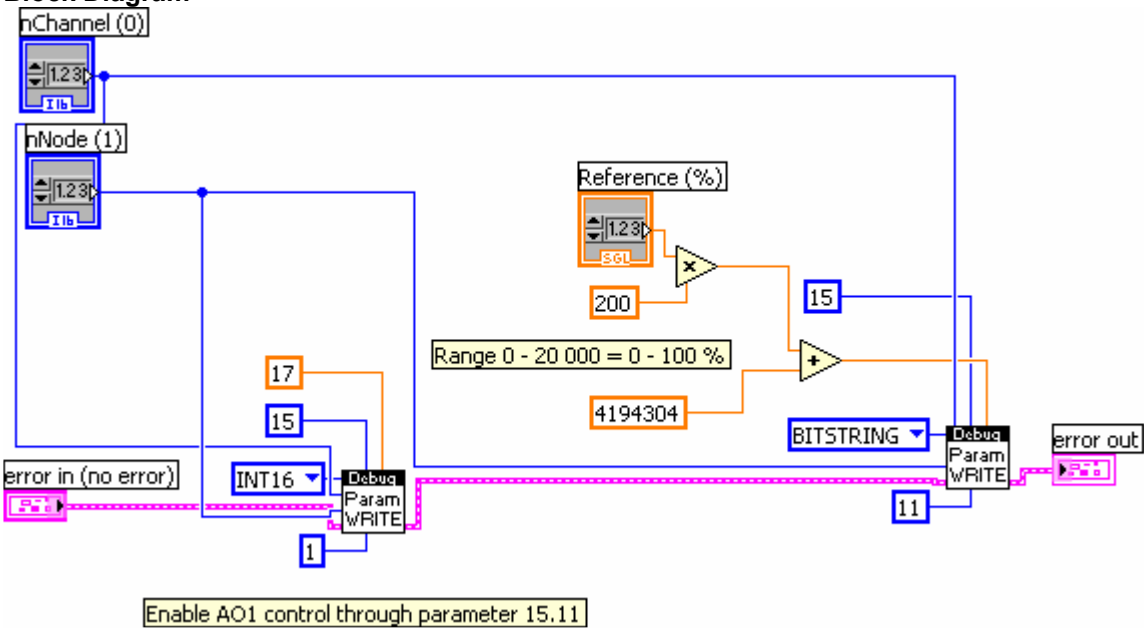


IO-Reference-AO1.vi
Sets a new speed reference through analog output 1 of ACS800.

Connector Pane

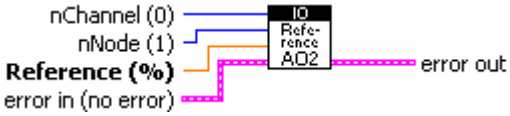


Block Diagram

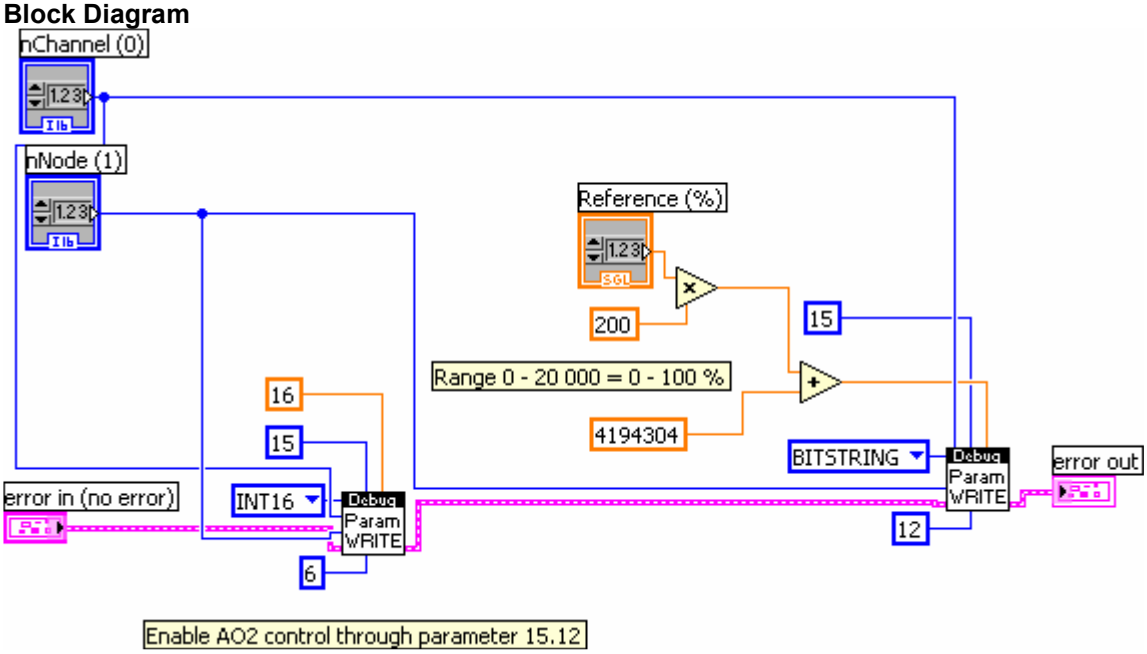


IO-Reference-AO2.vi
Sets a new torque reference through analog output 2 of ACS800.

Connector Pane



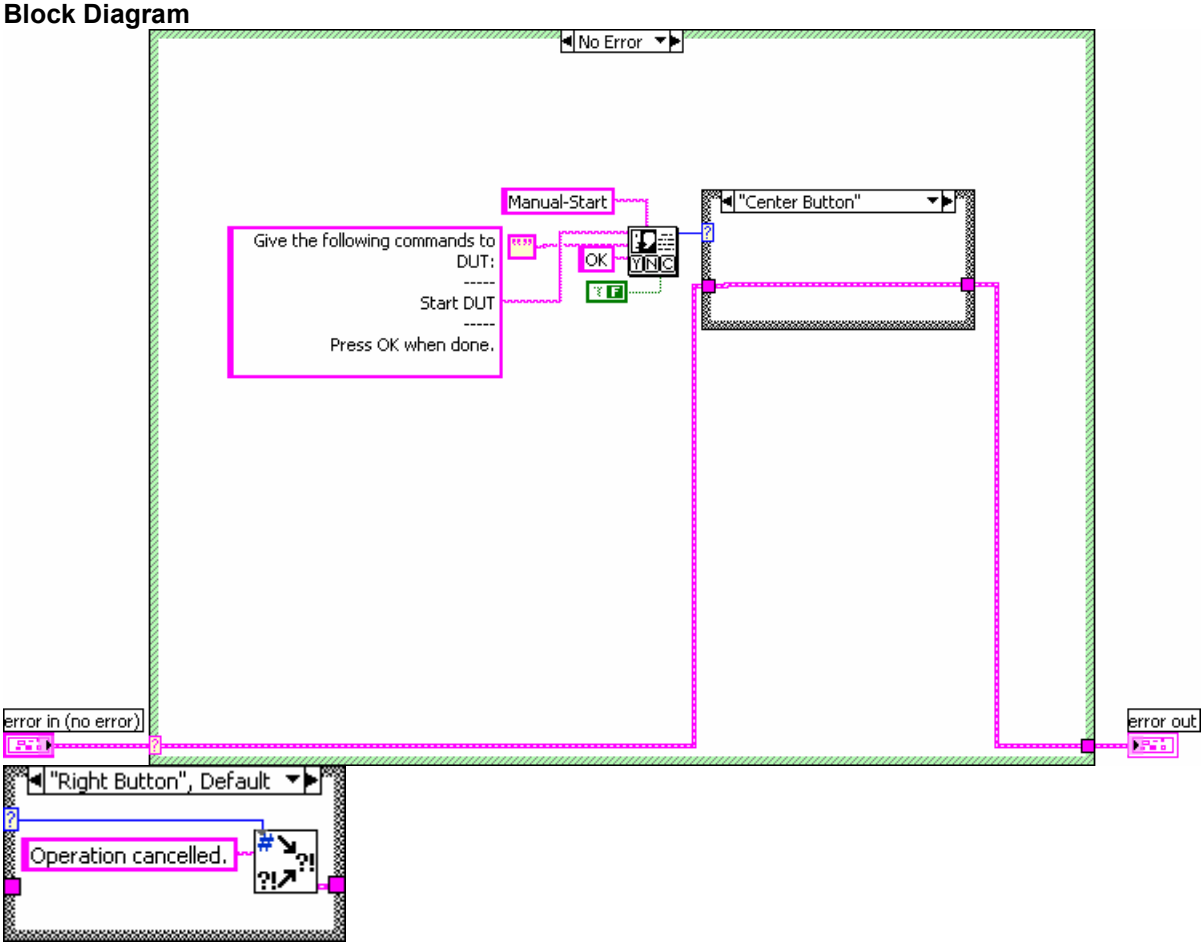
Block Diagram



Manual-Start.vi
Displays a dialog to start DUT.

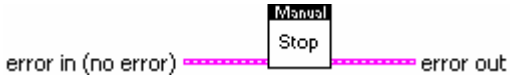
Connector Pane

Block Diagram

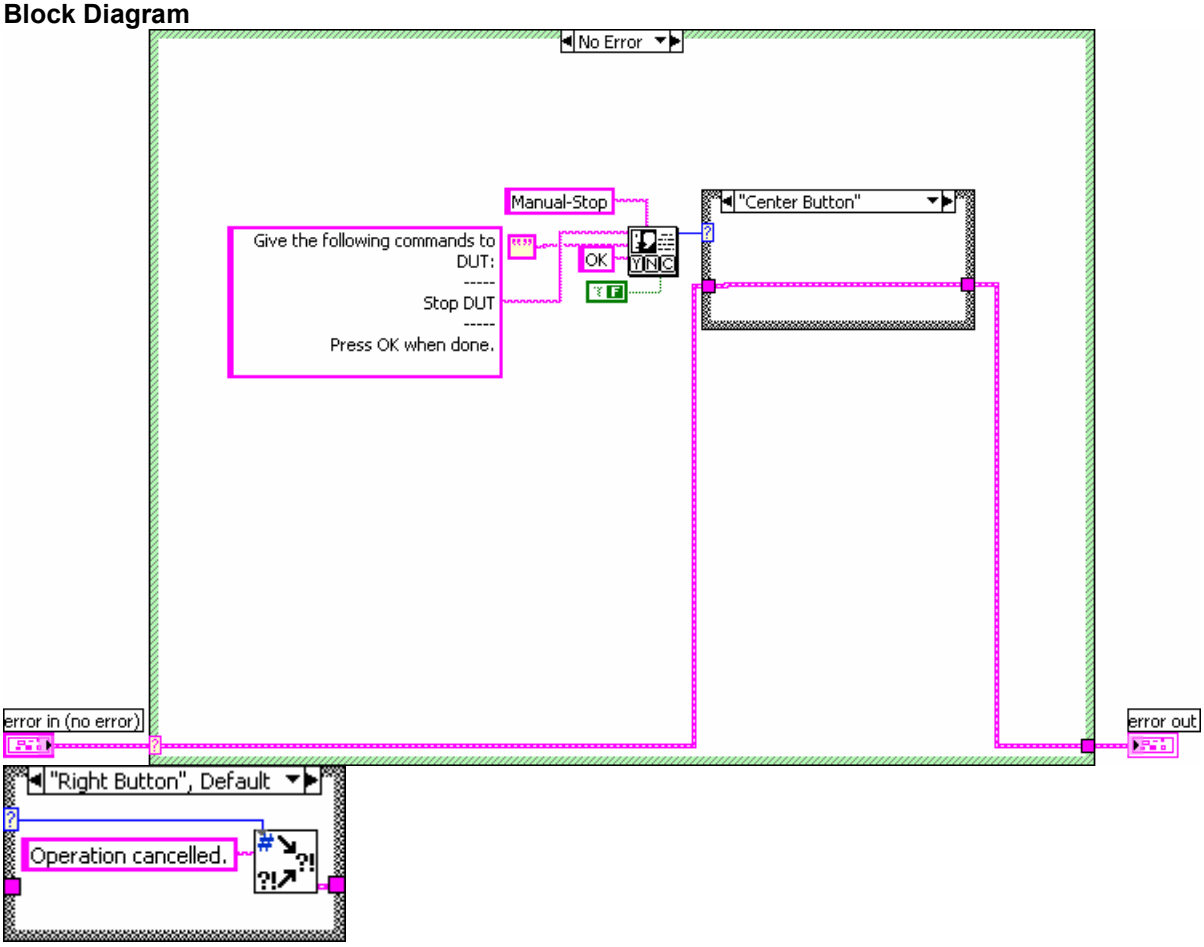


Manual-Stop.vi
Displays a dialog to stop DUT.

Connector Pane

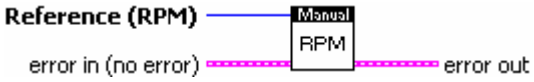


Block Diagram

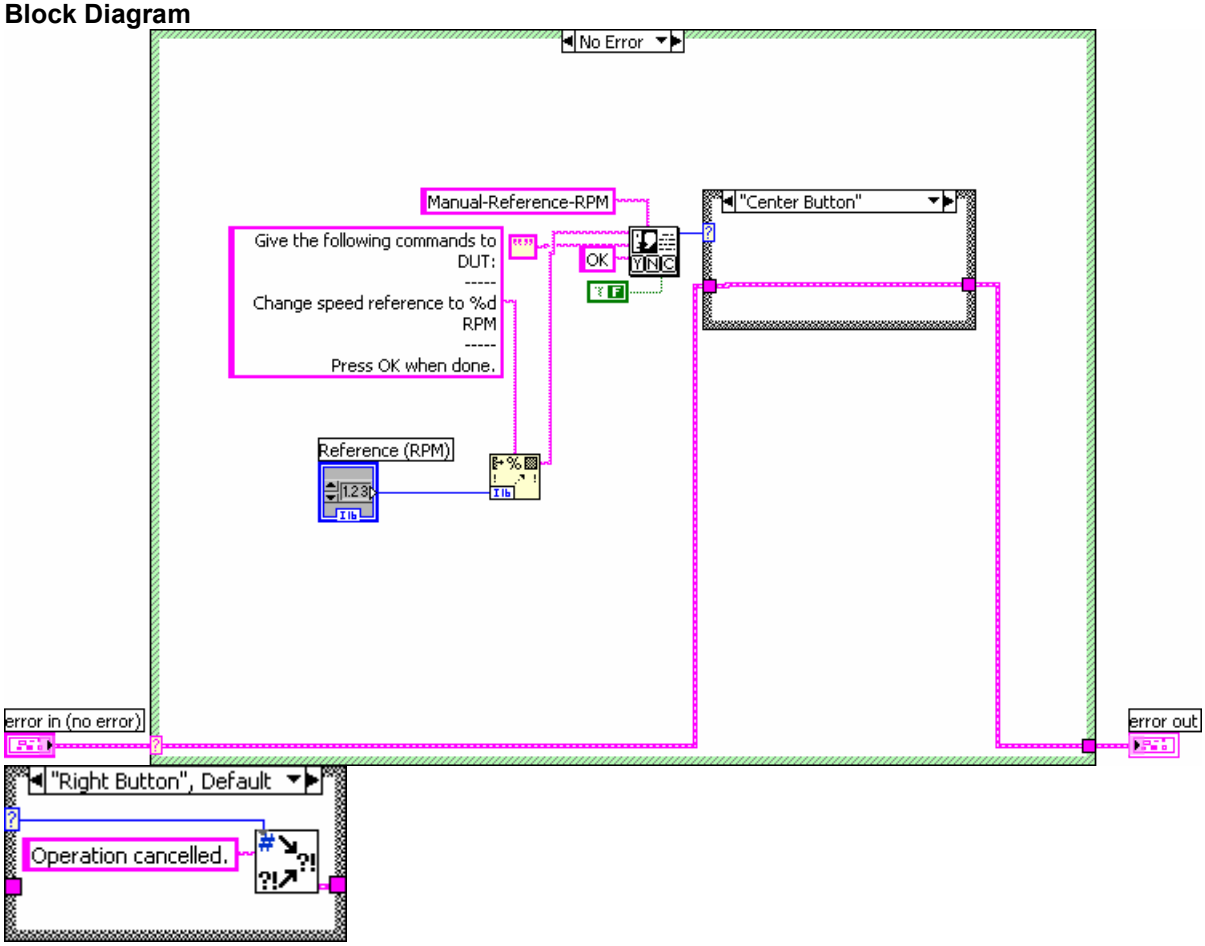


Manual-Reference-RPM.vi
Displays a dialog to set a new speed reference to DUT.

Connector Pane

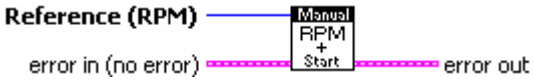


Block Diagram

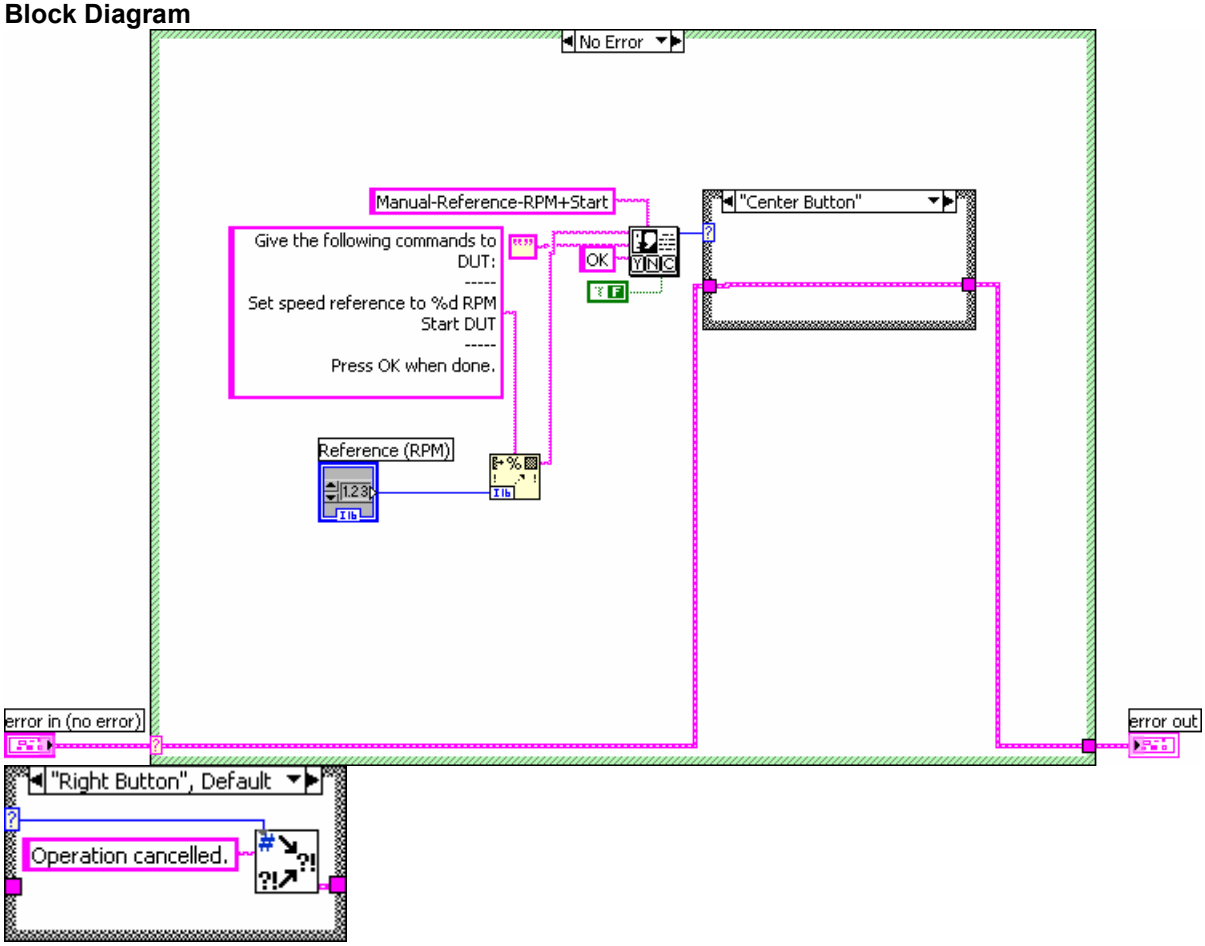


Manual-Reference-RPM+Start.vi
Displays a dialog to set a new speed reference and start DUT.

Connector Pane



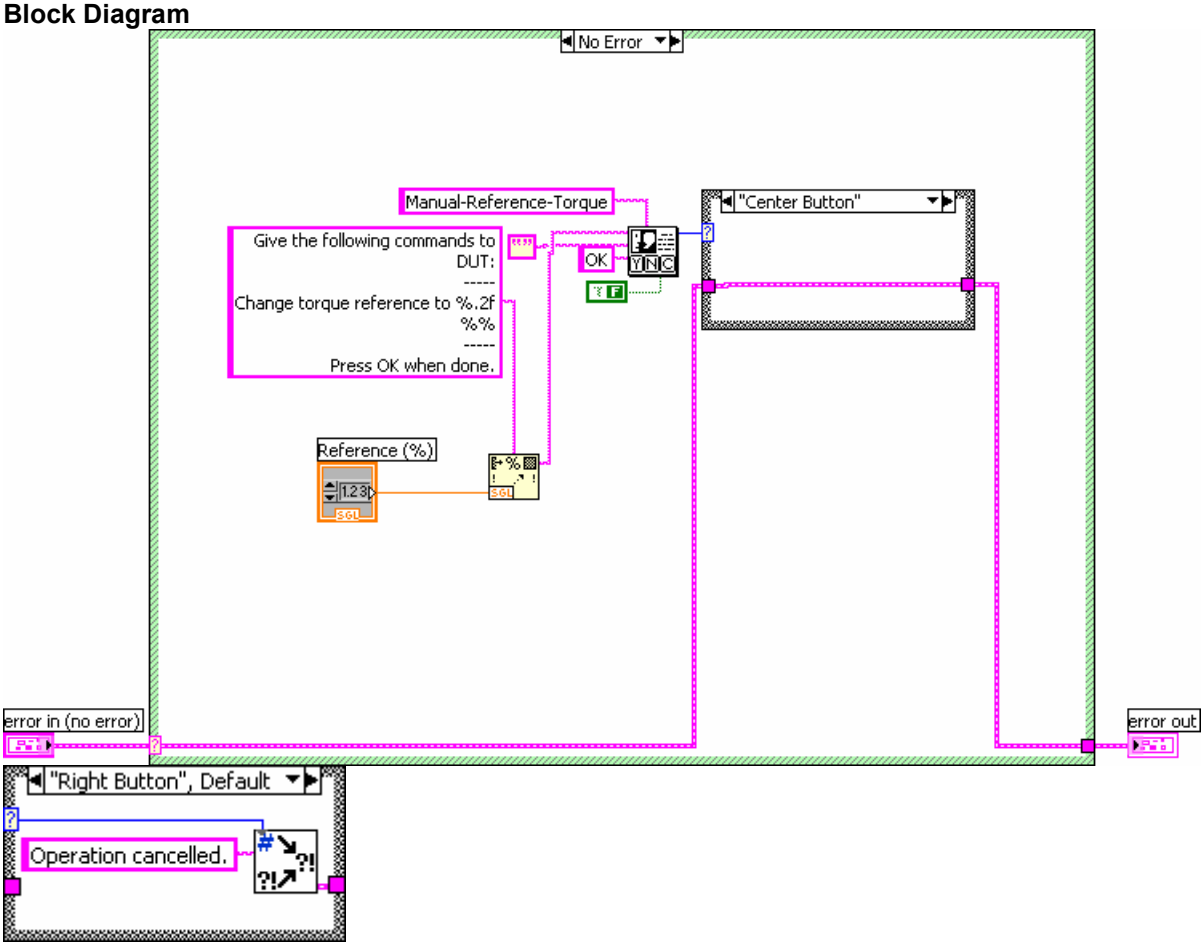
Block Diagram



Manual-Reference-Torque.vi
Displays a dialog to set a new torque reference to DUT.

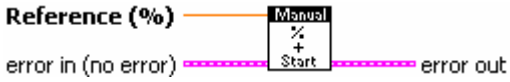
Connector Pane

Block Diagram

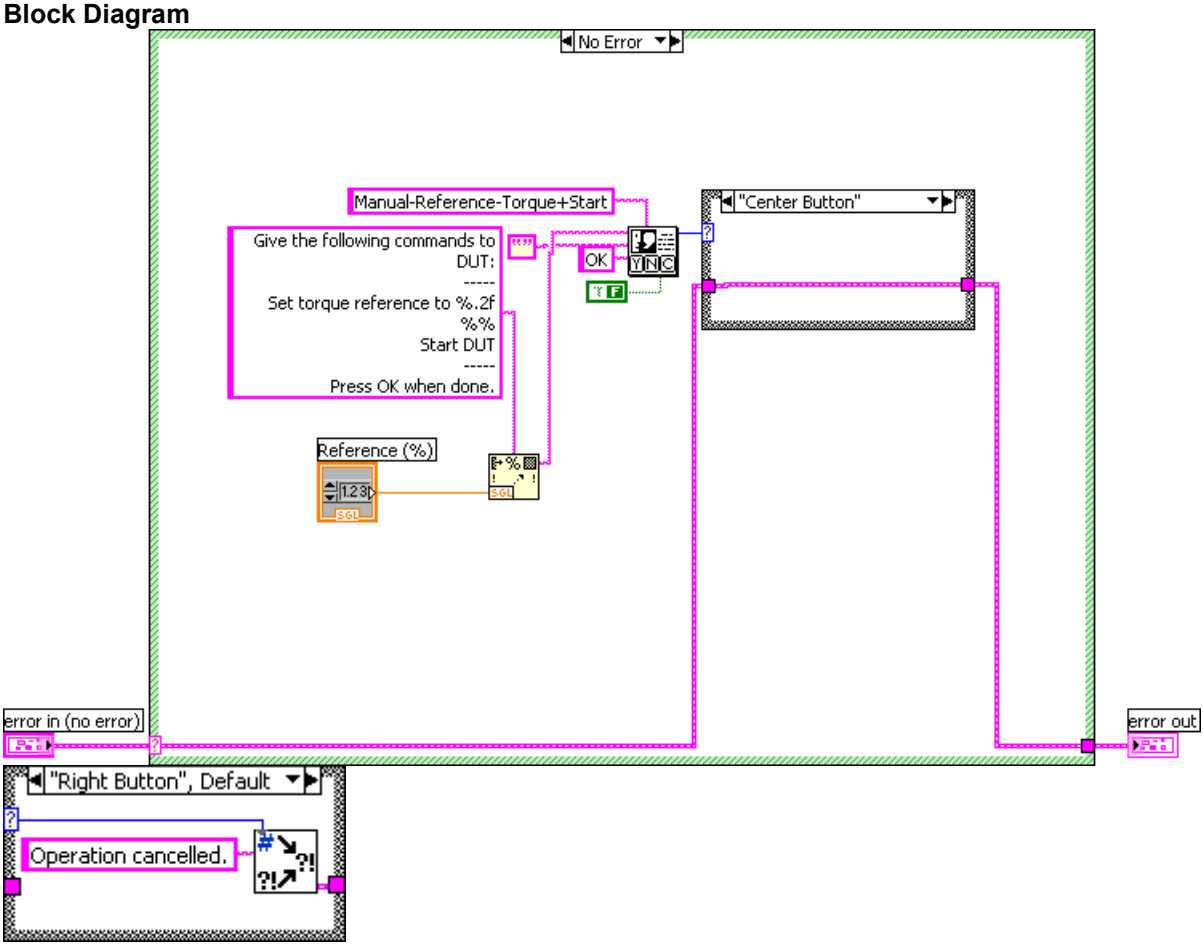


Manual-Reference-Torque+Start.vi
Displays a dialog to set a new torque reference and start DUT.

Connector Pane

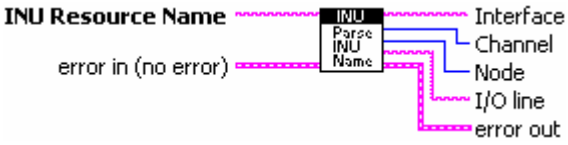


Block Diagram

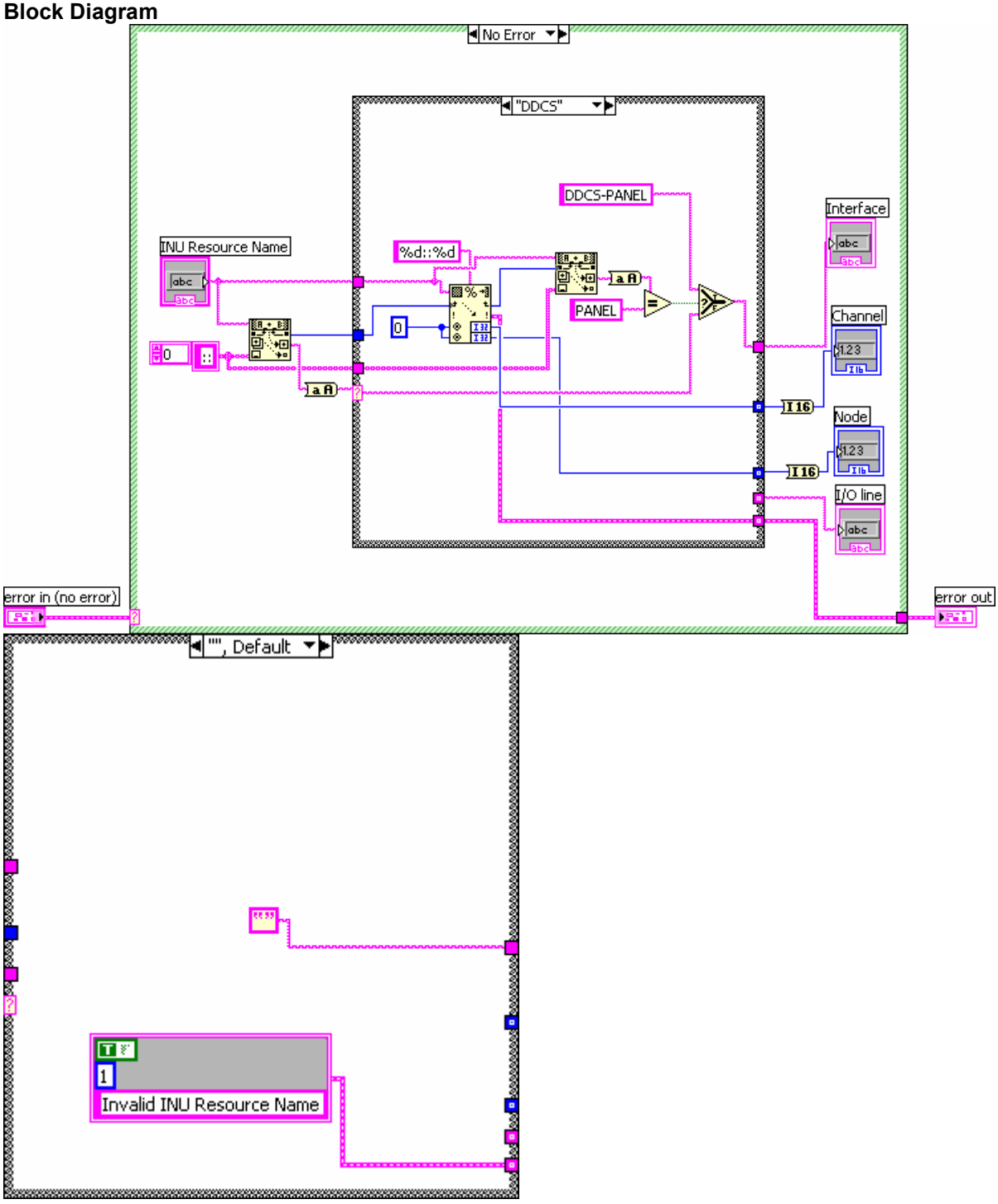


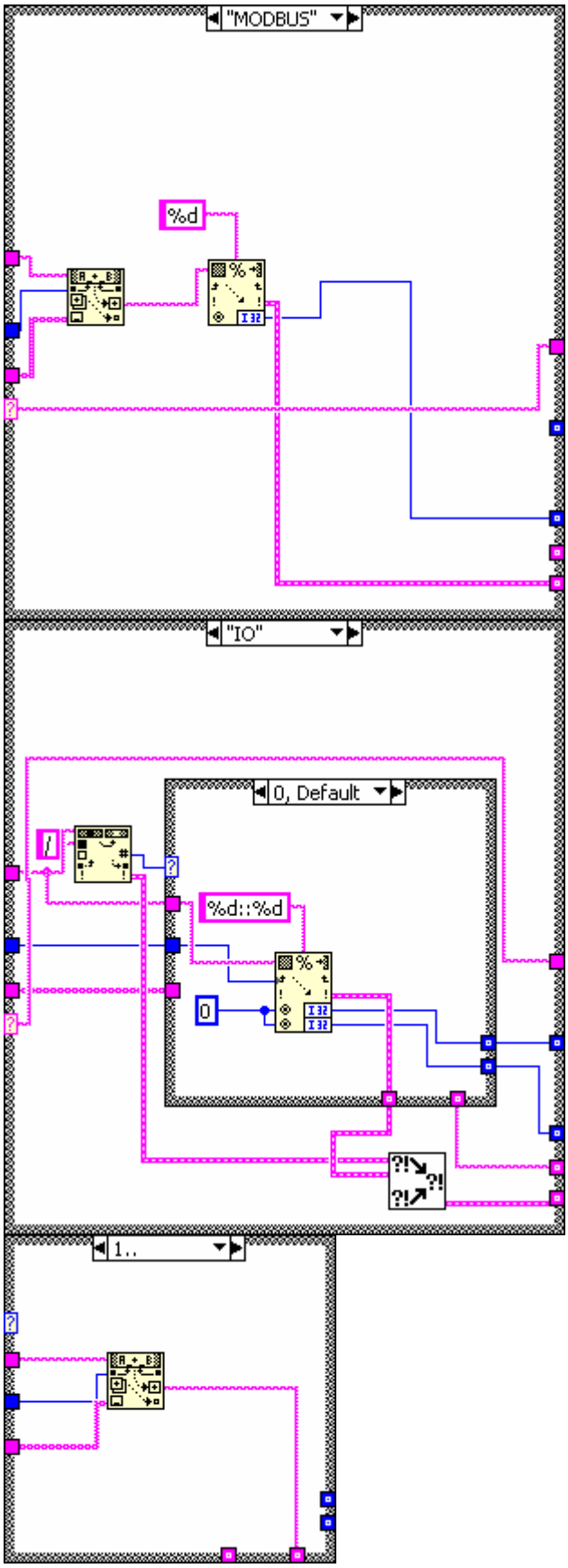
INU-ParseResourceName.vi
Parses the INU Resource Name.

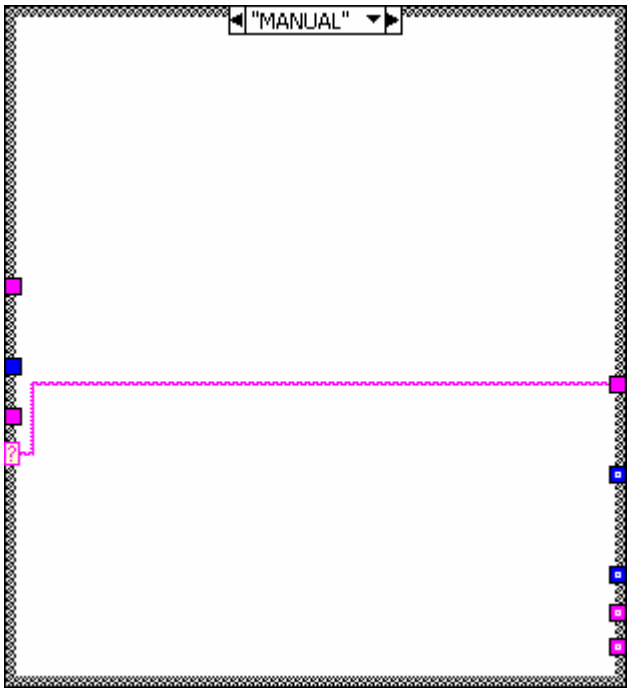
Connector Pane



Block Diagram

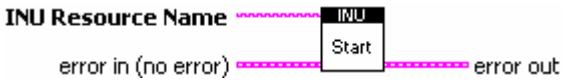



"MODBUS"
%d
"IO"
0, Default
%d::%d
0
1..

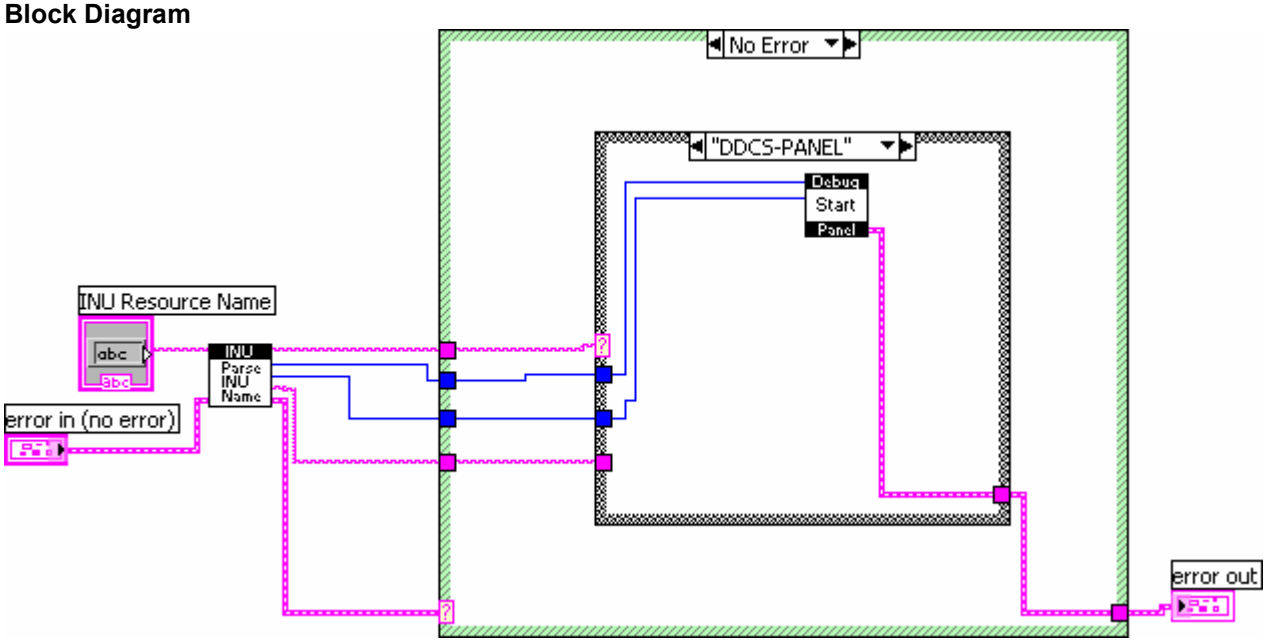


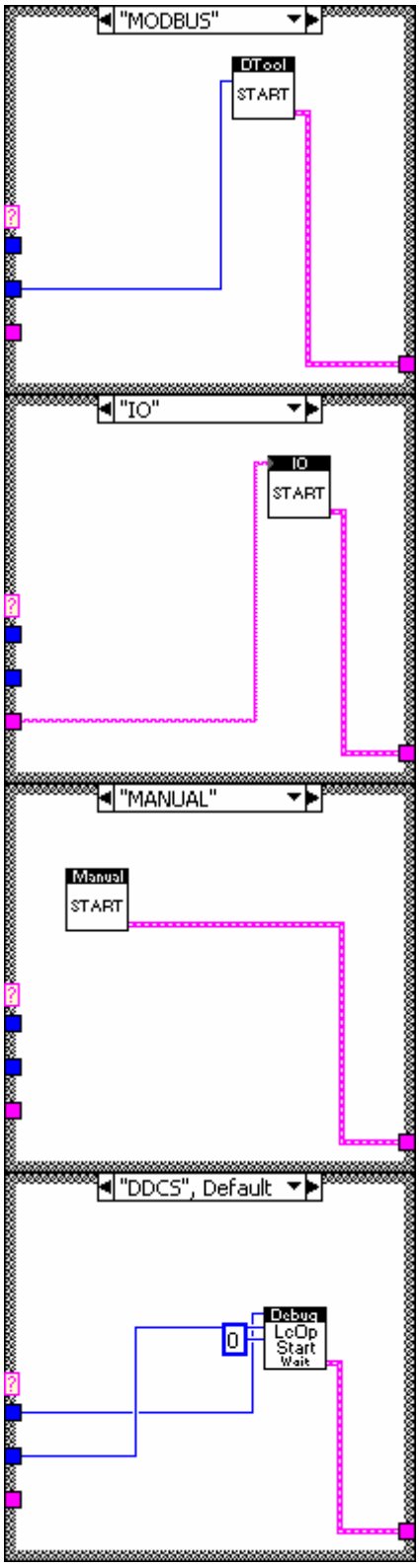
INU-Start.vi
Starts the motor of INU specified by INU Resource Name.

Connector Pane



Block Diagram

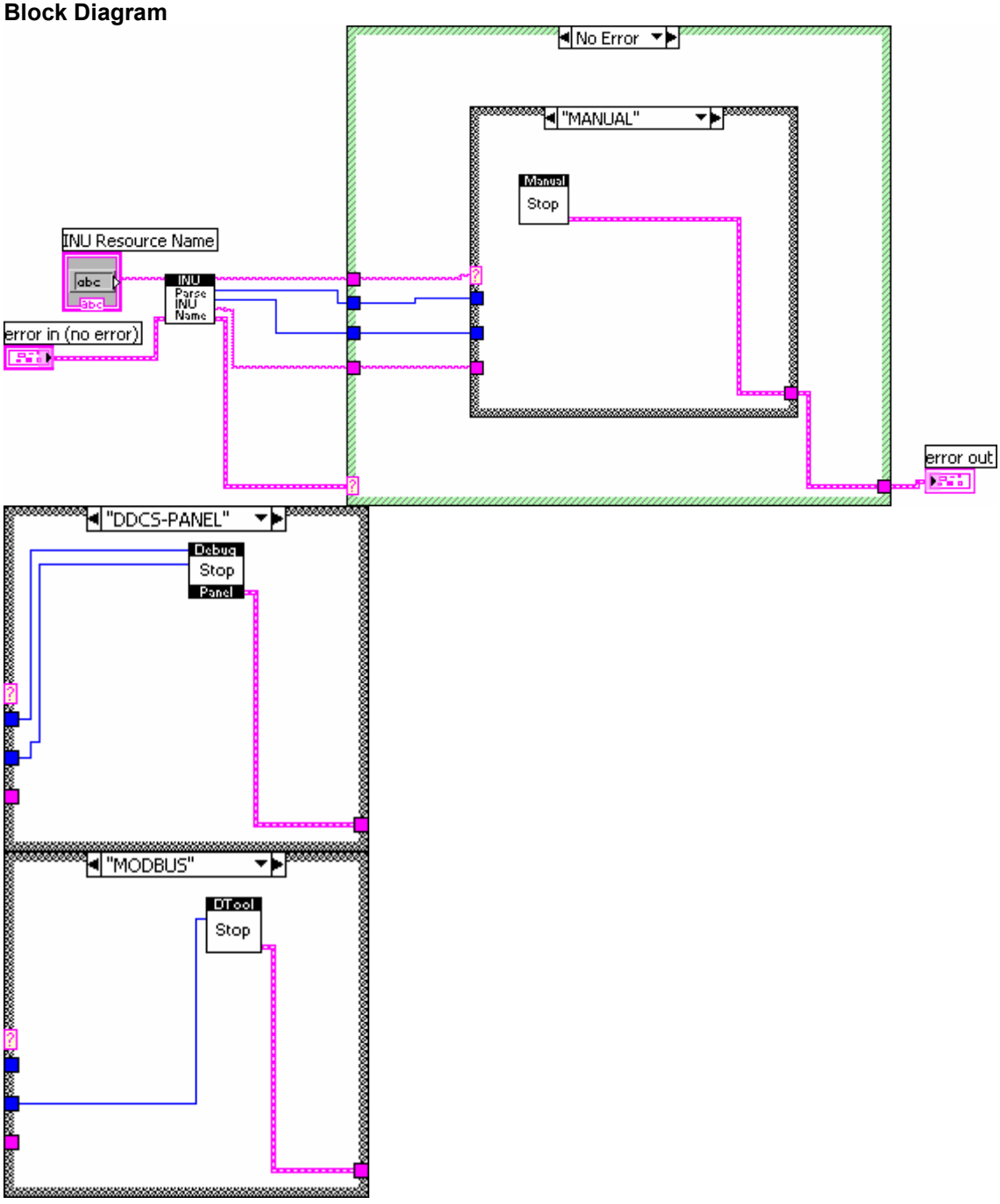


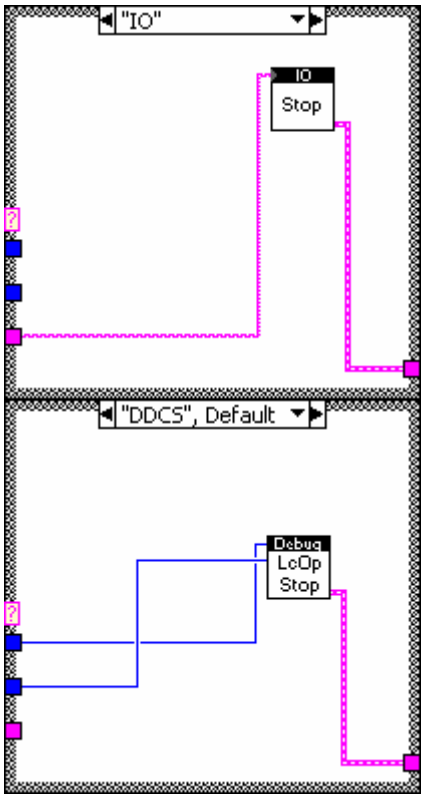

"MODBUS"
DTool
START
"IO"
IO
START
"MANUAL"
Manual
START
"DDCS", Default
0
Debug
LcOp
Start
Wait

INU-Stop.vi
Stops the motor of INU specified by INU Resource Name.

Connector Pane

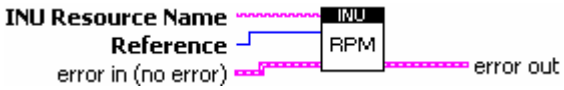
Block Diagram



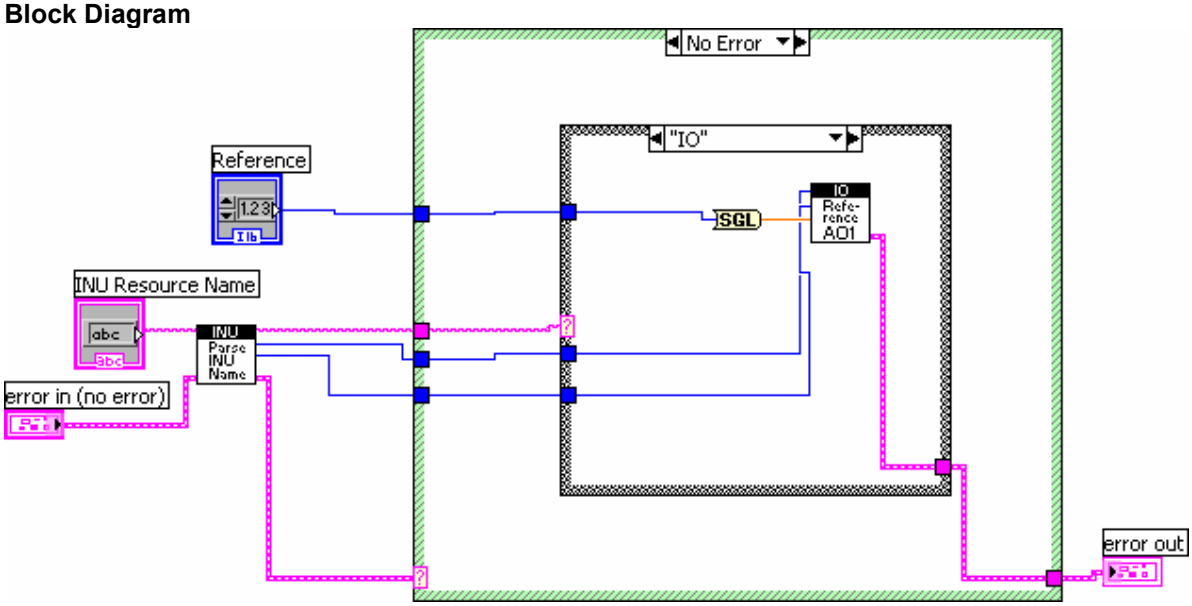


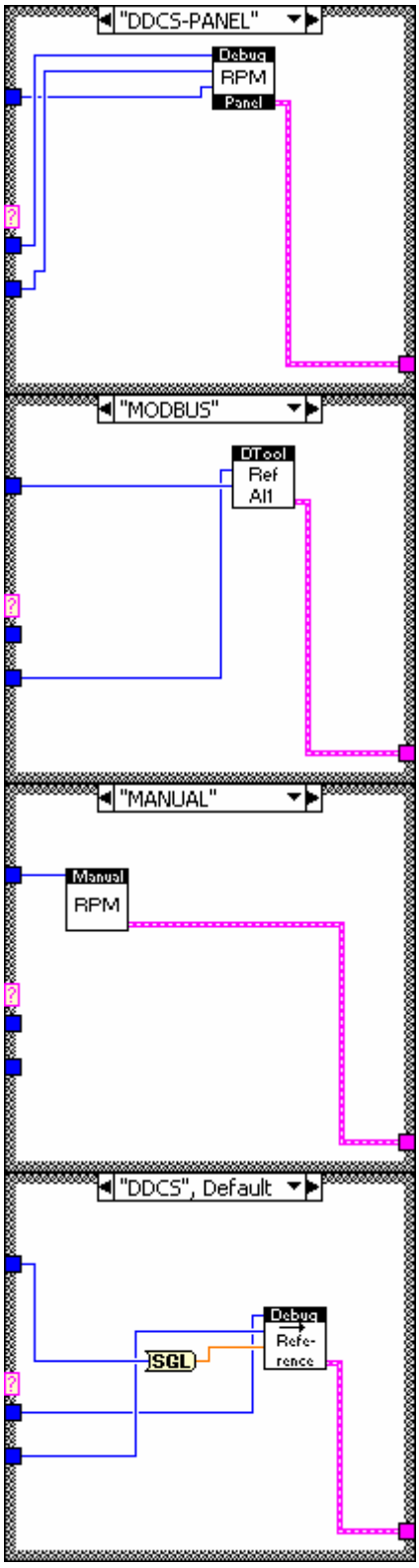
INU-Reference-RPM.vi
Sends speed reference to INU specified by INU Resource Name.

Connector Pane



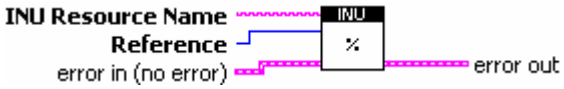
Block Diagram



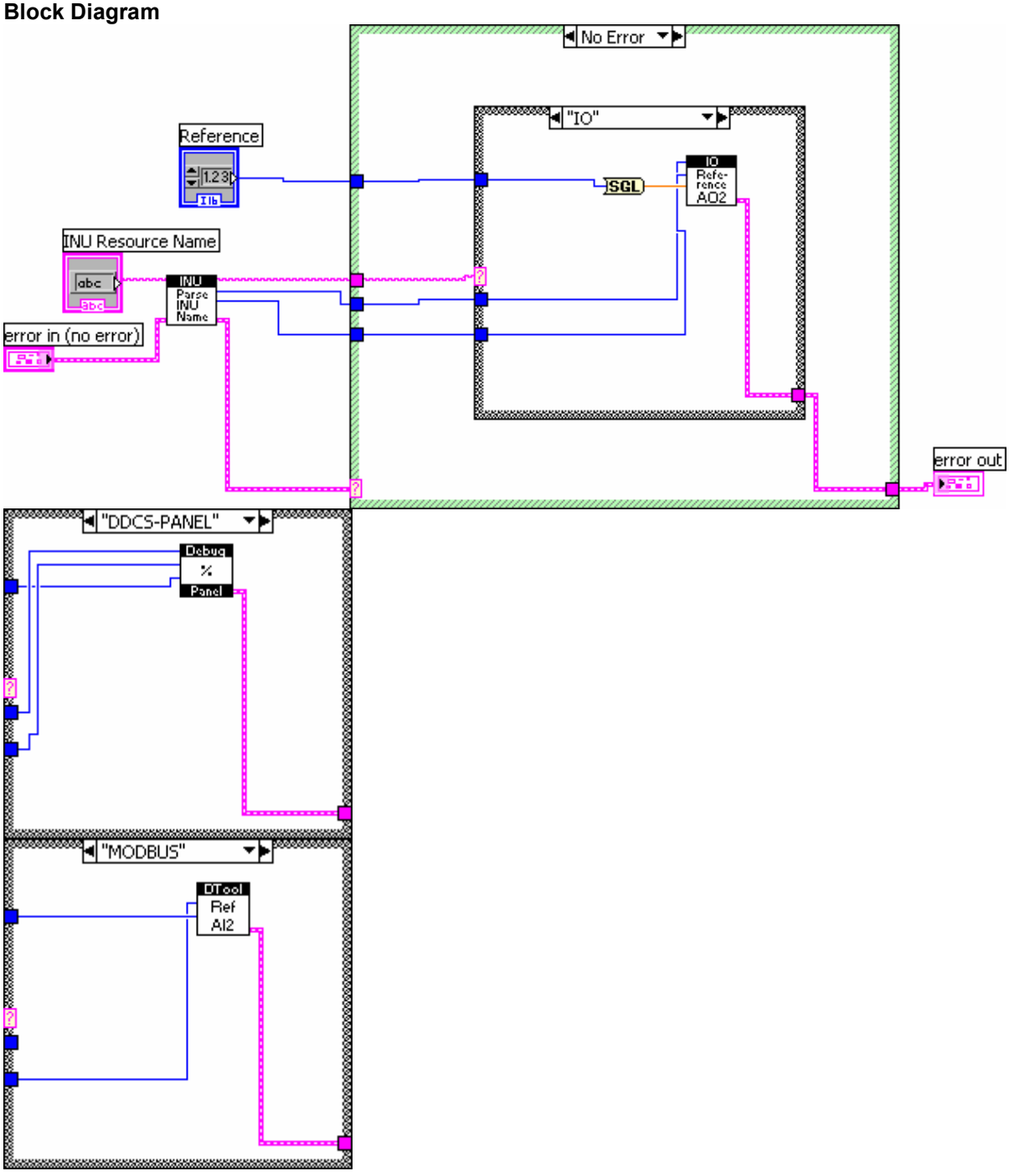

"DDCS-PANEL"
Debug
RPM
Panel
"MODBUS"
DTool
Ref
AI1
"MANUAL"
Manual
RPM
"DDCS", Default
Debug
Refe-
rence
SGL

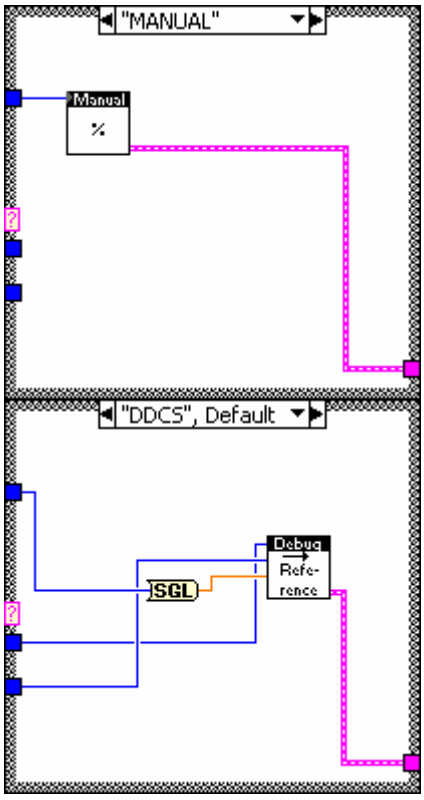
INU-Reference-Torque.vi
Sends torque reference to INU specified by INU Resource Name.

Connector Pane



Block Diagram




"MANUAL"
Manual
%
"DDCS", Default
Debug
Refe-
rence
SGL