Metropolia

**TEKNIIKKA JA LIIKENNE**

**Sähkötekniikka**

**Elektroniikka**

**INSINÖÖRITYÖ**

# TAAJUUSMUUTTAJAN LAAJENNUSMODUULITESTERIT

**Työn tekijä: Jarmo Strand**
**Työn valvoja: Marko Uusitalo**
**Työn ohjaaja: Ari Korpela**

**Työ hyväksytty: ___. ___. 2010**

**Marko Uusitalo**
**lehtori**

**ALKULAUSE**

Tämä insinöörityö tehtiin ABB Oy Drivesin yksikölle Helsingissä helmi - huhtikuun aikana 2010. Haluan kiittää projektissa mukana olleita ABB Oy Drivesin kollegoja, etenkin työni ohjaajaa insinööri Ari Korpelaa. Metropolia Ammattikorkeakoulusta haluan kiittää työni valvojaa lehtori Marko Uusitaloa.

Helsingissä 31.5.2010

Jarmo Strand

**TIIVISTELMÄ**

**Työn tekijä:** Jarmo Strand

**Työn nimi:** Taajuusmuuttajan laajennusmoduulitesterit

| **Päivämäärä:** 31.5.2010 | **Sivumäärä:** 48 s. + 27 liitettä |
|---|---|
| **Koulutusohjelma:** | **Suuntautumisvaihtoehto:** |
| Sähkötekniikka | Elektroniikka |

**Työn ohjaaja:** lehtori Marko Uusitalo, Metropolia Ammattikorkeakoulu

**Työn ohjaaja:** insinööri Ari Korpela, ABB Oy Drives

Insinöörityössä suunniteltiin sekä toteutettiin taajuusmuuttajan digitaali ja analogia I/O -laajennusmoduuleille uudenlaiset testausjärjestelmät. Jokaiselle moduulille (FIO-01, FIO-11, FIO-21, FIO-31) tehtiin oma testausohjelma. Testausohjelmat on nimetty erikseen jokaisen moduulin mukaan.

Uudet testausohjelmat laajennusmoduuleille toteutettiin visuaaliseen ohjelmointiympäristöön sijoittuvalla LabVIEW-ohjelmistolla. Ohjelmassa käytetään ohjelmointiin visuaalista G-kieltä.

Vanhassa testaustavassa pystytään kommunikoimaan laajennusmoduuleiden kanssa syöttämällä testi-ohjelmaan heksadesimaaliarvoja. Laajennusmoduulista saatu vastaus ilmoitetaan ohjelmassa myös heksadesimaaliarvoina. Lähetys- ja vastausarvot katsotaan erillisestä spesifikaatiosta. Systeemi on vielä keskeneräinen, aikaa vievä sekä vaatii paljon keskittymistä ja paneutumista.

Uusista testausohjelmista pyrittiin tekemään luotettavia ja käytännöllisiä, sekä niiden odotetaan nopeuttavan laajennusmoduuleiden testaamista tulevaisuudessa. Myös käyttöliittymää on pohdittu käyttäjän näkökulmasta.

Testausvaiheessa on otettava huomioon myös laajennusmoduuleille tulevat jännite- ja virtatasot, testausjärjestelmään liittyvät mekaaniset osat/johtimet, JEXT-01:lle tuleva syöttöjännite +24 V (DC) sekä tarvittavat syöttö- ja mittalaitteet.

**Avainsanat:** VPL, LabVIEW, FIO-01, FIO-11, FIO-21, FIO-31, käyttöliittymä

Metropolia

**ABSTRACT**

**Name:** Jarmo Strand

**Title:** Frequency converter extension module testers

**Date:** 31.5.2010 **Number of pages:** 48 p. + 27 attachments

**Department:** Electrical engineering **Study Programme:** Electronics

**Instructor:** **Marko Uusitalo, Senior Lecturer, Helsinki Metropolia University of Applied sciences**

**Supervisor:** **Ari Korpela, Engineer, ABB PLC Drives**

In this work a new test system for testing the digital and analog I/O extension modules of a frequency converter was designed and executed. An individual tester was made for each module. The testers were given the same names as the modules have.

The new extension module testers (FIO-01, FIO-11, FIO-21, FIO-31) were executed within the LabVIEW visual programming environment, which uses G-language.

The old test system allows the user to communicate with the extension modules by the user entering hexadecimal values in the test software. The program reports the result in hexadecimal values, as well. The send and receive values are found in a separate specification in an Excel spreadsheet. The system is time-consuming and it requires a great deal of concentration.

The new test environment should be reliable and practical and it is expected to minimize the extension module testing time in the future. Also the user interface has been considered from the point of view of the user.

In the test stage, the user has to pay attention to incoming voltage and current levels and mechanical parts/cables of the tester as well as JEXT-01 incoming +24 V (DC) and the supply and measuring equipment needed.

**Keywords:** VPL, LabVIEW, FIO-01, FIO-11, FIO-21, FIO-31, User Interface

**SISÄLLYS**

**ALKULAUSE**

**TIIVISTELMÄ**

**ABSTRACT**

**SISÄLLYS**

**LYHENTEITÄ JA MÄÄRITELMIÄ**

**LYHENTEITÄ JA MÄÄRITELMIÄ**

| | |
|---|---|
| ASIC | Application Specific Integrated Circuit; räätälöity mikropiiri tiettyyn käyttötarkoitukseen |
| Block Diagram | Lohkokaavio; laitteiston/ohjelmiston toiminnallinen kuvaus |
| .dll | Dynamic-Link Library; koostuu pienistä ohjelmista, joita voidaan kutsua suoritettavalla ohjelma exe-tiedostolla |
| FPGA | Field-programmable Gate Array; integroitu kenttäohjelmoitava mikropiiri |
| DUT | Device Under Test; testattavana oleva laite |
| GUI builder | Graphical User Interface builder; ohjelmiston kehitys työkalu, jolla pystytään muodostamaan graafinen käyttäjärajapinta |
| .h file | Header File; otsikkotiedosto, pääohjelman osaohjelma, joka koostuu erilaisista toiminnoista |
| LabVIEW | Laboratory Virtual Instrument Engineering Workbench; National Instrumentsin graafinen ohjelmointiympäristö |
| SubVI | Sub Virtual Instrument; LabVIEW-ohjelman aliohjelma |
| USB | Universal Serial Bus; yleismaailmallinen sarjaväyläarkkitehtuuri |
| VI | Virtual Instrument; virtuaalinen LabVIEW -ohjelma |
| VPL | Visual Programming Laguage; visuaalinen ohjelmointikieli |

# 1 JOHDANTO

ABB Oy, Drivesin yksikössä on aiemmin tutkittu melko vähän vikaantuneita digitaalisia ja analogisia I/O-laajennusmoduuleita (*digital/analog IO -extension module*) johtuen niiden melko pienestä menekistä. Nykyään laajennusmoduuleita on alettu myydä yhä enemmän. Odotettavissa on, että laajennusmoduuleiden myynti kasvaa entisestään uuden taajuusmuuttajaperheen markkinoille tulon myötä. Tässä insinöörityössä tehtiin testausjärjestelmä, jonka avulla pystyttään paikantamaan laajennusmoduuleiden vikaantunut tulo- tai lähtöliitäntä nopeasti ja luotettavasti.

Työn alussa esitellään laajennusmoduulit, joille testausjärjestelmät on suunniteltu. Työssä kerrotaan testausjärjestelmistä, niiden mekaanisista osista ja kommunikoinnista. Myös vanhaa vielä hieman keskeneräistä testausohjelmaa esitellään. Uudet testausohjelmat on toteutettu visuaalisella LabVIEW-ohjelmistolla. Tämä monipuolinen ohjelmointiympäristö käyttää ohjelmissaan G-kieltä. Erilaisia toimintoja (*functions*) omaavia komponentteja (*components*) sopivasti yhdistelemällä pystytään luomaan suuriakin ohjelmakokonaisuuksia.

Työssä kerrotaan LabVIEW-ohjelmistosta, sen komponenteista ja visuaalisesta ohjelmoinnista. Testausohjelmien toimintaa selvitetään lohkokaavioiden ja etupaneeleiden avulla. Testausjärjestelmien käyttöohje esitellään. Lopussa on pohdintaa testausjärjestelmistä, niiden taloudellisuudesta ja jatkokehitysmahdollisuuksista. Myös testausohjelmissa käytetyt raja-arvot, tarkkuudet ja kaavat esitellään.

# 2 TESTAUSJÄRJESTELMÄN AINEISTO JA MENETELMÄT

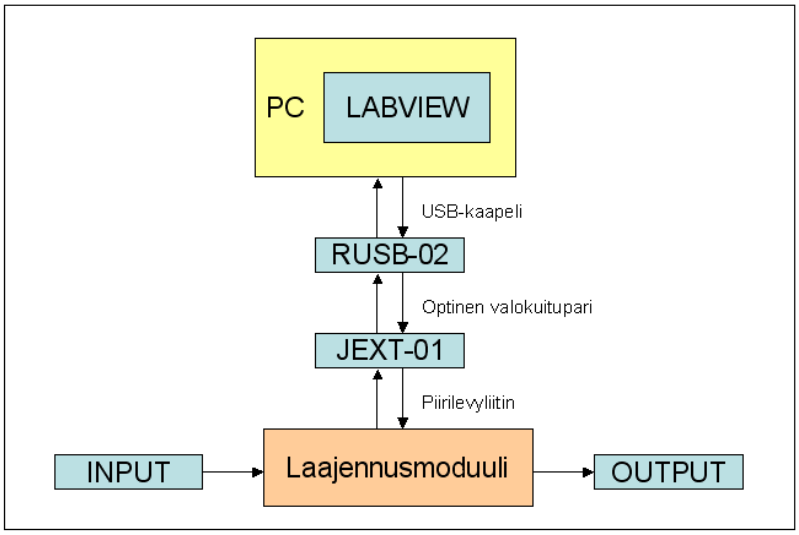
## 2.1 Testausjärjestelmän käyttötarkoitus

Testausjärjestelmällä pystytään selvittämään laajennusmoduuleiden (FIO-01, FIO-11, FIO-21, FIO-31) viallinen tulo/lähtö-liitäntä. Jos moduuleiden vika halutaan tarkasti paikantaa, joudutaan moduuleita tutkimaan myös komponenttitasolla, johon on olemassa omat menettelytavat.

Testausjärjestelmän testausohjelmistot on toteutettu visuaalisella LabVIEW -ohjelmistolla.

Moduuleiden tulot pystytään testaamaan syöttämällä haluttuun tuloon (*input*) virtaa tai jännitettä sopivalla teholähteellä (ks. kuva 1). Testausohjelmasta pystytään havaitsemaan, onko tulo kunnossa vai viallinen. Insinöörityön laajennusmoduuleiden tulojen testauksissa on käytetty teholähteenä signaalikalibraattoria. Moduuleiden lähdöt pystytään testaamaan asettamalla testausohjelmasta haluttuun lähtöön (*output*) tietty arvo. Arvo on mitattava ulkoisella mittalaitteella ulostulosta. Työssä laajennusmoduuleiden lähtöjen testauksissa on käytetty mittalaitteena signaalikalibraattoria tai digitaalista yleismittaria.

Relelähtöjen tilaa pystytään vaihtamaan testausohjelmasta. Releen koskettimen toiminnallisuus pystytään havaitsemaan relelähdöstä. Työssä releen koskettimien toiminnallisuus on mitattu digitaalisen yleismittarin ohmi -alueella. Testausjärjestelmän mekaaniset kytkennät (ks. 2.6).




*Kuva 1. Testausjärjestelmän kommunikointi*

## 2.2 Digitaali I/O (*input/output*) -laajennusmoduulit

Digitaali I/O -laajennusmoduulit ovat moduuleita, jotka pystytään kytkemään 20-napaisella piirilevyliittimellä taajuusmuuttajaan. Niitä käytetään, kun

halutaan lisätä taajuusmuuttajan digitaalisten I/O-liitäntöjen määrää. FIO-01 ja FIO-31 ovat digitaalisia I/O-laajennusmoduuleja.

### *2.2.1 FIO-01 Digitaali I/O -laajennusmoduuli (Digital IO Extension)*

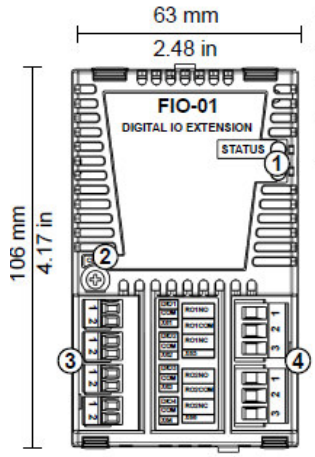
FIO-01 sisältää neljä digitaalista tulo/lähtö-toimintoa ja kaksi relelähtöä (*relay output*). Kuvassa 2 on esitetty FIO-01:n etupaneeli.

Digitaalinen tulo/lähtö -toiminto (*DIO1, DIO2, DIO3, DIO4*)

- 24 V:n loogiset tuloarvot: ”0” < 5 V, ”1” > 15 V
- lähtö, lähdevirta 50 mA ja nieluvirta 50 mA

Releulostulo (RO1, RO2)

- 250 V AC / 30 DC, 2 A
- NO = rele lepotilassa auki
- COM = yhteinen
- NC = rele lepotilassa kiinni [1.]




| | |
|---|---|
| 1 | Diagnostic LED |
| 2 | Fixing screw |
| 3 | Digital I/O |
| 4 | Electromechanical relays |

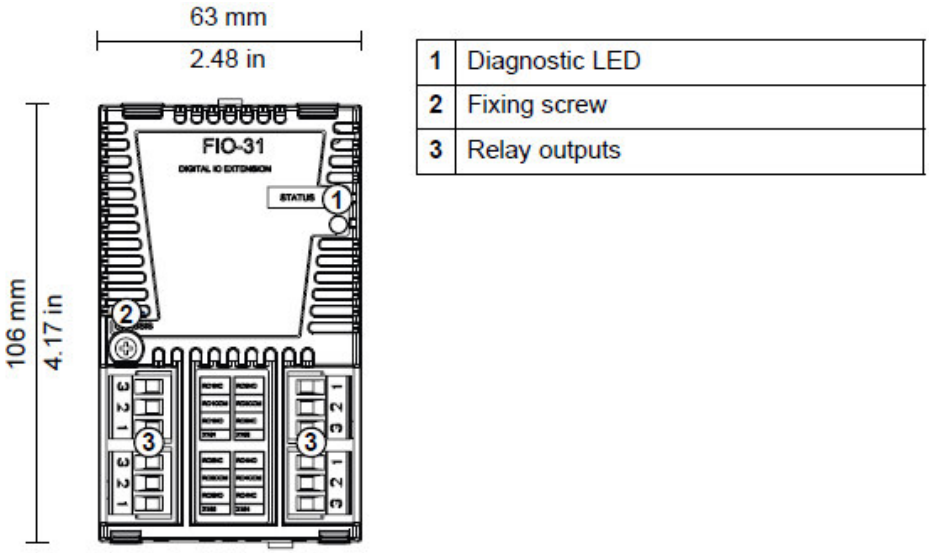
*Kuva 2. FIO-01 Digitaali I/O -laajennusmoduuli [1]*

### *2.2.2 FIO-31 Digitaali I/O -laajennusmoduuli (Digital IO Extension)*

FIO-31 sisältää neljä relelähtöä. Relelähdöt sijoittuvat FIO-31:lle kuvan 3 mukaisesti.

Releulostulo (RO1, RO2, RO3, RO4)

- 250 V AC / 30 DC, 2 A
- NO = rele lepotilassa auki
- COM = yhteinen
- NC = rele lepotilassa kiinni [2.]




*Kuva 3. FIO-31 Digitaali I/O -laajennusmoduuli [2]*

## 2.3 Analogia I/O -laajennusmoduulit

Analogia I/O -laajennusmoduulit ovat moduuleita, jotka pystytään kytkemään 20-napaisella piirilevyliittimellä taajuusmuuttajaan. Niitä käytetään, kun halutaan lisätä taajuusmuuttajan analogisten I/O-liitäntöjen määrää. FIO-11 ja FIO-21 ovat analogisia I/O -laajennusmoduuleja.

### *2.3.1 FIO-11 Analogia I/O -laajennusmoduuli (Analog IO Extension)*

FIO-11 sisältää kolme analogiatuloa, analogialähdön ja kaksi digitaalista tulo/lähtö -toimintoa. Kuvassa 4 on esitetty FIO-11:n etupaneeli.

Analogiatulo (AI1, AI2, AI3)

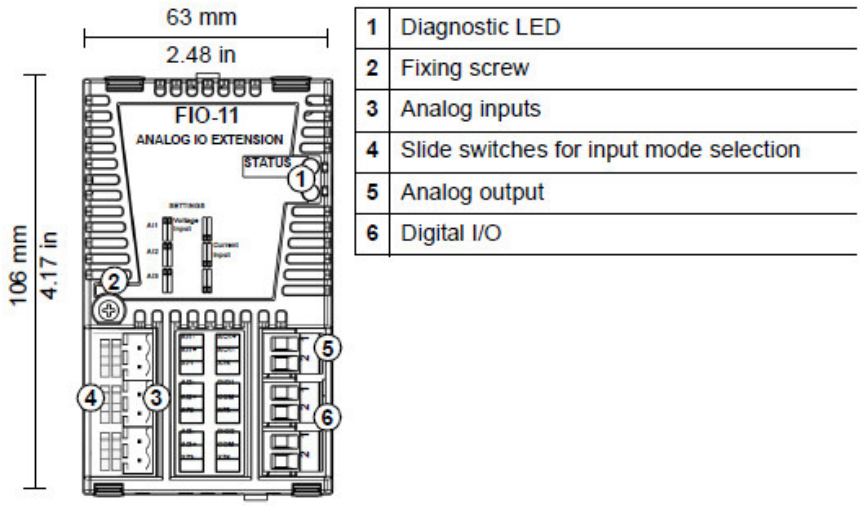
- virtatulo -20…20 mA
- jännitetulo -10…10 V

Analogialähtö (AO1)

- virtalähtö 0…20 mA

Digitaalinen tulo/lähtö-toiminto (DIO1, DIO2)

- 24 V:n loogiset tuloarvot: ”0” < 5 V, ”1” > 15 V
- lähtö, lähdevirta 50 mA ja nieluvirta 50 mA [3.]




| | |
|---|---|
| 1 | Diagnostic LED |
| 2 | Fixing screw |
| 3 | Analog inputs |
| 4 | Slide switches for input mode selection |
| 5 | Analog output |
| 6 | Digital I/O |

*Kuva 4. FIO-11 Analogia I/O -laajennusmoduuli [3]*

### *2.3.2 FIO-21 Analogia I/O -laajennusmoduuli (Analog IO Extension)*

FIO-21 sisältää analogiatulon, analogialähdön, digitaalitulon ja kaksi releläh-töä. Tulot ja lähdöt ovat sijoittuneet FIO-21:lle kuvan 5 osoittamalla tavalla.

Analogiatulo (AI1)

- virtatulo 0…20 mA

Analogialähtö (AO1)

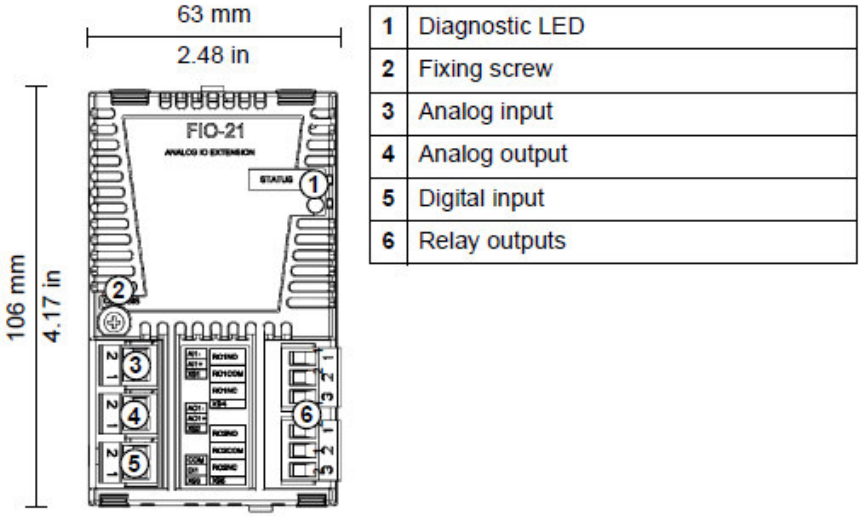
- virtalähtö 0…20 mA

Digitaalitulo (DI1)

- 24 V:n loogiset tuloarvot: ”0” < 5 V, ”1” > 15 V

Relelähtö (RO1, RO2)

- 250 V AC / 30 DC, 2 A
- NO = rele lepotilassa auki
- COM = yhteinen
- NC = rele lepotilassa kiinni [4.]




*Kuva 5. FIO-21 Analogia I/O -laajennusmoduuli [4]*

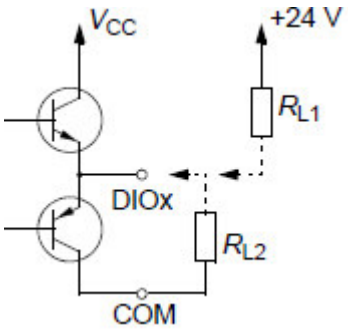
## 2.4 Laajennusmoduuleissa käytettyjä lähtötyyppejä

Digitaalilähdöissä on käytetty kolmea eri lähtötyyppiä: 2-tila lähtö (*push-pull*), avokollektorilähtö (*open collector*) ja avoemitterilähtö (*open emitter*). Analogialähdöissä on käytetty avoemitterilähtöä.

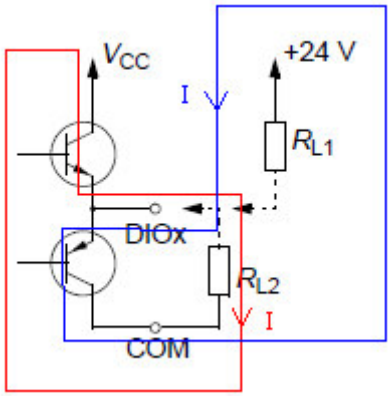
### *2.4.1 2-tila lähtö (push-pull)*

Ylemmän transistorin kautta pystytään kuormaan $R_{L2}$ työntämään (*source*) virtaa (ks. kuvat 6 ja 7). Samanaikaisesti alempi transistori on ei-johtavassa

tilassa. Kun ylempi transistori asetetaan ei-johtavaan tilaan, $R_{L1}$:n kautta pystytään imemään (*sink*) virtaa alemmalle transistorille.




*Kuva 6. 2-tila lähtö [3]*

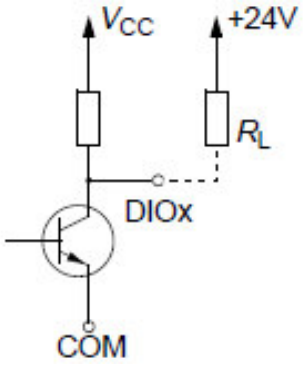



*Kuva 7. Virran kulkusuunnat*

Toisen transistorin ollessa johtavassa tilassa on toinen transistori ei -johtavassa tilassa. Virta kulkee kerrallaan vain toisen kuorman kautta.

### *2.4.2 Avokollektorilähtö (open collector)*

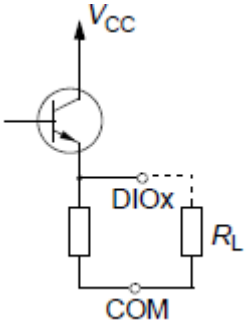
Avokollektorilähtö pystyy olemaan kahdessa eri lähtötilassa korkeaimpedanssisessa (ns. kelluva-tila) ja 0 -tilassa. Jos loogisia tiloja 0 ja 1 halutaan käyttää, niin lähtöön on kytkettävä ylösvetovastus. Avokollektorilähtöä käytetään usein logiikkapiiri-, väyläsovelluksissa tai pienen kuorman ohjaamisessa.



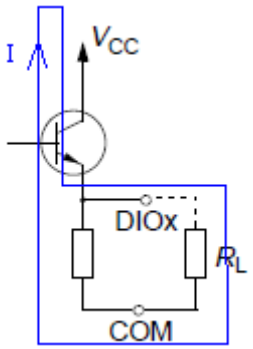

*Kuva 8. Avokollektorilähtö [3]*

### *2.4.3 Avoemitterilähtö (open emitter)*

Virta kulkee transistorin kollektorin kautta emitterille, josta se jatkaa matkaa kuormalle ja sitä kautta paluuvirtajohdinta (COM) pitkin jännitelähteelle. Kuvissa osa virrasta kulkee myös kuorman rinnalla olevan vastuksen kautta:




*Kuva 9. Avoemitterilähtö [3]*

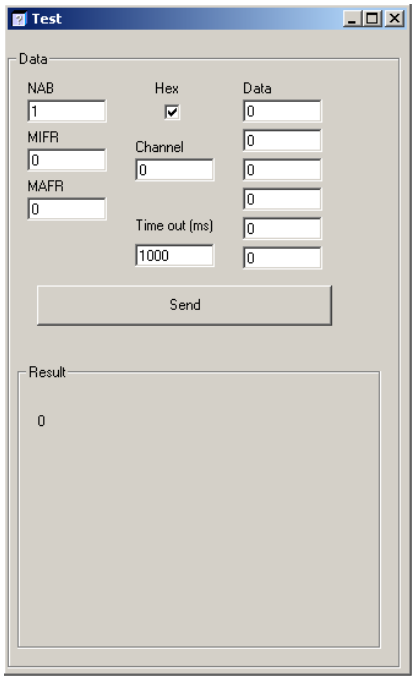



*Kuva 10. Virran kulkusuunta*

## 2.5 Laajennusmoduuleiden aiemmat testausmenetelmät

Laajennusmoduuleita on aikaisemmin testattu test.exe -ohjelman avulla. Ohjelma on aikoinaan suunniteltu ABB'een tuotekehitystiimin avuksi DDCS-kommunikointijärjestelmiin. Ohjelmaa on käytetty erilaisissa sovelluksissa. Sillä pystytään syöttämään dataa heksadesimaaliarvoina suoraan laajennusmoduulin FPGA:aan VIE4 logiikan rekistereihin tai ASIC-mikropiirille (FIO-01). Ohjelma lukee automaattisesti vastaanotetun datan VIE4:ltä tai ASIC:lta ja tulostaa sen näytölle kohtaan *Result* (ks. kuva 11). Saatuja heksadesimaalisia arvoja on verrattava laajennusmoduulille tehtyyn spesifikaatioon [5, 6, 7, 8].

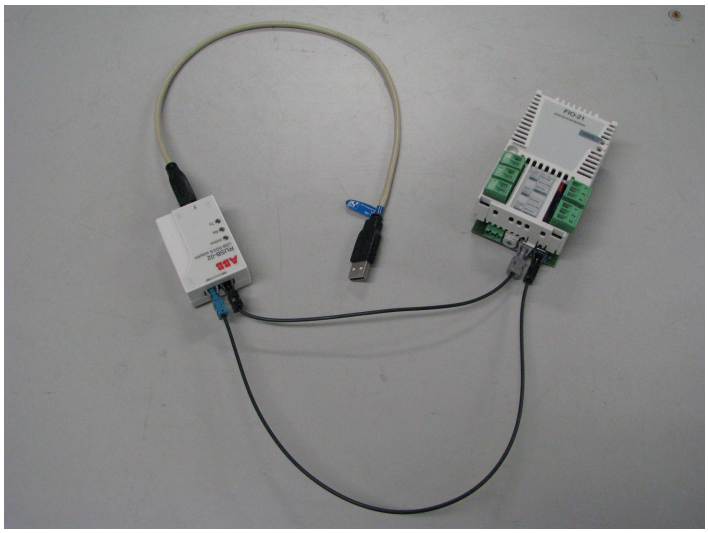
Laajennusmoduulin analogia- sekä digitaalituloihin on syötettävä sopivan suuruista jännitettä tai virtaa. Lähtöjen virta- tai jännitetaso on mitattava ulkoisella mittalaitteella. Käytettyjä mitta- ja syöttölaitteita ovat muun muassa digitaalinen yleismittari, oskilloskooppi ja signaalikalibraattori.




*Kuva 11. Test-ohjelman etupaneeli*

## 2.6 Testausjärjestelmän mekaaniset osat

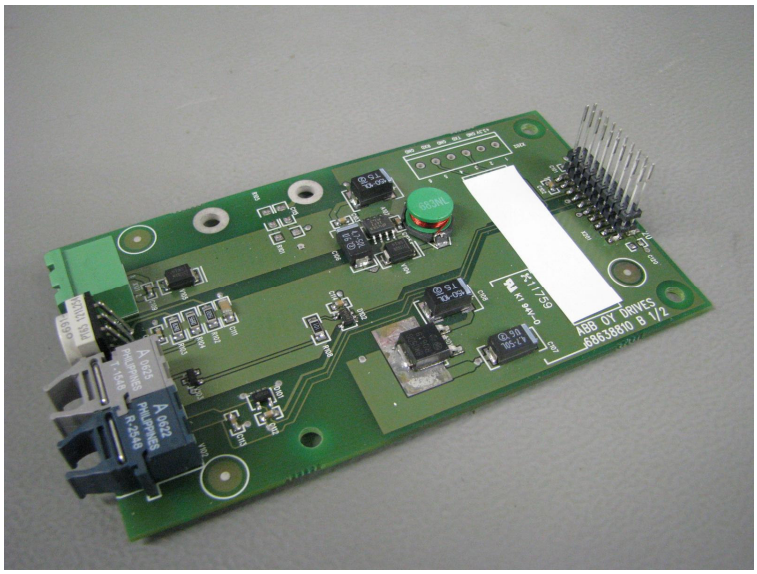
Testausjärjestelmässä tarvitaan testattavien laajennusmoduuleiden lisäksi myös muita mekaanisia osia (ks. kuva 12). JEXT-01, RUSB-02 sekä niiden ja tietokoneen väliset datakaapelit. On huomioitava myös, että JEXT-01 tarvitsee toimiakseen 24 V:n syöttöjännitteen.



*Kuva 12. Testausjärjestelmän mekaaniset osat*

### 2.6.1 JEXT-01-adapteri

JEXT-01 muuntaa optisen datan laajennuskortille sopivaksi galvaaniseksi dataksi ja päinvastoin. Kuvassa 13 on esitelty melko vähän komponentteja sisältävä JEXT-01-adapteri. Kuvan vasenmassa ylälaidassa olevaan vihreään riviliittimeen tuodaan JEXT-01:n käyttöjännite +24 V (*DC*). Kuvan etulaidassa näkyy optiset lähetin- sekä vastaanotinkomponentit.




*Kuva 13. JEXT-01-adapteri*

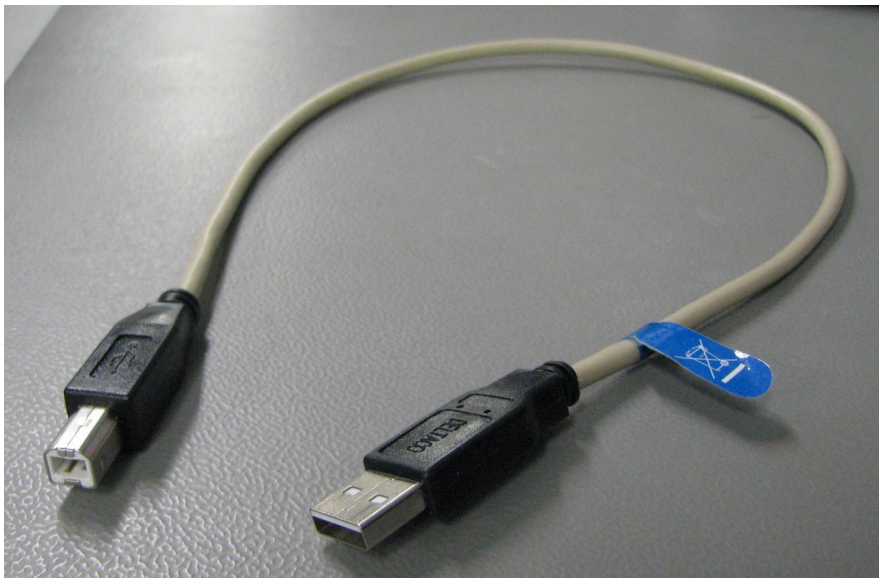
### 2.6.2 RUSB-02 USB/DDCS -adapteri

RUSB-02 suorittaa USB/DDCS-muunnoksen, tällöin tiedon kulku muuttuu galvaanisesta optiseen muotoon sekä päinvastoin, optisien lähetin ja vastaanotin komponenttien välityksellä:


*Kuva 14. RUSB-02 USB/DDCS -adapteri*

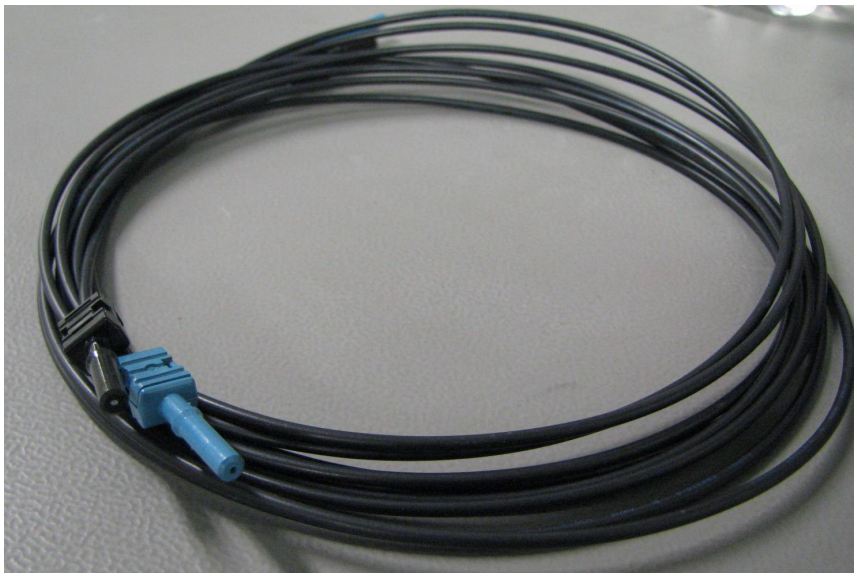
### *2.6.3 USB- ja valokaapeli*

Tietokoneen ja RUSB-02:n välillä käytetään galvaanista USB-kaapelia. Tietokoneen päädyssä on A-tyyppinen USB-liitin ja RUSB-02:n päädyssä on B-tyyppinen USB-liitin.



*Kuva 15. Galvaaninen USB-kaapeli*

RUSB-02:n ja JEXT-01:n välisessä kommunikoinnissa käytetään optista valokuitua, jossa signaali kulkee kuidun sisällä valon muodossa. Lähetettävä ja vastaanotettava data kulkevat eri kuiduissa muodostaen kuituparin:



*Kuva 16. Optinen valokuitupari*

## 2.7 Ohjelman ja optiokortin välinen kommunikointi

RUSB-02 kiinnitetään tietokoneeseen USB-kaapelilla (vrt. kuva 1). RUSB-02 muuntaa datan USB-muodosta DDCS-muotoon ja päinvastoin. Eli signaalin siirtotapa muuttuu galvaanisesta optiseen tai optisesta galvaaniseen siirtosuunnan perusteella. RUSB-02:n ja JEXT-01:n välillä käytetään optista valokuituparia. JEXT-01-adapteri muuttaa optisen tiedon galvaaniseksi, laajennusmoduulille sopivaksi sarjamuotoiseksi dataksi ja päinvastoin laajennusmoduulilta tulevan datan optiseen muotoon RUSB-02:lle.

## 2.8 Visuaalinen ohjelmointikieli (*visual programming language*)

Visuaalinen ohjelmointikieli (VPL) koostuu kaavioista (*charts*), kaavoista (*forms*), ikoneista (*icons*) ja symboleista (*symbols*). VPL pystytään jakamaan kolmeen pääryhmään: vuokaavio esitysmuotoon, kieliin, joissa ei käytetä vuokaavioita ja taulukkokieliin. Esitystavan visuaaliset komponentit on pyritty toteuttamaan siten, että ne nostattaisivat käyttäjän käsityskykyä eivätkä olisi vain pino suoritettavia käskyjä.

Ohjelmistojen erilaiset komponentit sisältävät funktioita. Funktioita sopivasti yhdistelemällä pystytään rakentamaan hyvinkin laajoja ohjelmakokonaisuuksia. Tietty ohjelmallinen tavoite pystytään saavuttamaan usealla eri tavalla, erinäisiä muuttujia hyväksi käyttäen. Etenemistapa on vapaa, jolloin ohjelmoija pystyy itse päättämään itselleen sopivimman etenemistavan. [9.]

Puhdas visuaalinen kieli koostuu systeemistä, jossa täysin visuaalisesti tehty ohjelma pystytään jäsentämään, kääntämään ja suorittamaan samassa kehitysympäristössä. [10.]

Ohjelmat, kuten Visual Basic ja Visual C++, eivät ole visuaalisia ohjelmointikieliä. Ne ovat tekstuaalipohjaisia ohjelmointikieliä, jotka käyttävät graafisia kääntäjiä helpottaakseen ohjelmointirajapintoja. Käyttäjärajapinta on visuaalisesti toteutettu, mutta varsinainen ohjelmakoodi on tekstuaalisessa muodossa. Kuvaus visuaalinen ohjelmointi on hieman harhaanjohtava ilmaus,

siksi Fred Lakin onkin ehdottanut sille korvaavaa nimeä *executable graphics*. [11.]

## 2.9 Visuaalinen ohjelmointiympäristö LabVIEW

LabVIEW on tehokas ja joustava National Instrumentsin tekemä visuaalinen kehitysympäristö. LabVIEW'tä on käytetty paljon muun muassa testaus-, mittaus-, teollisuusautomaatio- ja tietojenanalysointi ympäristöissä.

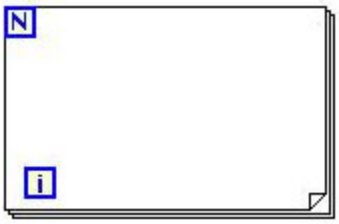
LabVIEW perustuu visuaaliseen ohjelmointikieleen, jota kutsutaan myös G-ohjelmointikieleksi. Siinä graafiset symbolit määrittävät suoritettavat toimenpiteet. Ohjelmoinnissa käytetään monenlaisia kaavioita ja kuvaajia, jotka sisältävät jonkin toiminnon. Graafisissa kuvaajissa käytetään tiedemiehille ja insinööreille tuttua terminologiaa. LabVIEW sisältää laajat kirjastot virtuaalisinstrumenteille ja funktioille.

Tekstipohjaisella Mathscript-osiolla pystytään täydentämään visuaalista ohjelmointiympäristöä. Sen kautta ohjelmaan pystytään nopeasti sisällyttämään erilaisia laskutoimituksia, ja se toimii erinomaisena käyttäjärajapintana erilaisille tekstipohjaisille, komentokielisille lauseille. [12.]

## 2.10 Työssä käytettyjä LabVIEW-komponentteja

*For*-toistorakenteessa (*for loop*) n arvo tuodaan (N) terminaaliin, jolloin kaavion sisältö suoritettaan n-kertaa (ks. kuva 17). Toistoterminaali (i) laskee senhetkisten toistojen määrän, joka tulostuu 0…1-n.

*For*-toistorakenteeseen voidaan lisätä siirtorekisteri tai siirtorekistereitä. Toistolaskuri pystyy laskemaan $2^{31}$-1, jonka jälkeen se jää paikoilleen lukemaan $2^{31}$-1. Toistolaskurin kapasiteettia pystytään kasvattamaan siirtorekisterin avulla.

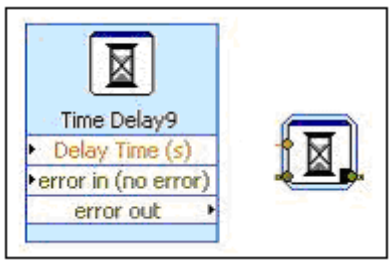



*Kuva 17. For-toistorakenne [13]*

*While*-toistorakenne (*while loop*) toistaa kaavion sisältöä jatkuvasti, kunnes ehdollinen terminaali (tuloterminaali) saa tietyn Boolen-arvon. Tuloterminaaliin pystytään myös liittämään virhekäsittely. Toistolaskuri, vrt. *For* -toistorakenne.


*Kuva 18. While-toistorakenne [13]*

Aikaviivelohkon (*time delay express VI*) viiveen pituus sekunteina pystytään määrittämään lohkon sisältä [*Delay Time (s)*]-kohdasta:




*Kuva 19. Aikaviivelohko [13]*

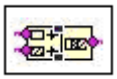
Yhtäsuuri?-toiminto (*equal? function*) vertailee onko sisäänmeno 0, jos näin on, niin ulostulo on 1 (tosi). Muissa tapauksissa ulostulo on 0 (epätosi). Sisäänmeno on numeerinen skalaari-arvo, ja ulostulo on Boolen arvo.

*Kuva 20. Yhtäsuuri?-toiminto [13]*

Boolen arvo muunnetaan 0- ja 1 -arvoiksi [*Boolean to (0, 1) function*]. Toiminto muuntaa Boolen arvon 16-bittiseksi kokonaisluvuksi (0,1).

*Kuva 21. Boolen arvo muunnos 0- ja 1 -arvoiksi [13]*

Toiminto peräkkäistää kaikki sisäänmenevät merkkijonot yhdeksi merkkijonoksi ulostuloon (*concatenate strings function*):

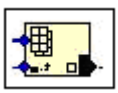


*Kuva 22. Toiminto peräkkäistää merkkijonot [13]*

Toiminto muuntaa merkkijonon etumerkittömien tavujen taulukoksi (*string to byte array function*). Taulukon jokaisella tavulla on ASCII-arvo.

*Kuva 23. Toiminto muuntaa merkkijonon 8-bittiseksi taulukoksi [13]*

Toiminto osoittaa yksittäisen tietopaikan muistista (työssä tavun) (*index array function*):

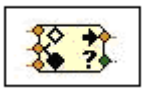


*Kuva 24. Toiminto osoittaa muistipaikan [13]*

Toiminto muuntaa osa-komponenttitavut kokonaisluvuksi (*join numbers function*):

*Kuva 25. Toiminto yhdistää komponentit [13]*

Toiminnossa haluttu **x**-arvo viedään sisäänmenoon, sekä ylä- ja alarajat määritellään. Ulostuloon saadaan Boolen arvo 1 (tosi), jos **x** on tarkasteltavien rajojen sisäpuolella ja 0 (epätosi), jos **x** on tarkasteltavien rajojen ulkopuolella. (*In range and coerce function.*)



*Kuva 26. Toiminto tutkii raja-arvot [13]*

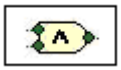
Ohjelmoinnissa käytetyn *XOR*-portin (*exclusive or function*) kuvaaja on esitetty kuvassa 27. Taulukon 1 totuustaulussa on selvitetty, miten *XOR*-portti toimii.

*Kuva 27. XOR-portti [13]*

*Taulukko 1. XOR-portin totuustaulu*

| Input 1 | Input 2 | **Output** |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Ohjelmoinnissa käytetyn *AND*-portin (*AND function*) kuvaaja on esitetty kuvassa 28. Taulukon 2 totuustaulussa on selvitetty, miten *AND*-portti toimii.



*Kuva 28. AND-portti [13]*

*Taulukko 2. AND-portin totuustaulu*

| Input 1 | Input 2 | **Output** |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Ohjelmoinnissa käytetyn *XNOR*-portin (*not exclusive or function*) kuvaaja on esitetty kuvassa 29. Taulukon 3 totuustaulussa on selvitetty, miten *XNOR* -portti toimii.

*Kuva 29. XNOR-portti [13]*

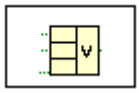
*Taulukko 3. XNOR-portin totuustaulu*

| Input 1 | Input 2 | **Output** |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Kuvassa 30 on esitetty työssä käytetyn *NOT*-portin (*NOT function*) kuvaaja.

*Kuva 30. NOT-portti [13]*

Kuvassa 31 on esitetty työssä käytetyn *OR*-portin (*OR function*) kuvaaja. Komponentin valikosta pystytään valitsemaan onko portti *OR*, *AND*, *XOR*, vai toimiiko se kertojana tai summaimena. Myös toiminnon sisäänmenojen lukumäärä pystytään määrittämään.



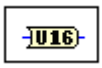
*Kuva 31. OR-portti [13]*

Taulukon 4 totuustaulussa on selvitetty, miten *OR*-portti toimii.

*Taulukko 4. OR-portin totuustaulu*

| Input 1 | Input 2 | **Output** |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Toiminnolla numero muunnetaan etumerkittömäksi 16-bittiseksi kokonaisluvuksi (*to unsigned word integer function):*




*Kuva 32. Komponentti muuntaa numeron kokonaisluvuksi [13]*

Toiminnossa ylempi sisäänmenoarvo jaetaan alemmalla sisäänmenoarvolla. Vastaus saadaan ulostulosta. (*Divide function*):

*Kuva 33. Komponentti suorittaa jakolaskun [13]*

Suurempi tai yhtäsuuri kuin? -toiminto (*greater or equal? function*) tutkii, onko ylempi sisäänmenoarvo suurempi tai yhtäsuuri kuin alempi sisäänmenoarvo. Vastaus saadaan ulostulosta.

*Kuva 34. Suurempi tai yhtäsuuri kuin? -komponentti [13]*

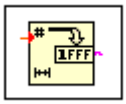
Suurempi kuin 0? -toiminto (*greater than 0? function*) tutkii, onko sisäänmenoarvo suurempi kuin 0. Vastaus saadaan ulostulosta Boolen arvona.

*Kuva 35. Suurempi kuin 0? -komponentti [13]*

Pienempi kuin? -toiminto (*less? function*) tutkii, onko ylempi sisäänmenoarvo pienempi, kuin alempi sisäänmenoarvo. Vastaus saadaan ulostulosta Boolen arvona.

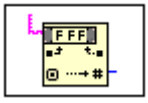
*Kuva 36. Pienempi kuin? -komponentti [13]*

Toiminnossa numeerinen arvo muunnetaan heksadesimaaliseksi merkkijonoksi (*number to hexadecimal string function*):



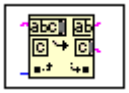

*Kuva 37. Komponentti muuntaa numeerisen arvon heksadesimaaliseksi merkkijonoksi [13]*

Toiminnossa heksadesimaalinen merkkijono muunnetaan numeeriseen muotoon (*hexadecimal string to number function)*.



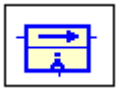

*Kuva 38. Komponentti muuntaa heksadesimaalisen merkkijonon numeeriseen muotoon [13]*

Toiminnossa etsitään jokin merkki merkkijonosta tai jaetaan merkkijono osiin (*search/split string function*):



*Kuva 39. Työssä komponenttia on käytetty jakamaan merkkijono osiin [13]*

Toiminnolla pystytään ohjelmissa toteuttamaan erilaisia takaisinkytkentöjä (*feedback*):



*Kuva 40. Takaisinkytkentä-komponentti [13]*

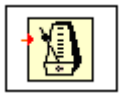
Toiminnon alempi sisäänmenoarvo lisätään ylempään sisäänmenoarvoon. Vastaus saadaan ulostulosta. (*Add function.*)

*Kuva 41. Yhteenlasku-komponentti [13]*

Toiminnon sisäänmenot kerrotaan keskenään, ja vastaus saadaan ulostulosta (*multiply function*):

*Kuva 42. Kertolasku-komponentti [13]*

Toiminto luo aikaviiveen ohjelmaan (ks. kuva 43). Haluttu aikaviive syötetään ms-arvona toiminnon sisälle. (*Wait until next ms multible function.*)



*Kuva 43. Aikaviive-komponentti [13]*

Ohjelmoinnissa käytetyn merkkivalon kuvaaja on esitetty kuvassa 44:


*Kuva 44. Merkkivalo-komponentti [13]*

Ohjelmoinnissa käytetyn kytkimen kuvaaja on esitetty kuvassa 45:


*Kuva 45. Kytkin-komponentti [13]*

Ohjelmoinnissa käytetyn numeerisen syötekentän kuvaaja on esitetty kuvassa 46:


*Kuva 46. Numeerinen syötekenttä -komponentti [13]*

Ohjelmoinnissa käytetyn numeerisen osoitinkentän kuvaaja on esitetty kuvassa 47:


*Kuva 47. Numeerinen osoitinkenttä -komponentti [13]*

Ohjelmoinnissa käytetyn etumerkittömän sanan osoitinkentän kuvaaja on esitetty kuvassa 48:


*Kuva 48. Etumerkittömän sanan osoitinkenttä -komponentti (16-bittiä) [13]*

Ohjelmoinnissa käytetyn etumerkittömän tavun osoitinkentän kuvaaja on esitetty kuvassa 49:


*Kuva 49. Etumerkittömän tavun osoitinkenttä -komponentti (8-bittiä) [13]*

Ohjelmoinnissa käytetyn etumerkillisen kaksoissanan osoitinkentän kuvaaja on esitetty kuvassa 50:


*Kuva 50. Etumerkillisen kaksoissanan osoitinkenttä -komponentti (32-bittiä) [13]*

Ohjelmoinnissa käytetyn merkkijonon osoitinkentän kuvaaja on esitetty kuvassa 51:


*Kuva 51. Merkkijonon osoitinkenttä -komponentti [13]*

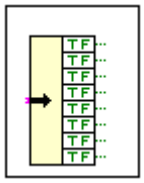
Toiminnossa numero muunnetaan Boolen taulukkomuotoon (*number to Boolean array function*):

*Kuva 52. Komponentti muuntaa numeron Boolen taulukkomuotoon [13]*

Toiminnossa taulukkomuoto muunnetaan klusterimuotoon (*array to cluster function*):

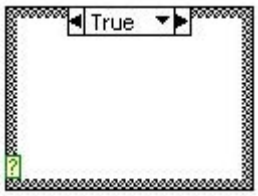
*Kuva 53. Komponentti muuntaa taulukkomuodon klusterimuotoon [13]*

Kuvan 54 komponentti jakaa klusterinipun individuaalisiin elementteihin (*unbudle function*):




*Kuva 54. Komponentti jakaa klusterinipun osiin [13]*

*Case*-valintarakenne (*case structure*) sisältää alikaavion/kaavioita tai tapauksia. Arvo tuodaan valintasisäänmenoon (?), joka päättää, mikä alikaavio suoritetaan (ks. kuva 55). Sisäänmenoarvo voi olla Boolen arvo, merkkijono, kokonaisluku tai luettelotyyppi. (Kuvat 17 - 54.)[13.]




*Kuva 55. Case-valintarakenne [13]*

## 3 DIGITAALI JA ANALOGIA I/O -LAAJENNUSMODUULITESTERIT

Jokaiselle neljälle laajennusmoduulille (FIO-01, FIO-11, FIO-21, FIO-31) on suunniteltu oma testausohjelma LabVIEW'llä. Vaihtoehtoisesti olisi pystytty tekemään yksi testausohjelma, joka olisi sisältänyt kaikki neljä

testausohjelmistoa omissa alikansioissaan. Testausohjelmistot suunniteltiin yksitellen, jolloin lopputuloksena syntyi erilliset testausohjelmat.

Testausohjelmien lohkokaaviot ja etupaneelit eivät eroa suuresti toisistaan. Toistuvat samannäköiset rakenteet yhdenmukaistavat testausohjelmia. Etenkin etupaneeleista on pyritty tekemään yhdenmukaisia, minkä kautta on saavutettu helppolukuisuus.

## 3.1 dll- ja h-tiedoston yhdistäminen

ABB:llä on olemassa dwc_ddcp.dll ja dwc_ddcp.h (ks. liite 1) tiedostot, joiden pohjalta DDCS-kommunikointijärjestelmää lähdettiin rakentamaan. Sähköalalla paljon käytettyjä ohjelmointikieliä ovat mm. C ja C++. Ohjelmoinnissa pääohjelma koostuu usein useista erilaisista osatiedostoista. Pääohjelman osatiedosto, otsikkotiedosto (*header file*) muodostuu usein erinäisistä määrittelyistä, luokista, aliohjelmista, funktioista sekä muista tunnistetiedoista.

Dll-tiedosto sisältää lähdekoodin, joka toteuttaa tietyt funktiot. Työssä käytetty dll-tiedosto toimii DDCS-kommunikointirajapintana.

Työn otsikkotiedoston (dwc_ddcp.h) ohjelma-koodia muutettiin, jotta testausohjelma kykeni kommunikoimaan laajennusmoduulin kanssa. LabVIEW -käyttöä varten koodista poistettiin SendDDCCx:n ja ReceiveDDCCx:n edestä DWC_API teksti, sekä muualta joistakin koodin kohdista FAR. Tämän kautta pystyttiin yhdistämään LabVIEW'ssä h- ja dll-tiedosto.

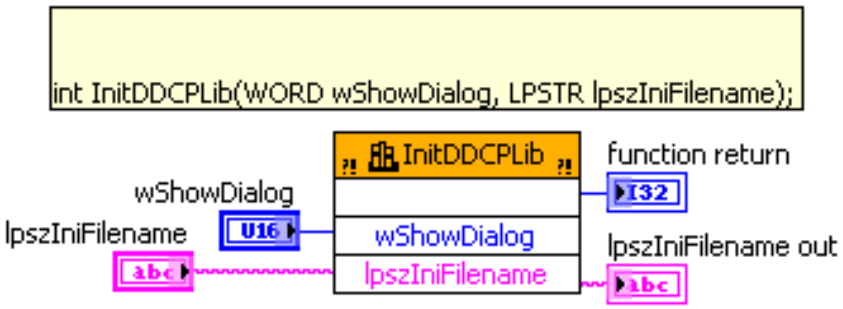
Dll- ja h-tiedosto pystytään yhdistämään LabVIEW'ssä yhteisen kirjastotoiminnon (*Shared Library Function*) avulla. Toiminto löytyy LabVIEW'n valikosta *Tools* → *Import* → *Shared Library*. Toiminto yhdistää tiedostot keskenään ja esittää funktiot, jotka pystytään luomaan ohjelmaan. Luodut funktiot sisältävät lohkokaavion ja etupaneelin.

### 3.2 Alustuslohko (Init DDCP.vi)

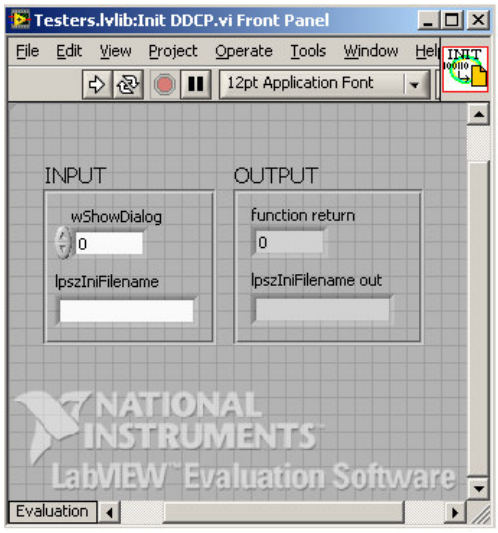
Laajennusmoduulin alustuksessa käytetään Init-lohkoa, joka toimii eräänlaisena herätelohkona. Ajettaessa Init-lohko, *function retur* -ulostuloon ilmestyy numero yksi, jos alustus on onnistuneesti suoritettu (ks. kuva 57). Samanaikaisesti testausohjelman *Initialization*-merkkivalo vaihtuu siniseksi (ks. liite 2). Sinisessä tilassa oleva merkkivalo kertoo, että laajennusmoduuli on asetettu aktiiviseen tilaan, jolloin se pystyy vastaanottamaan ja lähettämään dataa.

*WShowDialog* arvoksi on asetettu nolla. Merkkijonotuloon (*lpszIniFilename*) ei ole asetettu merkkejä, jolloin merkkijonolähtöön ei tulostu mitään. Etupaneelin (*Front Panel*) ylälaidassa oleva kuvake pystytään räätälöimään kulloisellekin LabVIEW-ohjelmalle sopivaksi (ks. kuva 57). Kuvakkeeseen pystytään myös määrittelemään sisäänmenot ja ulostulot, jolloin LabVIEW tunnistaa ohjelman automaattisesti.

Init DDCP.vi, Send DDCC.vi ja Receive DDCC.vi ovat työn aliohjelmia. Aliohjelmia on käytetty pääohjelmissa räätälöidyillä kuvakkeilla.



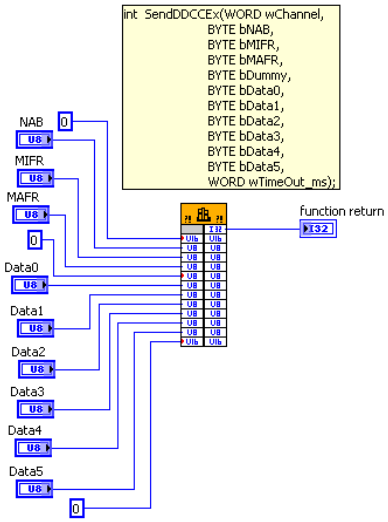

*Kuva 56. Alustuslohkon lohkokaavio*



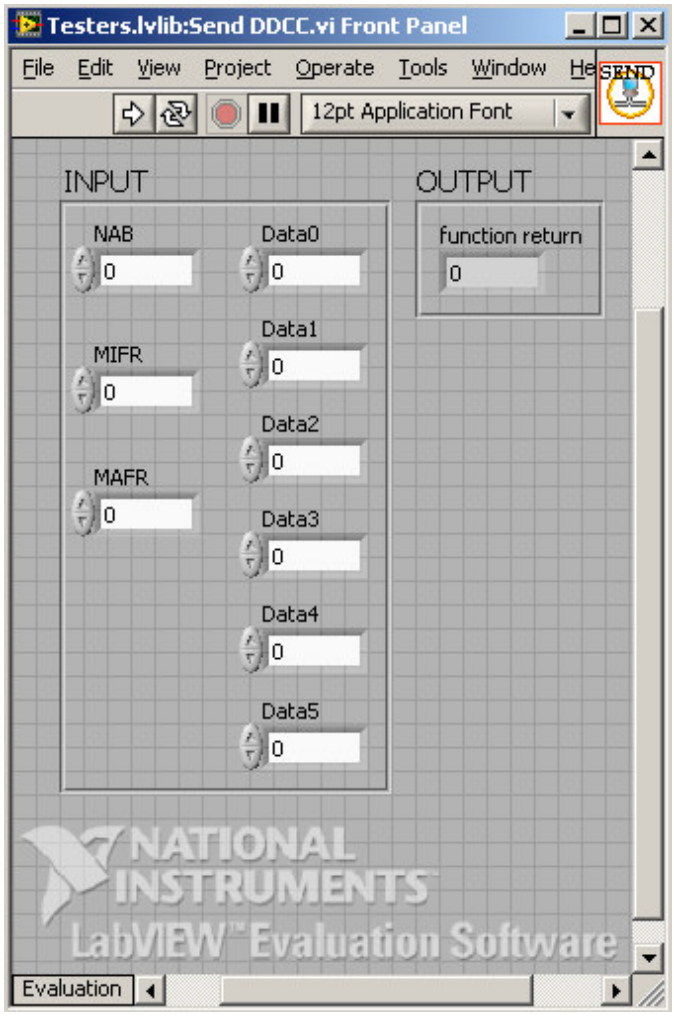

*Kuva 57. Alustuslohkon etupaneeli*

## 3.3 Lähetyslohko (Send DDCC.vi)

Lähetys-aliohjelmalla pystytään lähettämään dataa laajennusmoduuleiden ASIC:lle tai FPGA:n rekistereille (ks. kuva 59). Data lähetetään ohjelmassa kokonaislukuina. ABB:n *Test Steps and Limits Specification* [5, 6, 7, 8] -dokumenteissa on lähetettävä data ilmoitettu heksadesimaaliarvoina. Kuuteen datasisäänmenoon (*Data0* - *Data5*) pystytään asettelemaan haluttu data kokonaisluvuilla 0 - 255. bDummy, wTimeOut_ms ja wChannel arvot on asetettu nolliksi.




*Kuva 58. Lähetyslohkon lohkokaavio*

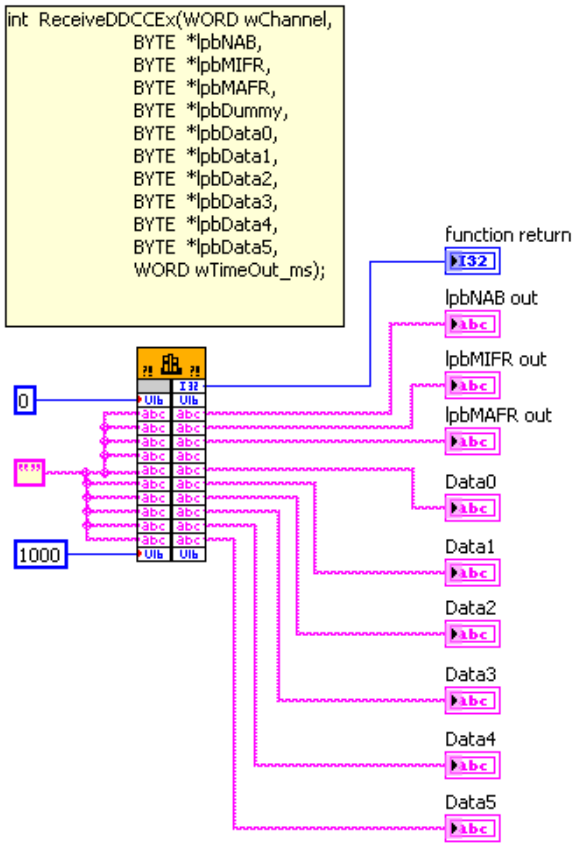



*Kuva 59. Lähetyslohkon etupaneeli*

### 3.4 Vastaanottolohko (Receive DDCC.vi)

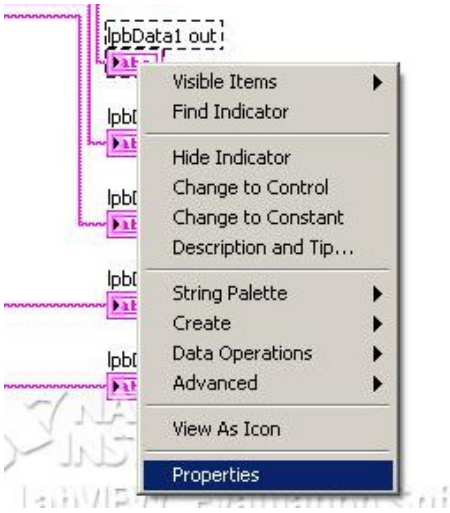
Vastaanotto-aliohjelmalla pystytään lukemaan laajennusmoduuleiden ASIC -piiriltä tai FPGA:n rekistereistä data-arvoja (ks. kuva 63). Vastaanotetut data-arvot näkyvät kohdissa *Data 0* - *Data 5*. Testausohjelmissa vastaanotto-aliohjelmaa käytetään kortin ID-numeron ja logiikkaversion tarkastamisessa sekä analogia- että digitaalituloissa.

Vastaanottolohkon arvot saadaan ohjelmassa ASCII-merkkeinä. ASCII -merkit on myöhemmin muutettu numeeriseen tai heksadesimaaliseen muotoon.

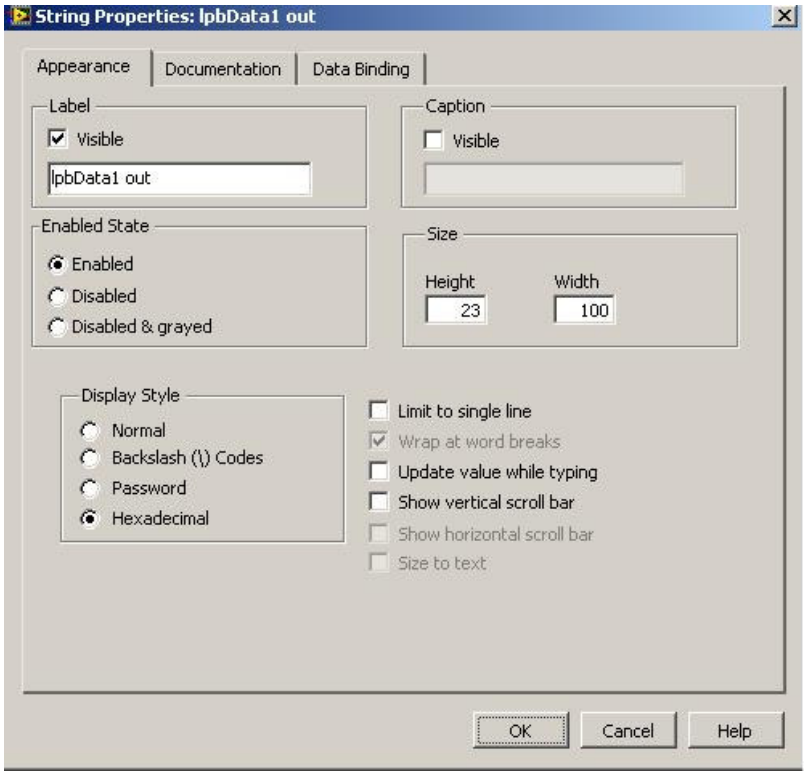



*Kuva 60. Vastaanottolohkon lohkokaavio*

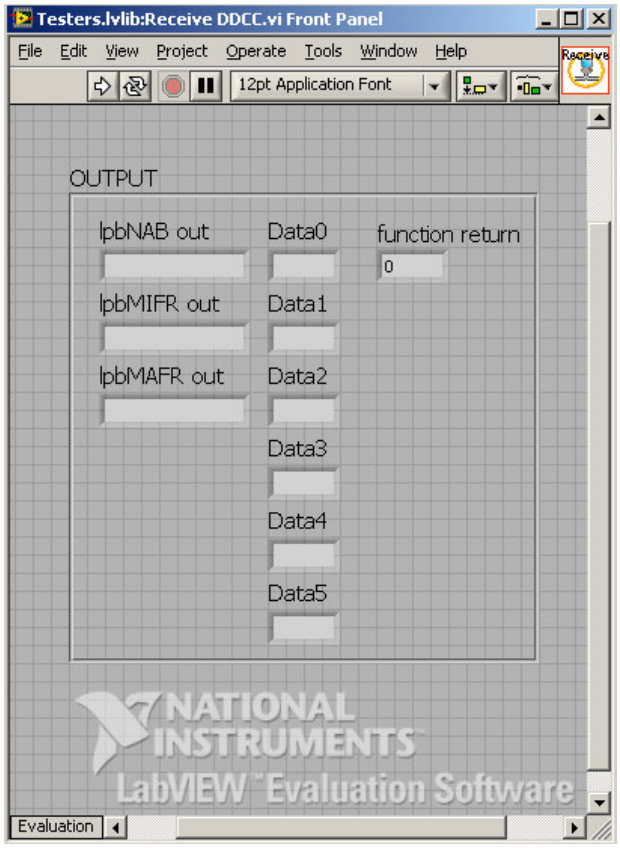
Jokaisen ulostulevan datan osoitinlaatikon ominaisuuksista pystyttiin asettamaan esitysmuoto kulloisellekin datalle. Työssä käytettiin heksadesimaalista esitysmuotoa. Kuvissa 61 ja 62 on esitetty kyseisen muodon asettaminen.




*Kuva 61. Näpäytys hiiren oikealla näppäimellä kuvakkeen päällä; Properties*




*Kuva 62. Välilehti Appearance; Display Style; Hexadecimal*




*Kuva 63. Vastaanottolohkon etupaneeli*

### 3.5 Moduulin ID-numeron tarkistaminen

Numeerinen 1-arvo tuodaan sisään for-lohkoon, joka sisältää lähetyslohkon (ks. liite 3). *For*-silmukka toistetaan yhden kerran, jonka jälkeen numeerinen 1-arvo siirtyy seuraavaan *for*-lohkoon. Toisen *for*-lohkon sisälle on sijoitettu vastaanottolohko, jonka kautta pystytään lukemaan laajennusmoduulin ID-koodi *Data 2*:sta. Vastaanotettu ASCII-koodi muutetaan desimaaliluvuksi, käyttämällä sitä välillä taulukkomuodossa. Desimaalilukua verrataan laajennusmoduulille tehdyn *Excel*-taulukon [5, 6, 7, 8] *Data 2*:n heksadesimaaliseen vastauskoodiin.

Kun ID-koodi tulkitaan oikeaksi, *Board ID Answer* -moduulin ulostulo asettuu loogiseen 1-tilaan ja etupaneelin ID-merkkivalo syttyy palamaan. Kun taas ID-koodi tulkitaan vääräksi, ulostulo asettuu loogiseen 0-tilaan ja merkkivalo pysyy pois päältä. Ulostulon asettuessa loogiseen 0-tilaan, testausohjelma lakkaa toimimasta ja palaa alkutilanteeseen.

### 3.6 FPGA-piirillä varustetun laajennusmoduulin logiikkaversion tarkistus

0- tai 1-arvo tuodaan ID-lohkolta logiikkaversiolohkon sisäänmenoon. Logiikkaversiolohko koostuu kahdesta erillisestä *for*-lohkosta (ks. liite 3). Ensimmäisessä *for*-lohkossa lähetetään heksadesimaalinen logiikkaversiokysely laajennusmoduulille. Toisen *for*-lohkon sisällä luetaan FPGA:n rekistereistä saatu ASCII-koodinen logiikkavastaus.

ASCII-koodit tulostetaan etupaneelille *Data 0* - *Data 3* -tietopaikoista. Etupaneelille tulostuu VIE4, jos kaikki on kunnossa. VIE4 on FPGA-piirillä käytetty, ABB:lle räätälöity logiikka. Jokainen ASCII-koodi vastaa yhtä binääritavua. Version tarkastus ja sen hyväksyminen tapahtuu ainoastaan ASCII -koodin V kohdalla. Jos ASCII-koodi V toteutuu, logiikkaversiolohkosta jatkaa matkaa looginen 1-arvo, jos ei, niin jatkaa looginen 0-arvo. (Tämä logiikkaversion tarkastus ei koske FIO-01:tä, koska se sisältää ASIC-mikropiirin).

### 3.7 Laajennusmoduulille tehtävät muut alustukset

Laajennusmoduulin muut alustukset tehdään sen jälkeen, kun laajennusmoduulin alustava alustus (Init DDCP.vi) on tehty, sekä kortin ID-numero ja logiikkaversio on onnistuneesti tunnistettu. Tämän jälkeen alustetaan laajennusmoduulikohtaiset asetukset, joita tarvitaan kortin testaamiseen. Esim. A/D Config, sisäisen yksikanavaisen 16-bittisen analogia/digitaali -muuntimen alustus (ks. liite 4). *Enable Relay Operation* -kohdasta asetetaan päälle releulostulot (ks. liite 5).

Kun kaikki laajennusmoduulille tehtävät alustukset on käyty läpi, syttyy testausohjelman etupaneelin nimen viereinen pyöreä merkkivalo palamaan. Merkkivalo kertoo, ovatko kaikki laajennusmoduulille tehtävät alustukset ja tarkastukset menneet läpi. Merkkivalo palaa vihreänä, kun kaikki on OK ja punaisena, jos jokin kohdista ei ole mennyt läpi. Punaisen merkkivalon syttyessä testausohjelma lakkaa toimimasta ja palaa alkutilanteeseen.

### 3.8 Testausohjelmien pääkytkimet

Vain yksi pääkytkin pystyy kerrallaan olemaan ON-positiossa. Jos toinen pääkytkin asetetaan ON-positioon toisen kytkimen ollessa päällä, kummankin pääkytkimen alaiset I/O-toiminnot lakkaavat toimimasta, ja etupaneelin merkkivalot menevät pois päältä. Toteutus on tehty pääkytkimen jälkeisellä yhdellä AND-portilla, jossa on 5 - 7 sisäänmenoa riippuen laajennusmoduulista sekä yhdellä NOR-portilla (ks. liite 3).

### 3.9 While-toistorakenteen sisäistenlohkojen samankaltaisuus

Testausohjelmien lohkokaaviosta (ks. liitteet 3 - 6) pystytään havaitsemaan, että *while*-silmukan sisällä olevat tulo- tai lähtölohkot, tai niiden yhdistelmät, ovat hyvin samankaltaisia toistensa kanssa. Lähinnä ainut eroavaisuus

ilmenee lähetyslohkojen datassa, vastaanottavan lohkon datassa sekä joissakin kaavoissa ja rajojen määrittelyissä.

Esim. FIO-01:n lohkokaaviosta (ks. liite 6) pystytään havaitsemaan, että releteotetus on samankaltainen RO1:lle kuin RO2:lle sekä digitaaliset tulo/lähtö toteutukset ovat samankaltaiset toisiinsa nähden (DIO1, DIO2, DIO3, DIO4). Ainoa poikkeavuus ilmenee ASIC:lle syötetyssä ja ASIC:lta vastaanotetussa datassa. Datat eroavat jokaisessa eri I/O-toteutuksessa lukuunottamatta nolla-tilaa, joka on yhteinen kaikille lähtötoteutuksille (RO1, RO2, DO1 - DO4).

Suurin osa ohjelman logiikkapiireistä liittyy jollakin tavalla I/O-liitäntöjen nollaukseen, etupaneelin merkkivaloihin ja ei-haluttujen samanaikaisten toiminnallisuuksien pois kitkemiseen.

### 3.10 Lohkokaavion selvennys digitaali I/O -laajennusmoduulista FIO-01

Kaikkien lähtöjen (DO1 - DO4, RO1, RO2) nollaus on toteutettu samalla lähetyslohkolla (ks. liite 6). Nollaavan lähetyslohkon edessä olevilla digitaalipiireillä on estetty nollauksen päällemeno ei-halutussa tilanteessa. Esim. jos relelähtö RO1 on päällä ja DIO1:n pääkytkin asetetaan ON-positioon, kumpikin I/O-toiminto kytkeytyy pois päältä.

Relelähdöt pystytään asettamaan päälle yhdellä lähetyslohkolla. Jatkuvatoistoinen releen päälle/pois-testi (*Repeat test,* ks. liite 6), on esitetty *while* -silmukan oikeassa ylälaidassa.

Toiminnossa RO1 *High* (*Repeat test*) -lähetyslohkon tuloon pystytään syöttämään millisekuntiarvo (ms) etupaneelin syötekentästä (ks. liite 2). RO1 *High* (*Repeat test*) -kohdassa asetetaan rele päälle (ks. liite 6). Sama syötetty ms-arvo jatkaa releen nollauslohkoon RO1 *Low* (*Repeat test*), kerrottuna kahdella, josta on saatu taajuus ja sopiva viive ennen releen pois-päältä kytkemistä. Releen päälle/pois-lohkojen välissä on myös laskuri, joka laskee päälle/pois-kytkentöjen määrän. Yksi jakso sisältää releen päälle- ja poiskytkennän.

Digitaalinen tulo/lähtö -toiminto (DIO1, DIO2) on jaettu kolmeen *for* -silmukkaan. Digitaalitulo tarvitsee vastaanottolohkon. Vastaanottolohkossa on tarkasteltu syötetyn digitaalitulon loogisen tilan 0 tai 1 paikkansapitävyys data paikoissa *Data 0* - *Data 5* (ks. liite 6). Loogisen 0-tilan arvo on jokaisen digitaalitulon *Data 1* -paikassa sama, heksadesimaalisena lukuna 40 eli desimaalilukuna 64. Loogista tilaa 1 vastaava desimaaliarvo *Data 1* -paikassa on eri jokaisessa digitaalitulossa. Esim. DIO1:ssä loogista 1-tilaa vastaa desimaaliluku 96. Molemmissa loogisissa tiloissa on myös samanaikaisesti oltava *Data 2*:n arvon desimaalisena 170, ja jäljelle jääneiden datalähtöjen arvojen on oltava 0.

Etupaneeli (ks. liite 2) näyttää myös 8-bittisenä binääriarvona *Data 1*:n arvon. Binääriseen muotoon looginen arvo on saatu muuntamalla desimaaliarvo ensin heksadesimaaliseen muotoon, minkä jälkeen arvo on muutettu binääriseen muotoon.

### 3.11 Lohkokaavion selvennys analogia I/O -laajennusmoduulista FIO-11

Digitaalilähtöjen sekä analogilähdön nollaus on toteutettu while -toistorakenteen sisällä peräkkäisissä lähetyslohkoissa *DO1, DO2 Low* ja *AO1 Low* (ks. liite 3). Analogiaulostulon (AO1) kanssa samaan *for*-silmukkaan on tehty visuaalisesti näyttävä neljän valon satunnaisesti ja peräkkäisesti syttyvä/sammuva -järjestelmä. Valojärjestelmällä on pyritty lisäämään etupaneeliin näyttävyyttä. Valoryhmittymä ei häiritse testausohjelmiston muita toimintoja.

AO1:n alussa on määritetty ehdot, joilla hyväksytysti sisään pystytään asettamaan vain luku 0 - 20. Muissa tapauksissa testausohjelma nollaa AO1-lähdön. Ensimmäisessä *case*-rakenteessa skaalataan desimaaliluku mA -arvoksi sopivalla kertoimella. Kerroin on katsottu spesifikaatiosta [14]. Sen jälkeen skaalattu arvo on muunnettu heksadesimaali- sekä desimaaliarvoksi. Toisessa *case*-rakenteessa heksadesimaaliarvo on jaettu kahteen eri osaan, jolloin on saatu kaksi erillistä heksadesimaaliarvoa. Arvot on muutettu takaisin numeeriseen muotoon, jotta ne pystytään lähettämään lähetyslohkon paikkojen *Data 4* ja *Data 5* kautta FPGA:lle. Jos ensimmäiseen

*case*-rakenteeseen syötetty luku on erisuuri kuin 0 - 20, niin *case*-lohkon *False* -puoli toteutuu. Kun *False* -puoli toteutuu, ovat lähtevät *Data 4* ja *Data 5* arvot nollia, jolloin AO1 nollautuu.

Digitaalinen tulo/lähtö on toteutettu samalla periaatteella kuin FIO-01:ssä. Ainut huomattava muutos on tulon heksadesimaalisen arvon tarkastamisessa. FIO-11:sta heksadesimaaliarvoa on tarkasteltu bittitasolta, jotta on pystytty varmistamaan looginen 0- tai 1 -tila. Tämä loogisten tilojen paikkansapitävyys on tarkastettu binäärisen muunnoksen jälkeisillä logiikkaporteilla.

Esim. kaksimerkkinen heksadesimaaliarvo Fx, joka binäärisessä muodossa mahtuu yhteen tavuun. Ensimmäiset neljä bittiä ovat ykkösiä ja loput bitit voivat olla mitä tahansa, jälkimmäisen heksadesimaaliarvon ollessa x. Kyseisestä tapauksesta tulisi tutkia loogisilla porttipiireillä, että tavun eniten merkitsevät neljä bittiä ovat ykkösiä.

Analogiatuloon on syötettävä jokin virta- tai jännitetaso. Laajennusmoduulin mekaaninen valintakytkin täytyy olla oikeassa positiossa. Ohjelman etupaneelin pääkytkin sekä valintakytkin täytyy olla myös oikeassa positiossa, jonka jälkeen pystytään lähettämään lukupyyntö lähetyslohkolla. Vastaanottolohkolla vastaanotetut arvot *Data 4* ja *Data 5* asetetaan peräkkäiseksi jonoksi. Jono muutetaan taulukkomuotoon ja sen kautta numeeriseen muotoon.

Etupaneelin valintakytkimellä pystytään valitsemaan lohkokaavion valintarakenteesta tosi- tai epätosilohko. Jos valintakytkin on lepotilassa, tarkastellaan jännitteen raja-arvoja (kolme eri tapausta), sekä numeerinen arvo skaalataan jännitearvoksi näytölle. Jos valintakytkin on asennossa 1, tarkastellaan virtaraja-arvoa ja numeerinen arvo skaalataan virta-arvoksi sopivalla kertoimella. Jännitteen raja-arvot on tarkastettu 0 V:n, 10 V:n ja -10 V:n kohdissa. Virran raja-arvot on tarkastettu 20 mA:n kohdassa. Numeeriset raja-arvot on katsottu spesifikaatiosta [6].

### 3.12 Lohkokaavion selvennys analogia I/O -laajennusmoduulista FIO-21

Analogiatulo (ks. liite 4) on lähes samanlainen kuin FIO-11:sta (vrt. liite 3). Erona on, että FIO-21:n analogiatuloon pystytään syöttämään ainoastaan virtaa, kun taas FIO-11:sta analogiatuloihin pystytään syöttämään myös jännitettä. FIO-21:n analogiatulon raja-arvot on tarkastettu 10 mA:n ja 20 mA:n kohdissa. FIO-21:n analogiatulo sisältää vain yhden kytkimen.

(Analogialähtö, vrt. FIO-11 analogialähtöön.)

Digitaalitulo (DI) on samantapainen kuin FIO-11:sta digitaalinen tulo/lähtö -toiminto (ks. 3.11). Ainut eroavaisuus moduuleiden välillä on, että FIO-21 ei sisällä lähtöpuolta laisinkaan.

[Relelähdöt, vrt. FIO-01 (3.10).]

### 3.13 Digitaali I/O -laajennusmoduulin FIO-31 -testausohjelma

FIO-31 sisältää neljä relelähtöä, sen lohkokaavio on esitelty liitteessä 4. Relelähtöjen toimintaperiaate on sama kuin FIO-01:n relelähtöjen (vrt. 3.10). FIO-31:n testausohjelma oli helpoin toteuttaa johtuen käytetystä lähtötyypistä.

### 3.14 Testausohjelmien käyttöliittymät

Visuaalisesta näkökulmasta katsottuna, ohjelmien etupaneeleista on pyritty tekemään silmää miellyttäviä, nykyaikaisia tai jopa futuristisia toteutuksia. Myös tulojen ja lähtöjen toistuva samankaltainen rakenne sekä helppo lukuisuus on pyritty ottamaan huomioon etupaneeleita suunniteltaessa. Etupaneelit on toteutettu erilaisin laatikkorakentein, värityksineen, fontteja ja merkkivaloja mukauttaen. Testausjärjestelmät on suunnattu alan

ammatti-ihmisten käyttöön, apua antamaan analogia/digitaali I/O -laajennusmoduuleiden vian paikantamisessa.

# 4 TESTAUSJÄRJESTELMIEN (FIO-01, FIO-11, FIO-21, FIO-31) KÄYTTÖOHJE

Ohjelmistolla pystytään selvittämään laajennusmoduuleiden (FIO-01, FIO-11, FIO-21, FIO-31) viallinen tulo/lähtö-liitäntä. Jos moduuleiden vika halutaan tarkasti paikantaa, joudutaan moduuleita tutkimaan myös komponenttitasolla, johon on olemassa omat menettelytavat.

Moduuleiden tulojen ja lähtöjen testaustavat on esitelty kohdassa 2.1. Testausjärjestelmän mekaaniset kytkennät (ks. 2.6).

Ohjelma pystytään avaamaan LabVIEW-ympäristöissä tupla-näpäyttämällä hiiren vasenta näppäintä ohjelman *exe*-tiedoston päällä (esim. FIO-01.exe). Tällöin FIO-01-testausohjelman etupaneeli ilmestyy näytölle (ks. liite 2). Testausohjelma pystytään käynnistämään ylälaidassa olevasta valkoisesta nuoli-näppäimestä (*RUN*). Ohjelma pystytään sammuttamaan punaisesta kahdeksankulmiosta, jossa on pienet valkoiset reunukset.

Testausohjelmien lohkot ovat samannäköisiä, sillä on pyritty helpottamaan käyttäjän etupaneelin lukua. Kunkin lohkon pääkytkin (*ON/OFF*) löytyy lohkon vasemmasta laidasta. Lohkoista pystytään havaitsemaan syötetäänkö tai mitataanko jännitettä/virtaa. Ainoastaan valkoisiin laatikkoihin (syötteisiin) pystytään asettamaan arvoja. Harmaat laatikot ovat osoitin kenttiä, niihin ei pystytä asettamaan arvoja. Ainoastaan yksi tulo tai lähtö tai tulo/lähtö pystyy olemaan kerrallaan päällä.

## 4.1 Laajennusmoduuleiden alustus

Testausohjelmien alustuslohko sijaitsee etupaneeleiden yläosassa heti laajennusmoduulin nimen alapuolella. Se sisältää kolme eri kohtaa, lukuun ottamatta FIO-01, joka sisältää kaksi kohtaa, koska FIO-01:ssä käytetään ASIC-piiriä. Kun laajennusmoduulille tehtävät alustukset on kunnossa, syttyy

kaikki siniset merkkivalot palamaan, sekä laajennusmoduulin vieressä oleva merkkivalo vaihtuu vihreäksi.

Kun laajennusmoduulin nimen vieressä oleva merkkivalo palaa punaisena, tarkoittaa se, että jokin laajennusmoduulille alussa tehtävistä alustuksista on mennyt vikaan (vika-tila). Tällöin testausohjelma palaa alkutilanteeseen.

Alustuslohkon vika-tilassa tarkista seuraavat asiat:

- testausjärjestelmään liittyvät kytkennät
- JEXT-01:n käyttöjännite
- tietokoneen rinnakkaiset DDCS-kommunikointi -järjestelmät (saattavat häiritä toimintaa)
- RUSB-02:n Active-merkkivalo (vihreä).

## 4.2 Digitaalinen tulo/lähtö -liitäntä (DIOx)

Haluttu digitaalinen tulo/lähtö (DIOx) asetetaan päälle pääkytkimestä. Valintakytkimestä pystytään valitsemaan onko päällä tulopuoli (*input*) vai lähtöpuoli (*output*). Asetettaessa pääkytkin päälle, oletuksena on, että tulopuoli on asetettu päälle. Ulostulopuolelta pystytään kytkimellä asettamaan ulostulo loogiseen tilaan 0 tai 1.

Tulopuolen *Boolean value* -kohta kertoo loogisentilan. Tulossa näkyy *Data 2*:n heksadesimaaliarvo sekä *Data 1*:n arvo heksadesimaalisena, desimaalisena ja binäärisenä. Jos tuloarvot ovat kunnossa, syttyy (*Values OK?*) -kohdan merkkivalo vihreäksi. Kun (*Values OK?*)-merkkivalo palaa punaisena, on arvossa tai arvoissa virheellisyyksiä.

### 4.3 Relelähtö (ROx)

Relelähtö (ROx) voidaan jakaa kahteen eri moodiin: toinen on normaalimoodi ja toinen on jatkuvatoistoinen-moodi (*ROx ON/OFF Repeat, ks. liite 7*). Normaalissa moodissa pääkytkimestä asetetaan releto iminto päälle ja releen tilaa pystytään vaihtamaan kulloisenkin relelähdön kytkimestä.

Jatkuvatoistoisessa moodissa pystytään asettamaan puolikas jaksonaika kohtaan *Input (ms)*, josta ohjelma laskee kokonaisen jaksonajan kertomalla puolikkaan jaksonajan kahdella. Toiminto asetetaan päälle pääkytkimestä, jolloin relelähtö alkaa vaihtaa tilaansa käyttäjän haluamalla taajuudella. Yksi jakso sisältää releen päälle/pois-kytkennän. Toistettujen jaksojen määrä on laskettu kohdassa *ON/OFF Repeates*.

### 4.4 Analogiatulo (AIx)

Pääkytkimestä pystytään asettamaan päälle haluttu analogiatulo (ks. liite 7). Valintakytkimestä pystytään valitsemaan käytetäänkö tuloa jännitteen vai virran mittaamiseen. Oletuksena on jännitteen mittaus. Sisään syötetty jännite/virta-arvo on skaalattu ohjelman sisällä, ja skaalattu arvo tulostuu kyseisen analogiatulon harmaaseen osoitinlaatikkoon.

Sisään syötettyä virta/jännite-arvoa pystytään tutkimaan ohjelman kohdassa *Limits detector*. *Limits detector* -kohdassa on tutkittu, onko syötetty arvo hyväksytysti rajojen sisäpuolella. Virralla on tutkittu raja-arvo 20 mA:n kohdassa. Jännitteellä raja-arvot on tutkittu 0 V:n, 10 V:n ja -10 V:n kohdissa. Jos merkkivalo palaa vihreänä on arvo rajojen sisäpuolella eli OK, jos merkkivalo palaa punaisena on arvo rajojen ulkopuolella.

Osoitin kentissä on näytetty myös *Data 4*:n ja *Data 5*:n heksadesimaaliarvot sekä niiden yhdistetty arvo heksadesimaalisena ja desimaalisena.

### 4.5 Analogialähtö (AOx)

Haluttu analogialähtö (AOx) pystytään asettamaan päälle pääkytkimestä (ks. liite 7). Valkoiseen syötekenttään pystytään asettamaan haluttu lähtöarvo 0 - 20 mA. Asetettu lähtöarvo pystytään mittaamaan ulkoisella mittalaitteella kyseisestä lähdöstä. Osoitinkentistä pystytään lukemaan heksadesimaalisena *Data 4*:n ja *Data 5*:n arvot yhdessä sekä erikseen. Desimaaliarvona näkyy yhdistetty *Data 4* ja *Data 5* arvo.

### 4.6 Digitaalitulo (DIx)

Digitaalitulo (DIx) toimii, kuten digitaalisen tulo/lähdön tulopuoli (vrt. 4.2). Digitaalitulo ei sisällä lähtöpuolta.

## 5 POHDINTAA JA JOHTOPÄÄTÖKSET

Testausohjelmistoissa käytetty visuaalinen ohjelmointiympäristö LabVIEW on hyvä vaihtoehto tekstuaalipohjaisille ohjelmointikielille. Myös tekstuaalipohjaisia ohjelmointikoodeja pystytään liittämään LabVIEW'iin. Kommunikointi laajennusmoduulin ja LabVIEW-ohjelmiston välillä pystyttiin toteuttamaan melko vaivattomasti. Toteutus tehtiin LabVIEW-ohjelmiston kirjasto -toiminnon avulla. Kirjasto-toiminnossa dll- ja h-tiedosto pystytään yhdistämään kätevällä tavalla toistensa kanssa ja niiden sisältämistä toiminnosta pystytään luomaan ohjelmaan valmiita osa-lohkoja.

Haasteita insinöörityössä asetti h-tiedoston muokkaaminen LabVIEW-ohjelmaan yhteensopivaksi. Se käytännössä tarkoitti, että h-tiedoston tekstuaalista ohjelmointikieltä jouduttiin muokkaamaan. Insinöörityössä käytettiin myös runsaasti aikaa vastaanottolohkon oikeanlaisen datan ulossaantiin.

Testausohjelmien tekovaiheessa pystyttiin havaitsemaan, että erinäisiä lohkokaavion toimintoja/lohkoja pystyttiin kätevästi sekä jouhevasti siirtämään,

muokkaamaan, lisäämään ja poistamaan LabVIEW-ohjelmiston sisällä. National Instruments on myös hyvin toteuttanut LabVIEW-ohjelmassa etupaneelin suunnittelu-osion. Työhön LabVIEW-ohjelmisto soveltui erittäin hyvin, mutta sen soveltuvuutta suurempiin ja monimutkaisempiin ohjelmointikokonaisuuksiin ei voida työn perusteella arvioida.

Testausohjelmien avulla pyritään havaitsemaan vikaantunut I/O-liitäntä analogia- ja digitaalilaajennusmoduuleista. Mutta jos jatkossa ohjelmissa havaitaan ohjelmallisia puutteellisuuksia tai erinäiset raja-arvot muuttuvat, niin testausohjelmia pystytään helposti muokkaamaan halutunlaisiksi.

## 5.1 Testausohjelmien rajat, tarkkuus ja tarvittavat kaavat

Testausohjelmien tulojen/lähtöjen raja-arvojen tarkastelu on toteutettu ABB:n spesifikaatioiden mukaisesti [5, 6, 7, 8]. Spesifikaatioissa on määrätty tietyt rajat kulloisellekin laajennuskortin I/O:lle. Erinäisissä revisioissa saattaa olla eroavaisuuksia. Rajat on määritelty testausohjelmissa tarkasti kulloisellekin laajennusmoduulille, tällöin mittaustuloksista on saatu luotettavia. Ohjelmien täydellinen toimivuus/luotettavuus pystytään havaitsemaan, vasta useiden laajennusmoduuleille tehtyjen testien kautta.

Etupaneelin analogiatulojen osoitinkentät näyttävät kolmen desimaalin tarkkuudella mitatun mA- tai V-arvon. Analogialähtöjen ja jatkuvatoistoisten relelähtöjen syötekentille ei ole asetettu tarkkuutta. Tarkkuutena on käytetty LabVIEW-ohjelman oletusarvoja.

Relelähtöjen toisto-toiminnossa taajuus eli jaksonaika (*period*) on laskettu kertomalla 2:lla sisään syötetty ms-arvo (*input ms*). Releen päälle/pois (*on/off*) -kytkentöjen määrä on laskettu ohjelmassa. Yhteen jaksoon sisältyy releen päälle ja poispäältä kytkentä.

### *5.1.1 Digitaali I/O -laajennusmoduulin FIO-01 tarkastelu*

Kaikissa digitaalituloissa loogista 0-tilaa vastaa *Data 1*:stä saatu heksadesimaaliarvo 40, joka on desimaalisena 64. Samanaikaisesti *Data 2*:n arvo

täytyy olla heksadesimaaliarvona AA sekä *Data 0*:n, *Data 3*:n, *Data 4*:n ja *Data 5*:n arvot tulee olla nollia.

Digitaalitulo 1:n loogista 1-tilaa vastaa

- *Data 1*, heksadesimaaliarvo 60 (desimaaliarvona 96)
- *Data 2*, heksadesimaaliarvo AA
- *Data 0*, *Data 3*, *Data 4*, *Data 5* tulee olla nollia.

Digitaalitulo 2:n loogista 1-tilaa vastaa

- *Data 1*, heksadesimaaliarvo 50 (desimaaliarvona 80)
- *Data 2*, heksadesimaaliarvo AA
- *Data 0*, *Data 3*, *Data 4*, *Data 5* tulee olla nollia.

Digitaalitulo 3:n loogista 1-tilaa vastaa

- *Data 1*, heksadesimaaliarvo 48 (desimaaliarvona 72)
- *Data 2*, heksadesimaaliarvo AA
- *Data 0*, *Data 3*, *Data 4*, *Data 5* tulee olla nollia.

Digitaalitulo 4:n loogista 1-tilaa vastaa

- *Data 1*, heksadesimaaliarvo 44 (desimaaliarvona 68)
- *Data 2*, heksadesimaaliarvo AA
- *Data 0*, *Data 3*, *Data 4*, *Data 5* tulee olla nollia.

Digitaalilähdöille on spesifikaatiossa määrätty seuraavaa:

- poispäältä-tila (looginen 0 -tila), ulostulojännitteen tulee olla (0 - 0,3) V
- päällä-tila (looginen 1 -tila), ulostulojännitteen tulee olla (22 - 26) V. [5.]

### *5.1.2 Analogia I/O -laajennusmoduulin FIO-11 tarkastelu*

Analogiatulojen (AI1, AI2, AI3) rajat ovat samat riippumatta laajennusmoduulin versiosta. Rajat ovat määritetty spesifikaatiossa. Rajat on tarkasteltu kohdissa *Data 4* ja *Data 5*. Esim. *Data 4* → FF (hex) ja *Data 5* → FF (hex), peräkkäin ne ovat FFFF → ja numeerinen arvo on 65 535.

Jännite puolella testausohjelmassa on tutkittu seuraavat raja-arvot:

- 0 V, desimaaliarvo tulee olla 32 000 - 33 000
- 10 V, desimaaliarvo tulee olla 3 000 - 4 000
- -10 V, desimaalisarvo tulee olla 61 500 - 62 500.

Virta puolella testausohjelmassa on tutkittu seuraava raja-arvo:

- 20 mA, desimaaliarvo tulee olla 2 500 - 3 500. [6.]

Analogiatulojen virta/jännite-skaalaus on tehty spesifikaation mukaan. Molemmissa skaalauksissa *Data 4* ja *Data 5* arvot on yhdistetty sekä muutettu desimaaliseen muotoon (merkattu **Y**:llä).

Virta skaalauksessa **Y**:stä on vähennetty 32 778 ja erotus on jaettu luvulla (-1 479,5). Jännite skaalauksessa **Y**:stä on vähennetty 32 776,5 ja erotus on jaettu luvulla (-2 904). [14.]

Analogialähdön virta on skaalattu saadun s-postissa tulleen kaavan (ks. liite 8) perusteella, siinä *Data 4*:n ja *Data 5*:n yhdistetty desimaaliarvo on kerrottu 163,1, ja tulosta on vähennetty 0,88074. [15.]

Digitaalitulojen loogista 0-tilaa vastaa

- *Data 0*, heksadesimaaliarvo 3
- *Data 1*, heksadesimaaliarvo 10
- *Data 2* ja *Data 3* heksadesimaaliarvo 0
- *Data 4*, heksadesimaaliarvo **XX**
- *Data 5*, heksadesimaaliarvo C**X**.

**X**-merkki tarkoittaa, että arvo voi olla mikä tahansa. Esim. C**X**:n 8-bittinen binääriesitys on muotoa 1100**XXXX**. Ohjelmassa tavusta tutkittaisiin ainoastaan neljä enintenmerkitsevää bittiä.

Digitaalitulo 1:n loogista 1-tilaa vastaa, samat *Data 0* - *Data 4* arvot kuin digitaalitulon loogisessa 0-tilassa (ks. edellä), ainoastaan *Data 5*:n arvo on muuttunut EX:ään. Sama pätee digitaalitulo 2:n loogiseen 1-tilaan, ainoastaan sen *Data 5* arvo on muuttunut DX:ään.

Digitaalilähdöille on spesifikaatiossa määrätty seuraavaa,

- päällä (looginen 1-tila), ulostulo jännitteen tulee olla revisioissa B ,C, D, E (23,5 - 24,9) V ja revisio A:ssa (21,5 - 24,5) V.

(Manuaalisvisuaalisesti tarkasteltava, mitatusta digitaaliulostulosta.) [6.]

### *5.1.3 Analogia I/O -laajennusmoduulin FIO-21 tarkastelu*

Analogiatulossa on tutkittu kahden eri kohdan rajat (10 mA ja 20 mA). Kymmenellä mA:lla numeerisen arvon täytyy olla 29 000 - 31 500 ja kahdellakymmenellä mA:lla numeerisen arvon täytyy olla 58 000 - 61 000. [7.]

Analogiatulon virta on skaalattu mA:si s-postista saadun kaavan perusteella (ks. liite 8). *Data 4*:n sekä *Data 5*:n arvot on yhdistetty ja yhdistetty arvo on jaettu 2 904.

Analogialähdön virta on skaalattu myös s-postista saadun kaavan perusteella (ks. liite 8). *Data 4*:n ja *Data 5*:n yhdistetty desimaaliarvo on kerrottu 186:lla, jonka tulosta on vähennetty 0,88074. [15.]

Digitaalitulon, loogista 0-tilaa vastaa,

- *Data 0*, heksadesimaaliarvo 3
- *Data 1*, heksadesimaaliarvo 10
- *Data 2* ja *Data 3* heksadesimaaliarvo 0
- *Data 4*, heksadesimaaliarvo XX
- *Data 5*, heksadesimaaliarvo X8.

Digitaalitulon, loogista 1-tilaa vastaa,

- *Data 0*, heksadesimaaliarvo 3
- *Data 1*, heksadesimaaliarvo 10
- *Data 2* ja *Data 3* heksadesimaaliarvo 0
- *Data 4*, heksadesimaaliarvo XX
- *Data 5*, heksadesimaaliarvo XC. [7.]

## 5.2 Testausjärjestelmien soveltuvuus laboratorio/kenttä-käyttöön

Testausjärjestelmiä lähdettiin suunnittelemaan laboratoriokäyttöön. Tarkoituksena oli selvittää laajennusmoduuleista vikaantunut I/O-liitäntä mahdollisimman nopeasti ja luotettavasti. Tutkittavat moduulit ovat tulleet asiakaspalautuksina ympärimaailmaa, eli vikaantuneina. Myös vikaantuneita käyttämättömiä moduuleita tutkitaan jonkin verran.

Testausjärjestelmien ohjelmien -toteutus on tehty melko yksinkertaisella tavalla. Ohjelmat on pyritty tekemään idioottivarmasti toimiviksi ja rajoissa pysyviksi, sekä visuaalisesti näyttämään johdonmukaisilta ja helppolukuisilta. Rajoittunut aika sääteli lopputuloksen, kuten luultavasti melko usein opinnäytetöissä käy. Testausjärjestelmät soveltuvat hyvin laboratoriokäyttöön. Kenttäkäytössä voi ongelmia tuottaa JEXT-01-adapterin tarvitsema +24 V:n syöttöjännite. Muuten testausjärjestelmä sopii mainiosti myös kenttäkäyttöön.

Jos esim. kenttäkäytössä huoltoinsinööri haluaa tutkia I/O -laajennusmoduulia, hän tarvitsee kannettavan tietokoneen, jossa on tarvittava testausohjelma, yleismittarin, testausjärjestelmän mekaaniset osat (JEXT-01, RUSB-02) ja niiden väliset kaapelit. Lisäksi tarvitaan esim. signaalikalibraattorin, jolla pystytään tutkimaan kaikki I/O-liitännät lukuunottamatta relelähtöjä, jotka voidaan tutkia esim. yleismittarin ohmi-alueella. Myös käyttöjännite +24 V tarvitaan JEXT-01-adapterille.

Testausjärjestelmän kytkennät eivät vie paljoaan aikaa, jos johdot ja kaikki liittimet ovat asianmukaiset. Luultavasti kannettavantietokoneen käynnistäminen vie enemmän aikaa kuin kytkentöjen rakentaminen. Testaamisessa kuluu aikaa muutamia minuutteja riippuen testattavasta laajennusmoduulista ja testaajan aikaisemmasta testausohjelman käyttökokemuksesta.

## 5.3 Testausjärjestelmien taloudellisuus

Uudella testausjärjestelmällä laajennusmoduuleiden testausaikaa on saatu pienennettyä. Testausajan merkitys on suuri testausympäristöissä, erityisesti

tuotannon testausjärjestelmissä virheiden havaitsemiseen käytettyaika tulee olla pieni. Insinöörityössä käytettyjen laajennusmoduuleiden testausaikaan vaikuttaa, käyttöliittymä, kytkennät ja käyttäjän aikaisempi ohjelman käyttö kokemus.

Myös laajennusmoduuleiden valmistaja Wong's Electronics Co., Ltd. on kiinnostunut testausjärjestelmän dll- ja h-tiedostoista. Tiedostojen avulla Wong's pystyy kehittämään omia laajennusmoduuleiden testausjärjestelmiä. ABB hyötyy taloudellisesti siitä, kun vialliset moduulit pystytään jo valmistajalla havaitsemaan ja korjaamaan.

## 5.4 Testausjärjestelmien paranneltavuus

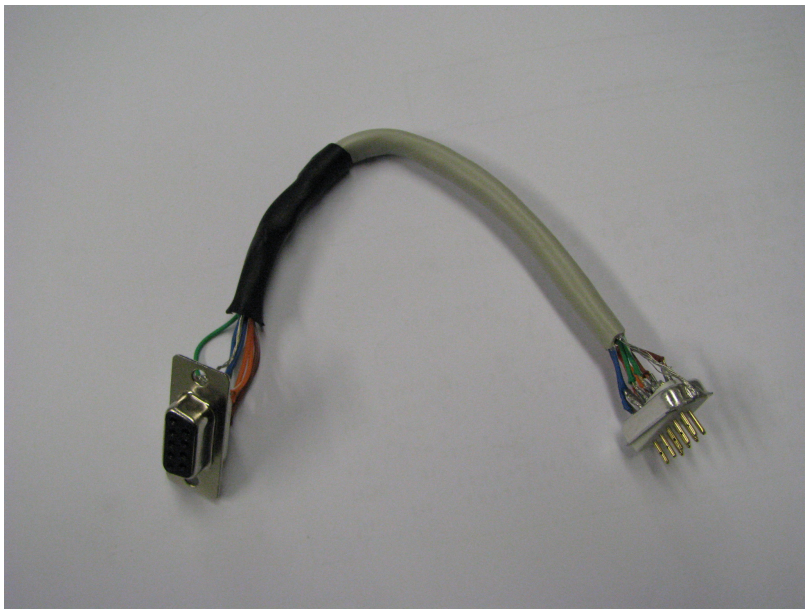
Lähes kaikkia testausjärjestelmiä pystytään aina parantamaan. Mutta se ei aina takaa, että DUT olisi nopeammin tai luotettavammin testattavissa. Työn kaikkia testausohjelmia pystytään jatkokehittämään erilaisiin testausympäristöihin sopiviksi. Testausohjelman, jonka voisi laittaa päälle, ja se kävisi automaattisesti kaikki I/O-liitännät yksitellen askeleittain lävitse sekä tulostaisi testitulokset, olisi aivan omiaan tässäkin sovelluksessa.

Lähes automaattisessa sovelluksessa tarvittaisiin lisää erinäisiä alustavia kytkentöjä tehtäväksi laajennusmoduulille, sekä jonkinmoinen lisä-moduulin, jotta tarvittavat takaisin kytkennät saadaan toteutettua. Sovellus ei välttämättä soveltuisi kenttäkäyttöön yhtä hyvin kuin esim. laboratoorio- tai kokoonpanolinjastokäyttöön. Jos on paljon alustavia kytkentöjä, joissa on erinäisiä johtimia/liittimiä, niin kentällä usein erinäisillä osilla on tapana kadota tai unohtua jonnekin. Käytännössä lähes automaattiseen testaus tarkoitukseen sopiva ohjelmisto voisi olla esim. National Instrumentsìn TestStand-testausympäristö.

## 5.5 Resolverin ja enkooderin rajapintalaajennusmoduuleiden testerit

Resolvereilla (*resolver*) ja enkoodereilla (*encoder*) pystytään mittaamaan moottoreiden akselin paikkatietoa tai kierrosnopeutta. Resolveri on analoginen ja enkooderi on digitaalinen. Resolverin ja enkooderin rajapintalaajennusmoduuleille (FEN-01, FEN-11, FEN-21, FEN-31) ehdittiin tekemään alustavat ohjelmistot LabVIEW'llä.

Ohjelmilla pystytään alustamaan moduulit, myös moduuleiden TTL tulo/lähtö-liitännät pystytään emuloimaan, laajennusmoduuleiden emulaatiolähdön avulla. Emuloinnissa tarvitaan kaapeli TTL-lähdön ja -tulon välille (ks. kuva 64). Kaapelissa kanavoiden (A+, A-, B+, B-, Z+, Z-) -signaalit kulkevat suoraan lähdöstä tuloon ja tulosta voidaan tarkastaa, onko kaikki lähetettävä data saapunut perille. Rajapintalaajennusmoduuleita on jatkokehitettävä, jotta kaikki muut moduuleiden tulot ja lähdöt pystytään testaamaan.



*Kuva 64. Emuloinnissa käytetty datakaapeli*

## LÄHTEET


[1] ABB Drives, Quick Guide. FIO-01 Digital I/O Extension. 2006.

[2] ABB Drives, Quick Guide. FIO-31 Digital I/O Extension. 2009.

[3] ABB Drives, Quick Guide. FIO-11 Analog I/O Extension. 2006.

[4] ABB Drives, Quick Guide. FIO-21 Analog I/O Extension. 2008.

[5] Rosti, Janne, FIOE-01 Test Steps and Limits Specification. 00529670.xls. Helsinki: ABB Oy, Drives. 02/2007.

[6] Rosti, Janne, FIOE-11 Test Steps and Limits Specification. 00529851.xls. Helsinki: ABB Oy, Drives. 02/2007.

[7] Rosti, Janne, FIOE-21 Test Steps and Limits Specification. 00580415.xls. Helsinki: ABB Oy, Drives. 01/2008.

[8] Hjerppe, Sauli, FIOE-31 Test Steps and Limits Specification. 00589924.xls. Helsinki: ABB Oy, Drives. 05/2009.

[9] Navarro-Prieto, Raquel - Cañas, Jose J., Are visual programming languages better? The role of imagery in program comprehension. 1999. [Viitattu 13.04.2010].Saatavissa: http://folk.uio.no/christho/inf3240/downloads/Navarro-Prieto_visual.pdf

[10] Pakoma, Paavo, Rekursiivisten algoritmien toteuttaminen labVIEW-ohjemointiympäristössä. 2008. [Viitattu 13.04.2010]. Saatavissa: http://www.kampus.uku.fi/gradut/2008/6279.pdf

[11] Mondofacto, Visual programming language. 02/1995. [Viitattu 13.04.2010]. Saatavissa: http://www.mondofacto.com/facts/dictionary?visual+programming+language

[12] Bishop Robert H., Learning with LabVIEW 8. New Jersey: Pearson Education, Inc. 2007.

[13] LabVIEW 2009, LabVIEW Help. 06/2008.

[14] Norrkniivila, Mikko, Extension I/O driver for FIOE-xxSoftware design specication. sw_spec_3860571.doc. 03/2008.

[15] Korpela, Ari, ABB Oy, Drives. Re: Skaalaus [sähköpostiviesti]. Vastaanottaja Jarmo Strand. Lähetetty 09.03.2010 [viitattu 19.04.2010].

[16] Silander, Simo, Ohjelmoinnin perusteet ja C-kieli. [Verkkodokumentti]. 1999. [Viitattu 08.04.2010]. Saatavissa: http://cs.stadia.fi/~silas/ohjelmointi/c_opas-Title.html

## LIITTEET

```
/*=============================================================================
====
ABB Drives          Copyright (c)1997, ABB Industry Oy.  All rights re-
served.

    Project:        Drives Window
    Subproject:     Communication Driver
    Author:         Markku Polvi
    Tools:          MSVC v1.52

    Include file:   dwc_ddcp.h

    Abstract:

    This include file contains DDCS access library interface of
dwc_ddcp.dll.
    Hardware: -DDCS ISA cards, currently 2 cards supported at fixed ad-
dresses
              0x280, 0x2a0
              -DDCS PCMCIA NDPA-01 requires an additional driver for
win95,
              available on a disk.
              See Drives Window documentation for further details.

    Notes:
        1. For Labwindows usage, first #define LABWINDOWS_RULES
        then include this file.
        2. Auto Node Numbering not operating with this interface of
dwc_ddcp.dll

    Version history:

    1.00    22.10.1996/MP   first prerelease
    1.01    30.10.1996/MP   for labwindows rules and restrictions:
                            interface definitions without need of ms win-
dows.h
    1.02    25.11.1996/MP   GetVersion.. split into GetMajorVersion-
OfDDCPLib and
                            GetMinorVersionOfDDCPLib,
                            on configuration dialog: Enables now all
changes to DCR0
                            and added radiobuttons for cable length set-
ting
    1.03    28.11.1996/MP   added ReadDDCC_ISAControl()
    1.04    21.01.1997/MP   added note about timer resolution in win 3.x
to
                            SendDDCC() and ReceiveDDCC()
    1.05    15.10.1997/MP   added SendDDCCEx() and ReceiveDDCCEx() for
LabVIEW usage
                            LabVIEW CallLibrary Function does not use
this include file
    1.06    17.10.1997/MP   fixed ..Ex types from DWC_APIDLL to DWC_API
    1.07    20.11.1997/MP   defined DWC_API etc. client usage of func-
tions 32/16bit
          1.23       01.10.2007/JE         Some enhancements and RUSB-02
extensions.



Labview käytössä poistettu mm.DWC_API,FAR jotta saadaan yhdistettyä .h
file ja .dll file (SendDDCCx, ReceiveDDCCx)
```

```
          esim.
          int  DWC_API ReceiveDDCC(WORD wChannel, LPDDCCP lpDDCCP, WORD
wTimeOut_ms)
          -->        int  ReceiveDDCC(WORD wChannel, LPDDCCP lpDDCCP,
WORD wTimeOut_ms)

          int  ReceiveDDCCEx(WORD wChannel,
                      BYTE FAR *lpbNAB,
                               -->        BYTE *lpbNAB,

================================================================================
==*/


/*================================================================================
====

    PRE-DEFINITIONS

================================================================================
==*/

#ifndef _INC_DDCP_LIB_H
#define _INC_DDCP_LIB_H

/*================================================================================
======

    COMMON TYPES & DEFINITIONS

================================================================================
====*/
#ifndef _INC_WINDOWS     /*windows.h not used*/

typedef short       BOOL;
#define FALSE       0
#define TRUE        1

typedef unsigned char       BYTE;
typedef unsigned short      WORD;
typedef unsigned long       DWORD;

#define FAR         _far
#define NEAR        _near

#define PASCAL              _pascal
typedef char FAR*           LPSTR;

#endif  /*_INC_WINDOWS     windows.h not used*/

//define client usage of functions
#ifdef WIN32
    #ifndef DWC_API
        #ifdef USE_DWC_SOURCE
            #define DWC_API
        #else
            #define DWC_API __declspec(dllimport)     //DWC API functions
        #endif
        #define DWC_APIPFN *                        //pointer to DWC API
functions
```

```
    #endif
#else
    #ifndef DWC_API
        #define DWC_API FAR PASCAL                //DWC API functions
        #define DWC_APIPFN _far _pascal *         //pointer to DWC API
functions
    #endif
#endif

/*=========================================================================
====

    CONSTANTS, MACROS & TYPES

=========================================================================
==*/
/*success and error codes returned by functions*/
#define DWC_SUCCESS                     0
#define ERR_DWC_BASE                    0
#define DWC_ERR_OFFSET_DDCP             (700)
#define ERR_DWC__NOREPLY                ((short)ERR_DWC_BASE-700) /*no
reply*/
#define ERR_DWC__HW_BUSY                ((short)ERR_DWC_BASE-701)
/*Hardware transmitter busy*/
#define ERR_DWC__INI_LOG                ((short)ERR_DWC_BASE-712)
/*diagnostics log initialization failed*/
#define ERR_DWC__INI_CONFIG             ((short)ERR_DWC_BASE-713)
/*config initialization failed*/
#define ERR_DWC__INI_HW                 ((short)ERR_DWC_BASE-714)
/*hardware initialization failed*/
#define ERR_DWC__NOT_ISA                ((short)ERR_DWC_BASE-715) /*not
an ISA DDCS card on this channel*/
#define DWC_ERR_CNT                     (6)


/*=========================================================================
====
    DDCC structure definition
=========================================================================
==*/

typedef struct tagDDCCP
{
    BYTE bNAB;      /* tx/rx node address byte*/
    BYTE bMIFR;     /* minor function code*/
    BYTE bMAFR;     /* major function code*/
    BYTE bDummy;
    BYTE bData[6];  /* tx/rx data*/
} DDCCP;

typedef DDCCP NEAR* PDDCCP;
typedef DDCCP FAR*  LPDDCCP;

typedef struct tagDDCC_CTRL
{
    BYTE bNAR;      /* own node address*/
    BYTE bDCR0;     /* ddcc control 0*/
    BYTE bDCR1;     /* ddcc control 1*/
} DDCC_CTRL;

typedef DDCC_CTRL NEAR* PDDCC_CTRL;
```

```
typedef DDCC_CTRL FAR*  LPDDCC_CTRL;

/*=========================================================================
====

    DDCC / Hardware

=========================================================================
==*/
//asic type (this definition is available since version 3.23, which im-
plements
// RUSB-02)
#ifndef DDCS_TYPE_OLD // NISA-03
#define DDCS_TYPE_OLD  0
#endif
#ifndef DDCS_TYPE_PLUS           // NDPA-02
#define DDCS_TYPE_PLUS (WORD)1
#endif
#ifndef DDCS_TYPE_RUSB           // RUSB-02
#define DDCS_TYPE_RUSB (WORD)2
#endif

/*=========================================================================
====

    FUNCTION PROTOTYPES

=========================================================================
==*/
#ifndef LABWINDOWS_RULES    /*things for labwindows are at the end*/
/*=========================================================================
======

 Function: GetMajorVersionOfDDCPLib

 Description:   retrieves major version number of library

 Parameters:

 Returns: word representing MAJOR version
 Notes:   may be called without initialization

=========================================================================
====*/
WORD GetMajorVersionOfDDCPLib(void);

/*=========================================================================
======

 Function: GetMinorVersionOfDDCPLib

 Description:   retrieves minor version number of library

 Parameters:

 Returns: word representing MINOR version
 Notes:   may be called without initialization

=========================================================================
====*/
WORD GetMinorVersionOfDDCPLib(void);
```

```
/*=====================================================================
====
 Function:          InitDDCPLib

 Description:       initialises library and configures hardware

 Parameters:
    wShowDialog     flag to enable showing of channel configuration dia-
log (bit 0)
                           and searching for and programmin all BRUs (bit
1)
    lpszIniFilename name of the initialisation file, init values
stored/retrieved

 Returns:       positive count of channels initialized, otherwise nega-
tive error code.

 Notes:         Must be called after loading and checking version of li-
brary
                If least significant bit of wShowDialog-parameter is set,
a dialog
                    box is presented to user.
                Dialog box contains communication parameters settings
                and options to Load from and Save to IniFile, and Get
default
                    channel configuration
                DDCC-ISA configured to interrupts disabled
                Channel enumeration from 0..(count of channels)-1)

=====================================================================
==*/
int InitDDCPLib(WORD wShowDialog, LPSTR lpszIniFilename);

/*=====================================================================
======

 Function:      CleanupDDCPLib()

 Description:   library cleanup

 Parameters:
 Returns:
 Notes:         may be called last before unloading library

=====================================================================
====*/
void CleanupDDCPLib(void);
/*=====================================================================
====
 Function:      ReadDDCC_ISAControl()

 Description:   reads ISA control register byte of selected channel

 Parameters:
        wChannel channel number, must be valid

 Returns:       ISA control register value in lowbyte,
                ERR_DWC__NOT_ISA if requested channel is not on ISA board

 Notes:
```

```
==========================================================================
==*/
int ReadDDCC_ISAControl(WORD wChannel);
/*==========================================================================
====
 Function:      WriteDDCC_ISAControl()

 Description:   writes given value to ISA control register of selected
channel

 Parameters:
        wChannel channel number, must be valid
        bControl value to write to control register

 Returns:       returns ISA control register value after written it

 Notes:
==========================================================================
==*/
int WriteDDCC_ISAControl(WORD wChannel, BYTE bControl);

/*==========================================================================
====
 Function:      SendDDCC()

 Description:   writes given values to DDCC of selected channel

 Parameters:
        wChannel        channel number, must be valid
        lpDDCCP         points to structure containing values to send
        wTimeOut_ms     timeout in milliseconds

 Returns:       zero if successful, ERR_DWC__HW_BUSY if transmitter is
busy
                ERR_DWC_INVALID_NETADDR if invalid channel number,
                ERR_DWC_NO_CHANNEL_PRESENT if no channel present

 Notes:         1.if transmitter is busy, SendDDCC() waits maximum the
time specified
                in wTimeOut_ms parameter, if can't reserve transmitter
during
                that time, returns ERR_DWC__HW_BUSY
                2. win 3.x timer resolution is 55ms, => set wTimeOut_ms >
55
==========================================================================
==*/
int  SendDDCC(WORD wChannel, LPDDCCP lpDDCCP, WORD wTimeOut_ms);
/*==========================================================================
====
 Function:      SendDDCCEx()

 Description:   writes given values to DDCC of selected channel
                alike SendDDCC with structure opened

 Parameters:    see below

 Returns:       see SendDDCC()

 Notes:         see SendDDCC()
==========================================================================
==*/
```

```
int  SendDDCCEx(WORD wChannel,
                        BYTE bNAB,      /* tx/rx node address byte*/
                        BYTE bMIFR,     /* minor function code*/
                        BYTE bMAFR,     /* major function code*/
                        BYTE bDummy,
                        BYTE bData0,    /* tx/rx data*/
                        BYTE bData1,    /* tx/rx data*/
                        BYTE bData2,    /* tx/rx data*/
                        BYTE bData3,    /* tx/rx data*/
                        BYTE bData4,    /* tx/rx data*/
                        BYTE bData5,    /* tx/rx data*/
                        WORD wTimeOut_ms);
/*=========================================================================
====
 Function:      ReceiveDDCC()

 Description:   reads values from DDCC of selected channel into buffer
                pointed to by lpDDCCP if arrived
 Parameters:
        wChannel        channel number, must be valid
        lpDDCCP         points to structure where received values are
read to
        wTimeOut_ms     timeout in milliseconds

 Returns:       zero if successful, ERR_DWC__NOREPLY if no reply arrived

 Notes:         1. caller is responsible of allocating buffer pointed to
by lpDDCCP
                size must be at least sizeof(DDCCP)
                2. Waits for reply maximum time specified in wTimeOut_ms
parameter.
                If no reply arrived during that time, returns
ERR_DWC__NOREPLY
                3. win 3.x timer resolution is 55ms, => set wTimeOut_ms >
55
=========================================================================
==*/
int  ReceiveDDCC(WORD wChannel, LPDDCCP lpDDCCP, WORD wTimeOut_ms);
/*=========================================================================
====
 Function:      ReceiveDDCCEx()

 Description:   writes given values to DDCC of selected channel
                alike ReceiveDDCC with structure opened

 Parameters:    see below

 Returns:       see ReceiveDDCC()

 Notes:         see ReceiveDDCC()
=========================================================================
==*/
int  ReceiveDDCCEx(WORD wChannel,
                        BYTE  *lpbNAB,      /* tx/rx node address byte*/
                        BYTE  *lpbMIFR,     /* minor function code*/
                        BYTE  *lpbMAFR,     /* major function code*/
                        BYTE  *lpbDummy,
                        BYTE  *lpbData0,    /* tx/rx data*/
                        BYTE  *lpbData1,    /* tx/rx data*/
                        BYTE  *lpbData2,    /* tx/rx data*/
                        BYTE  *lpbData3,    /* tx/rx data*/
```

```
                        BYTE  *lpbData4,    /* tx/rx data*/
                        BYTE  *lpbData5,    /* tx/rx data*/
                        WORD wTimeOut_ms);

/*=====================================================================
==== ---------lp------
 Function:      ReadDDCCStatusRegister()

 Description:   reads status register DDCC of selected channel

 Parameters:
        wChannel channel number, must be valid

 Returns:       value of DDCC status register
 Notes:         see DDCC specifications for values
=====================================================================
==*/
BYTE  ReadDDCCStatusRegister(WORD wChannel);

/*=====================================================================
====
 Function:      WriteDDCCNodeAddressRegister()

 Description:   writes node address register of DDCC of selected channel

 Parameters:
        wChannel channel number, must be valid
        bOwnNode node number to write

 Returns:       value of DDCC node address register, read after writing
 Notes:         see DDCC specifications for values
=====================================================================
==*/
BYTE  WriteDDCCNodeAddressRegister(WORD wChannel, BYTE bOwnNode);

/*=====================================================================
====
 Function:      WriteDDCCControlRegister0()

 Description:   writes control register 0 of DDCC of selected channel

 Parameters:
        wChannel channel number, must be valid
        bValue   value to write

 Returns:       value of DDCC control register 0, read after writing
 Notes:         see DDCC specifications for values
=====================================================================
==*/
BYTE  WriteDDCCControlRegister0(WORD wChannel, BYTE bValue);

/*=====================================================================
====
 Function:      WriteDDCCControlRegister1()

 Description:   writes control register 1 of DDCC of selected channel

 Parameters:
        wChannel    channel number, must be valid
        bValue      value to write
```

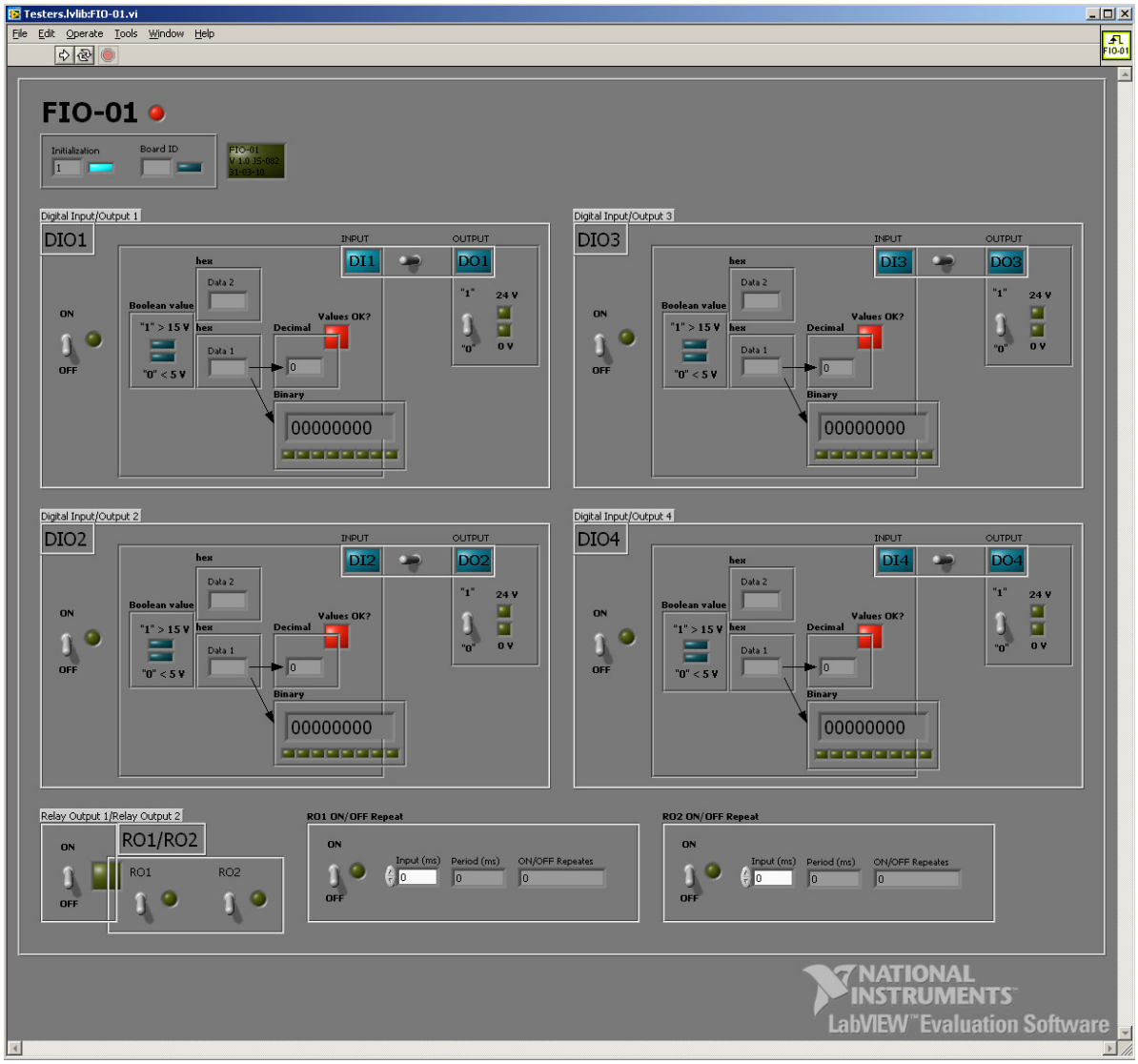
```
 Returns:       value of DDCC control register 1, read after writing
 Notes:         see DDCC specifications for values
================================================================================
==*/
BYTE  WriteDDCCControlRegister1(WORD wChannel, BYTE bValue);

/*================================================================================
====
 Function:      ReadDDCCConfiguration()

 Description:   reads node address register and control registers
                0 and 1 of DDCC of selected channel into given buffer
 Parameters:
        wChannel    channel number, must be valid
        lpCnfRegs   points to buffer to receive the read values

 Returns:       zero if successfull, otherwise negative error code

 Notes:         see DDCC specifications
                caller is responsible of allocating buffer pointed to by
lpCnfRegs,
                size must be at least sizeof(LPDDCC_CTRL)
================================================================================
==*/
int  ReadDDCCConfiguration(WORD wChannel, LPDDCC_CTRL lpCnfRegs);

/*================================================================================
====
 Function:      ShareDDCPLibrary()

 Description:   allows another program to use the communication library
                simultaneously
 Parameters:

 Returns:

 Notes:         NISA-03 and NDPA-02 channel should not be used by the
                program that allows sharing
                only those channels closed by the program are available
                for other programs
                                              see also CloseChannel()
                the function is an enhancement implemented in version
3.23,
                which also implements RUSB-02
/================================================================================
===*/
void  ShareDDCPLibrary(void);

/*================================================================================
====
 Function:      GetChannelType()

 Description:   returns hardware type

 Parameters:
        wChannel    channel number, must be valid

 Returns:             hardware type of the channel
                see definition DDCC / Hardware

 Notes:         use this function to see if RUSB-02 functionality is
```

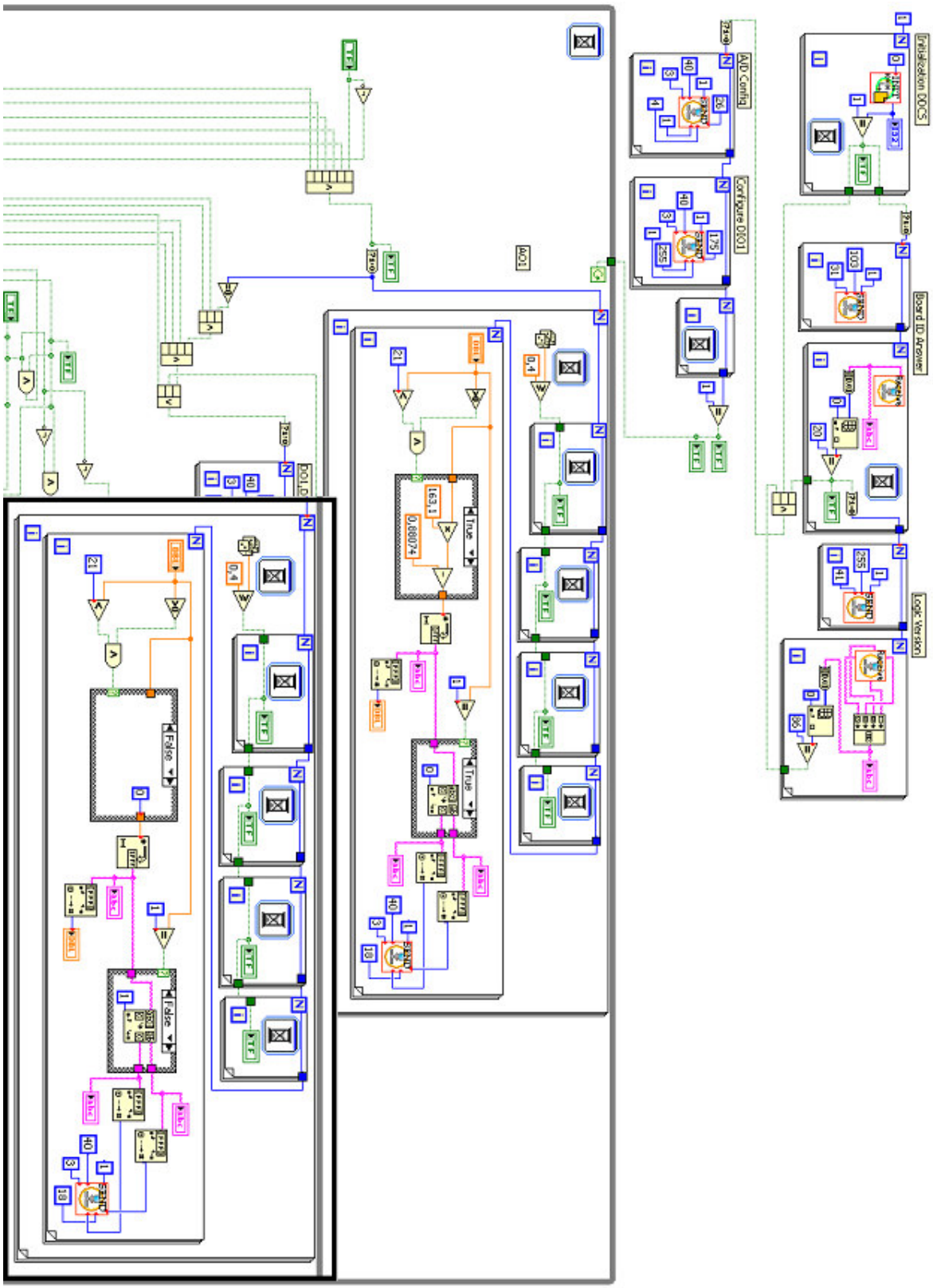
```
                available
                the function is an enhancement implemented in version
3.23,
                which also implements RUSB-02
/=========================================================================
===*/
WORD  GetChannelType(WORD wChannel);

/*=========================================================================
====
 Function:      GetChannelDDCCBase()

 Description:   returns DDCC base address or RUSB channel index

 Parameters:
        wChannel    channel number, must be valid

 Returns:             DDCC base address or RUSBChannelIndex
                depending on hardware type (see GetChannelType())

 Notes:         use this function to see get RUSB-02 channel index
                for RUSB specific functions (defined elsewhere)
                the function is an enhancement implemented in version
3.23,
                which also implements RUSB-02
/=========================================================================
===*/
WORD  GetChannelDDCCBase(WORD wChannel);

/*=========================================================================
====
 Function:      CloseChannel()

 Description:   closes a RUSB-02 channel

 Parameters:
        wChannel    channel number, must be valid

 Returns:

 Notes:         has no effect on NISA-03 or NDPA-02 channels
                the channel to be closed must not be used before or after
closing
                unless all other channels of the RUSB device has already
been closed
                of course, this limitation does not concern devices with
only one
                channel (such as current RUSB-02)
                use this function in case you use only some RUSB-02 chan-
nels and
                want another program to be able to use the channles you
do not
                need
                                            see also ShareDDCPLibrary()
                the function is an enhancement implemented in version
3.23,
                which also implements RUSB-02
/=========================================================================
===*/
void  CloseChannel(WORD wChannel);
```

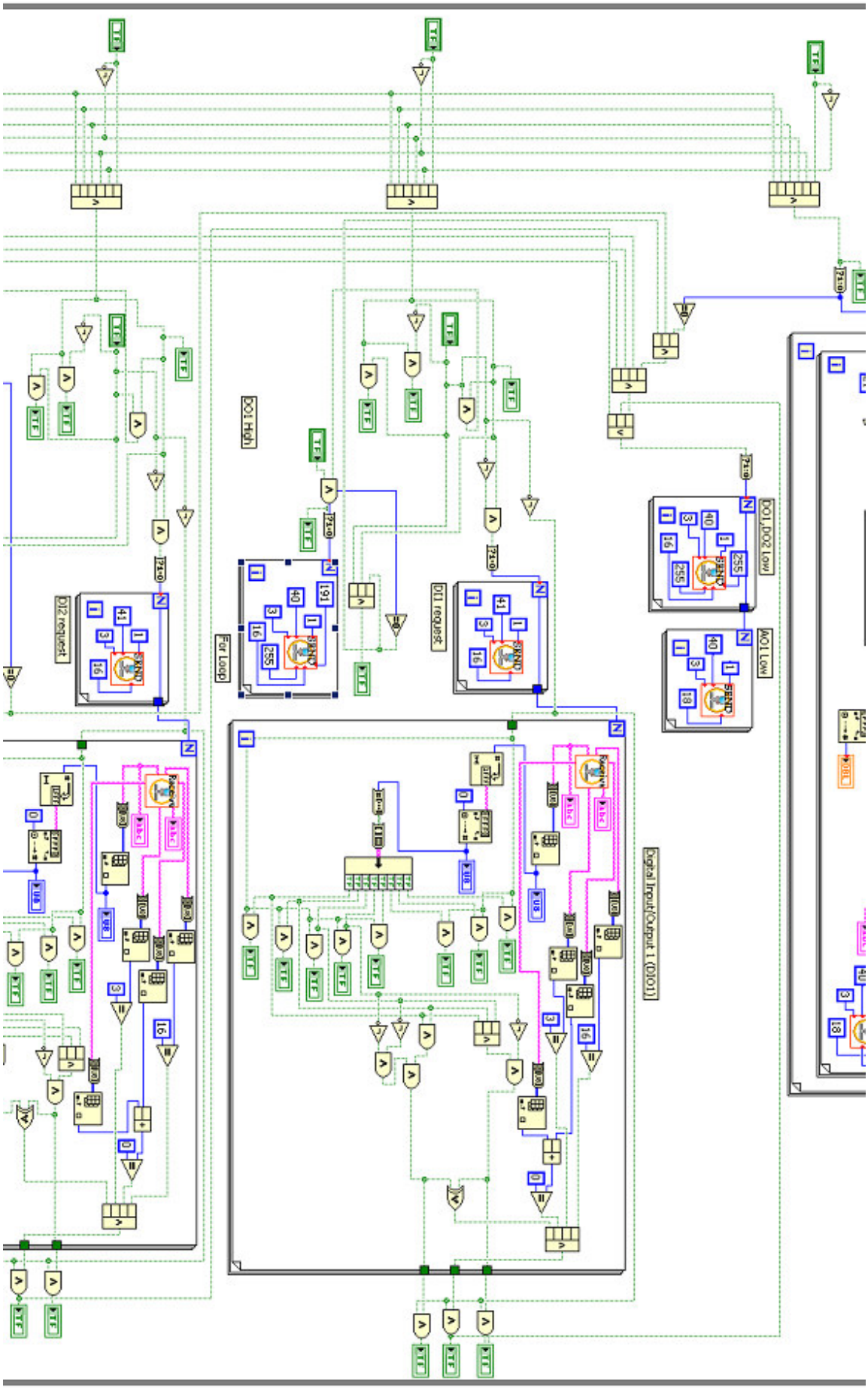
```
/*=====================================================================
====
 Function:      GetTimeStamps()

 Description:   returns latest time-stamp information about a RUSB-02
channel

 Parameters:
        wChannel    channel number, must be valid
        pPerformanceStampAtWrite
                    the performance counter value saved at last write
that
                                                        reatarts the RUSB-
02 time-stamp counter is returned here
                                                        unit can be re-
quested by calling QueryPerformanceFrequency()
                                                        (see Windows API)
        pReadDataTimeStamp
                    the RUSB-02 time-stamp value saved at last read is
                    is returned here
                                                        unit is 100 ns
(same as FILETIME)

 Returns:             TRUE if successful, FALSE otherwise

 Notes:         returns FALSE on NISA-03 or NDPA-02 channels
                also returns FALSE in case time-stamping of the channel
is
                                                disabled or not possible by
the computer hardware
                                                (see EnableTimeStamps())
                                                the time-stamp of the latest
read can be calculated by
                                                converting performance
counter value to FILETIME (requires
                                                synchronizing information
about performance counter and
                                                real time) and adding the Re-
adDataTimeStamp to it
                                                the function is mainly used
by DriveOPC to improve time-
                                                stamp accuracy of RUSB-02
channels
                the function is an enhancement implemented in version
3.23,
                which also implements RUSB-02
/=====================================================================
===*/
BOOL  GetTimeStamps(WORD wChannel, LARGE_INTEGER* pPerformanceStampAt-
Write,

          __int64* pReadDataTimeStamp);

/*=====================================================================
====
 Function:      EnableTimeStamps()

 Description:   enables or disable time-stamping on a RUSB-02 channel

 Parameters:
        wChannel    channel number, must be valid
```

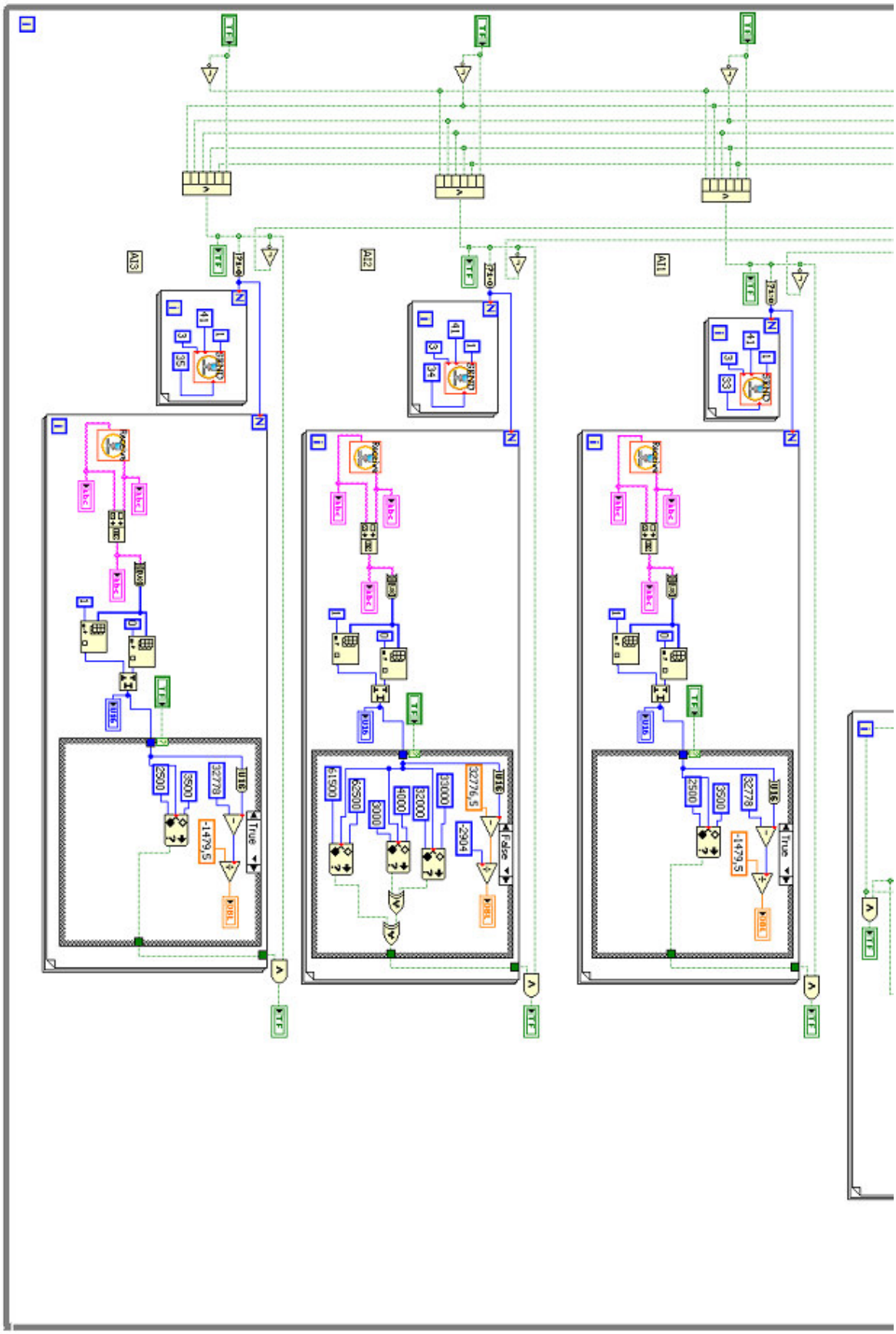
```
        bEnable     TRUE enables and FALSE disables time-stamping

 Returns:       previous value of time-stamp enable/disable

 Notes:         returns FALSE on NISA-03 or NDPA-02 channels
                time-stamping is disabled by default, so, if you
                                            want to use it, please enable
it
                                            enabling time-stamps means
heavier processor load
                                            in RUSB-02 handling
                                            see also GetTimeStamps()
                                            the function is mainly used
by DriveOPC to improve time-
                                            stamp accuracy of RUSB-02
channels
                the function is an enhancement implemented in version
3.23,
                which also implements RUSB-02
/=============================================================================
===*/
BOOL  EnableTimeStamps(WORD wChannel, BOOL bEnable);

/*=============================================================================
======

    TYPE DEFINITIONS OF INTERFACE FUNCTIONS:

=============================================================================
====*/
/* library version check*/
typedef WORD (DWC_APIPFN LPFNGetMajorVersionOfDDCPLib)(void);
typedef WORD (DWC_APIPFN LPFNGetMinorVersionOfDDCPLib)(void);

/* library initialization and cleanup*/
typedef int (DWC_APIPFN LPFNInitDDCPLib)(WORD , LPSTR );
typedef void (DWC_APIPFN LPFNCleanupDDCPLib)(void);

/*data transmit*/
typedef int (DWC_APIPFN LPFNSendDDCC)(WORD , LPDDCCP , WORD );
typedef int (DWC_APIPFN LPFNReceiveDDCC)(WORD , LPDDCCP , WORD );

/*special low-level DDCC access*/
/*not necessary to use if configuring by InitDDCPLib() satisfies*/
typedef int (DWC_APIPFN LPFNReadDDCC_ISAControl)(WORD );
typedef int (DWC_APIPFN LPFNWriteDDCC_ISAControl)(WORD , BYTE );
typedef BYTE (DWC_APIPFN LPFNReadDDCCStatusRegister)(WORD );

typedef BYTE (DWC_APIPFN LPFNWriteDDCCNodeAddressRegister)(WORD , BYTE );
typedef BYTE (DWC_APIPFN LPFNWriteDDCCControlRegister0)(WORD , BYTE );
typedef BYTE (DWC_APIPFN LPFNWriteDDCCControlRegister1)(WORD , BYTE );
typedef int (DWC_APIPFN LPFNReadDDCCConfiguration)(WORD , LPDDCC_CTRL );

/* (RUSB-02) enhancements in version 3.23 */
typedef void (DWC_APIPFN LPFNShareDDCPLibrary)();
typedef WORD (DWC_APIPFN LPFNGetChannelType)(WORD );
typedef void (DWC_APIPFN LPFNCloseChannel)(WORD );
typedef BOOL (DWC_APIPFN LPFNGetTimeStamps)(WORD , LARGE_INTEGER*,
__int64* );
typedef BOOL (DWC_APIPFN LPFNEnableTimeStamps)(WORD, BOOL );
```

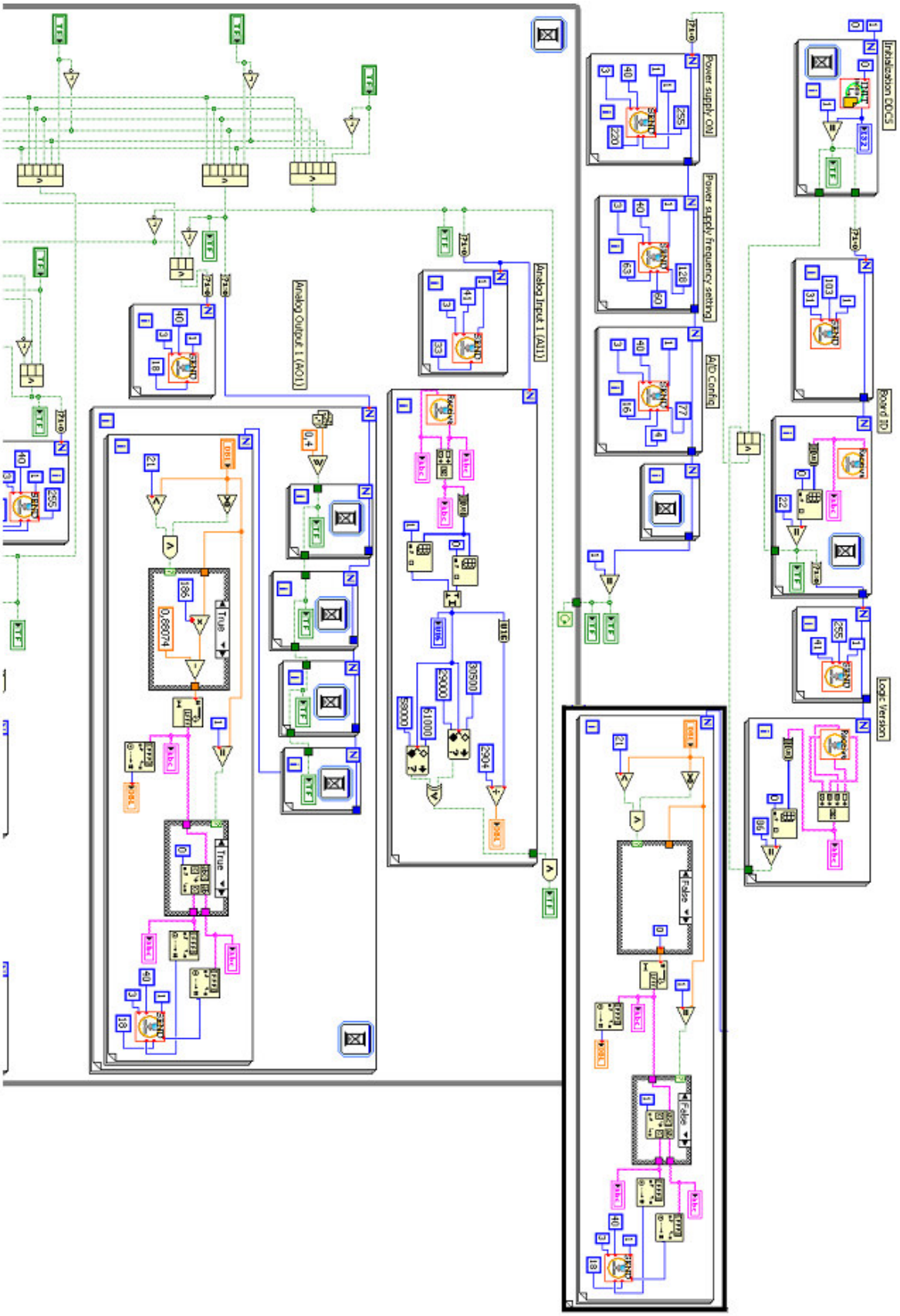
```
#else /*LABWINDOWS_RULES    prototypes for labwindows follow here:*/
/*
int -> short
large model expected
see descriptions of functions above*/
#define FAR     /*large model expected*/
short DWC_API GetMajorVersionOfDDCPLib(void);
short DWC_API GetMinorVersionOfDDCPLib(void);
short DWC_API InitDDCPLib(WORD wShowDialog, LPSTR lpszIniFilename);
void  DWC_API CleanupDDCPLib(void);
short DWC_API ReadDDCC_ISAControl(WORD wChannel);
short DWC_API WriteDDCC_ISAControl(WORD wChannel, BYTE bControl);
short DWC_API SendDDCC(WORD wChannel, LPDDCCP lpDDCCP, WORD wTimeOut_ms);
int  DWC_API SendDDCCEx(WORD wChannel,
                        BYTE bNAB,      /* tx/rx node address byte*/
                        BYTE bMIFR,     /* minor function code*/
                        BYTE bMAFR,     /* major function code*/
                        BYTE bDummy,
                        BYTE bData0,    /* tx/rx data*/
                        BYTE bData1,    /* tx/rx data*/
                        BYTE bData2,    /* tx/rx data*/
                        BYTE bData3,    /* tx/rx data*/
                        BYTE bData4,    /* tx/rx data*/
                        BYTE bData5,    /* tx/rx data*/
                        WORD wTimeOut_ms);
short DWC_API ReceiveDDCC(WORD wChannel, LPDDCCP lpDDCCP, WORD wTime-
Out_ms);
int  DWC_API ReceiveDDCCEx(WORD wChannel,
                        BYTE FAR *lpbNAB,      /* tx/rx node address
byte*/
                        BYTE FAR *lpbMIFR,     /* minor function code*/
                        BYTE FAR *lpbMAFR,     /* major function code*/
                        BYTE FAR *lpbDummy,
                        BYTE FAR *lpbData0,    /* tx/rx data*/
                        BYTE FAR *lpbData1,    /* tx/rx data*/
                        BYTE FAR *lpbData2,    /* tx/rx data*/
                        BYTE FAR *lpbData3,    /* tx/rx data*/
                        BYTE FAR *lpbData4,    /* tx/rx data*/
                        BYTE FAR *lpbData5,    /* tx/rx data*/
                        WORD wTimeOut_ms);
BYTE  DWC_API ReadDDCCStatusRegister(WORD wChannel);
BYTE  DWC_API WriteDDCCNodeAddressRegister(WORD wChannel, BYTE bOwnNode);
BYTE  DWC_API WriteDDCCControlRegister0(WORD wChannel, BYTE bValue);
BYTE  DWC_API WriteDDCCControlRegister1(WORD wChannel, BYTE bValue);
short DWC_API ReadDDCCConfiguration(WORD wChannel, LPDDCC_CTRL
lpCnfRegs);

#endif /*LABWINDOWS_RULES*/
/*=========================================================================
====

    POST-DEFINITIONS

=========================================================================
==*/


#endif /* !_INC_DDCP_LIB_H*/

/* EOF*/
```

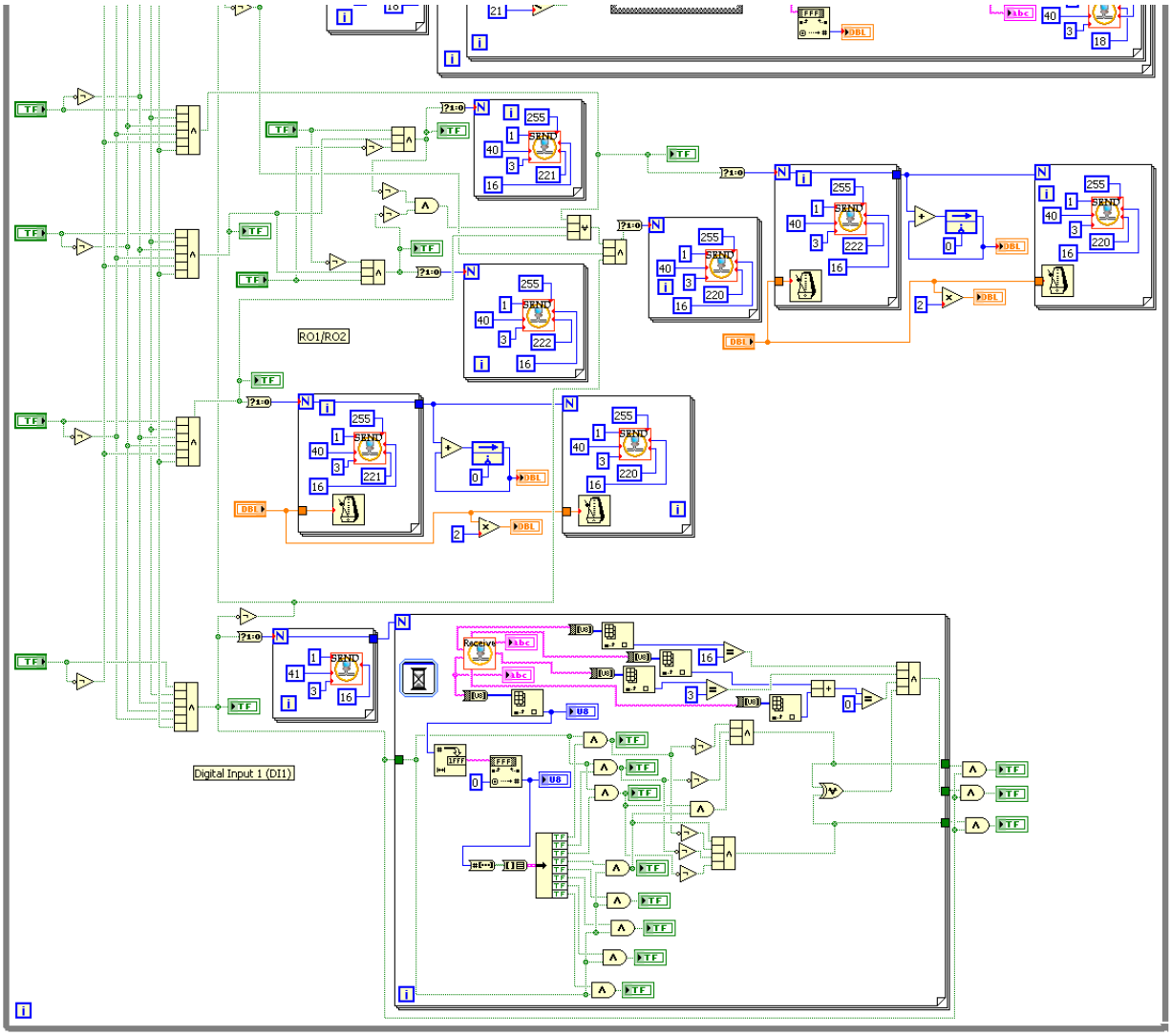

Testers.lvlib:FIO-01.vi
File Edit Operate Tools Window Help
FIO-01
Initialization
Board ID
Digital Input/Output 1
DIO1
INPUT
OUTPUT
DI1
DO1
hex
Data 2
Boolean value
"1" > 15 V
"0" < 5 V
hex
Data 1
Decimal
Values OK?
Binary
00000000
ON
OFF
"1"
"0"
24 V
0 V
Digital Input/Output 3
DIO3
DI3
DO3
Digital Input/Output 2
DIO2
DI2
DO2
Digital Input/Output 4
DIO4
DI4
DO4
Relay Output 1/Relay Output 2
RO1/RO2
RO1
RO2
RO1 ON/OFF Repeat
RO2 ON/OFF Repeat
Input (ms)
Period (ms)
ON/OFF Repeates
NATIONAL INSTRUMENTS
LabVIEW™ Evaluation Software

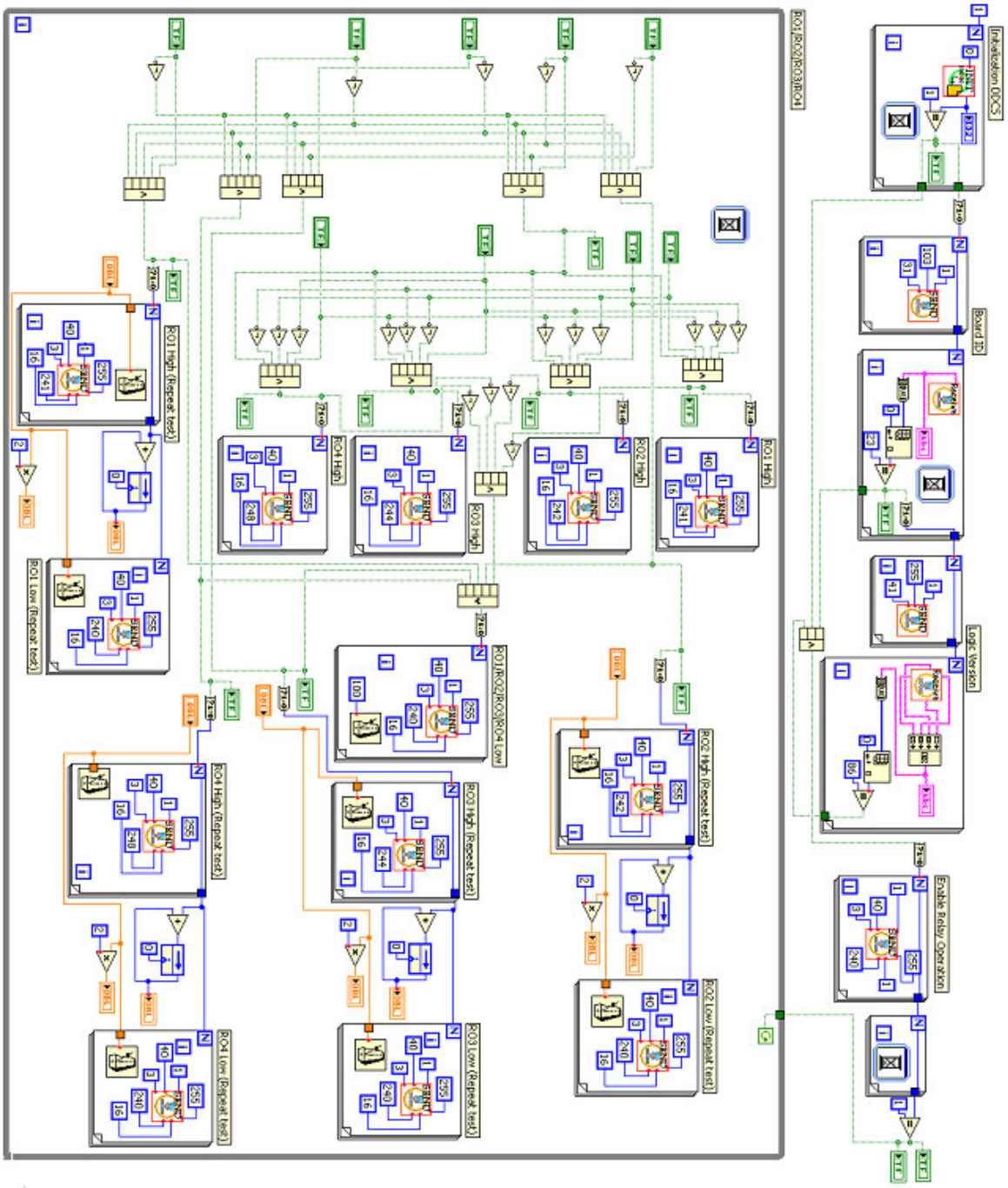

Initialization OOCS
AUD Config
Configure 0601
Board ID Answer
Logic Version
163.1
0.680274
True
False

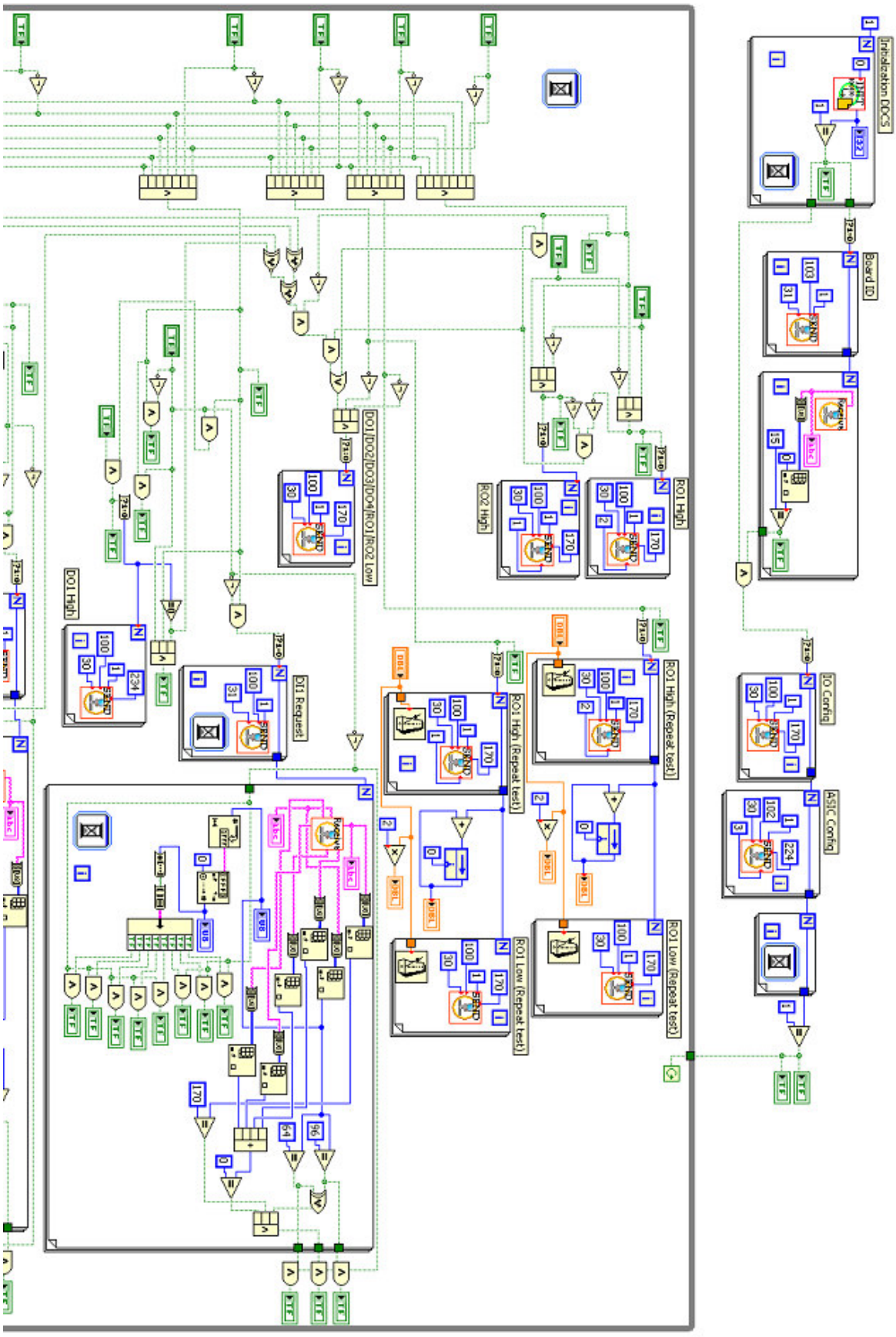

DO1 High
For Loop
DI2 request
DI1 request
DO1,DO2 Low
AO1 Low
Digital Input/Output 1 (DIO1)

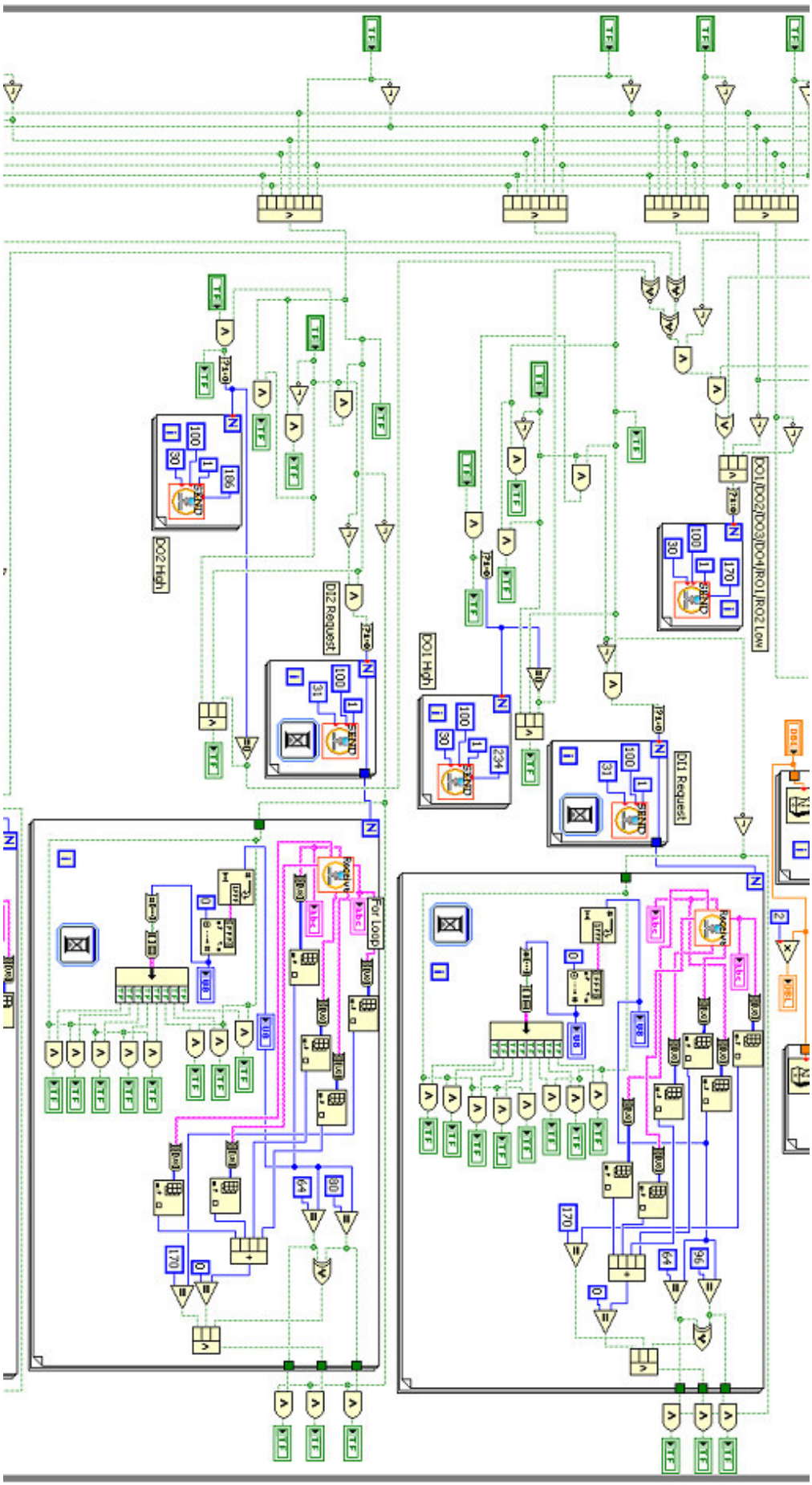

AI3
AI2
AI1
True
False

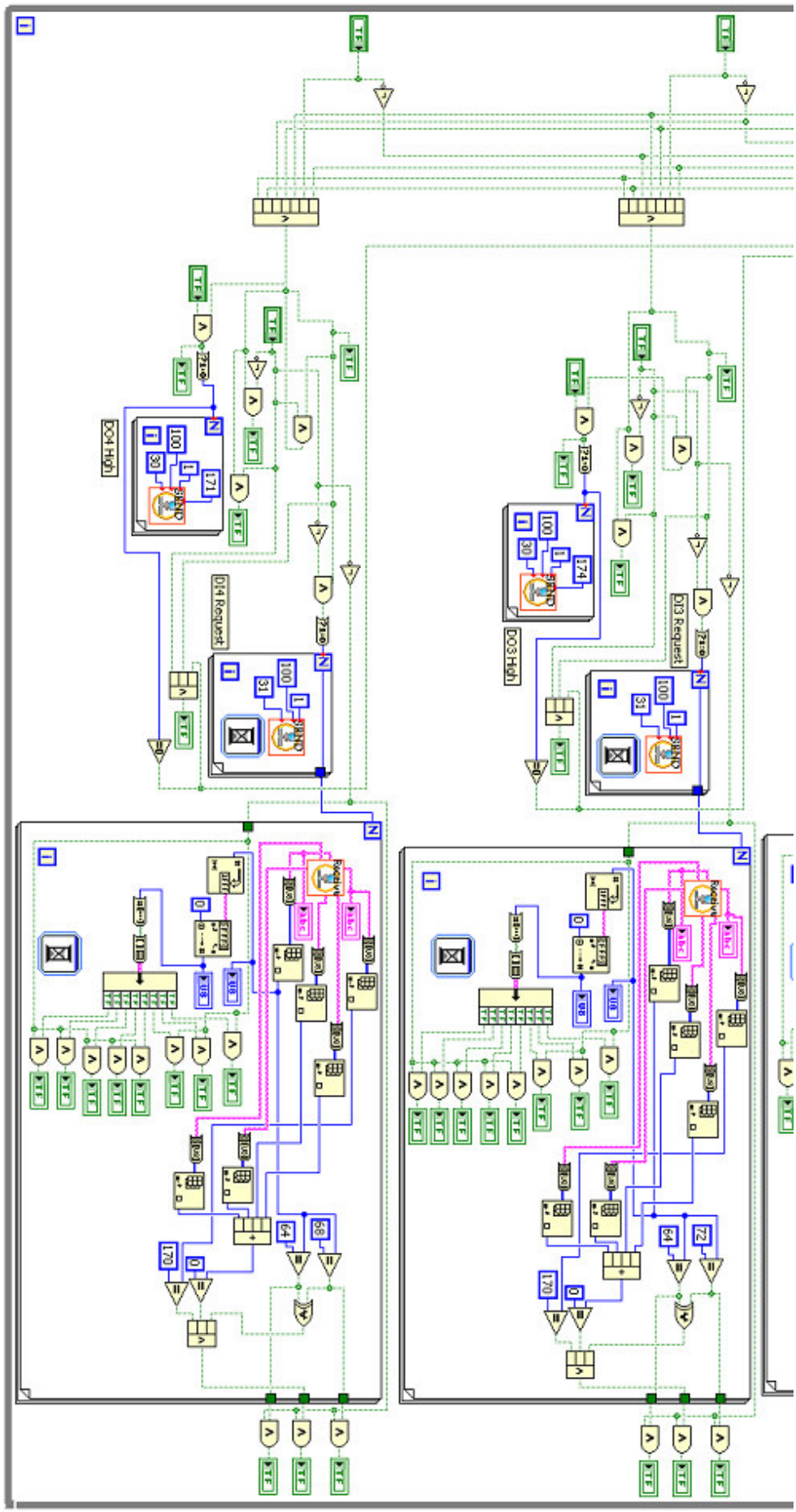

Initialization DOCS
Power supply ON
Power supply frequency setting
A/D Config
Board ID
Logic Version
Analog Output 1 (AO1)
Analog Input 1 (AI1)
True
False

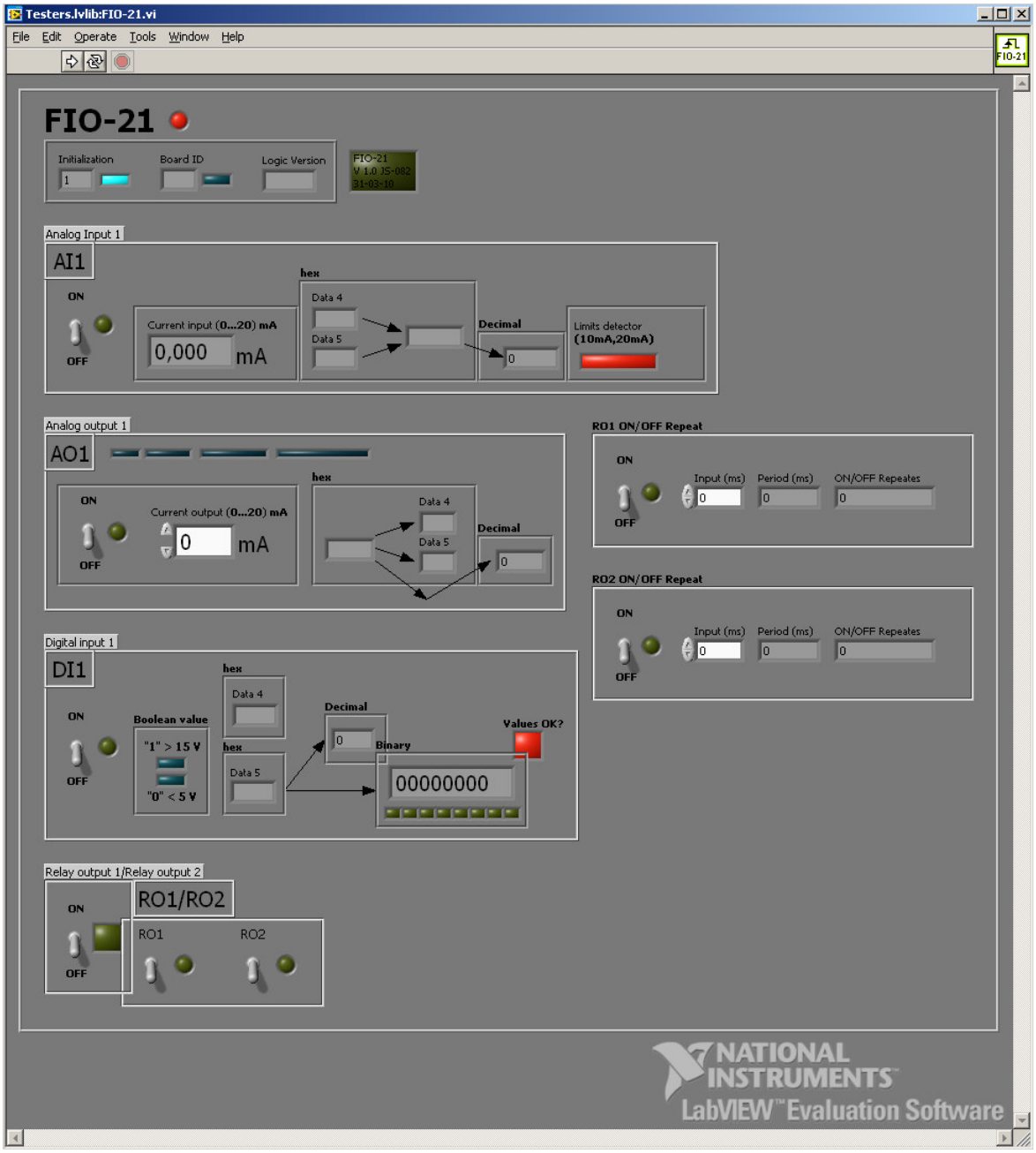

RO1/RO2
Digital Input 1 (DI1)


RO1 RO2 RO3 RO4
Initialization OOCS
Board ID
Logic Version
Enable Relay Operation
RO1 High (Repeat test)
RO1 Low (Repeat test)
RO1 High
RO2 High
RO3 High
RO1 RO2 RO3 RO4 Low
RO4 High (Repeat test)
RO3 High (Repeat test)
RO2 High (Repeat test)
RO4 Low (Repeat test)
RO3 Low (Repeat test)
RO2 Low (Repeat test)


Initialization DOCS
Board ID
IO Config
ASIC Config
DO1 High
RO1 High
RO2 High
DO1/DO2/DO3/DO4/RO1/RO2 Low
DI1 Request
RO1 High (Repeat test)
RO1 Low (Repeat test)


DO2 High
DI2 Request
DO1 High
DI1 Request
For Loop


DO4 High
DI4 Request
DO3 High
DI3 Request


Testers.lvlib:FIO-21.vi
File Edit Operate Tools Window Help
FIO-21
Initialization
Board ID
Logic Version
FIO-21
V 1.0 JS-082
31-03-10
Analog Input 1
AI1
ON
OFF
Current input (0...20) mA
0,000 mA
hex
Data 4
Data 5
Decimal
0
Limits detector
(10mA,20mA)
Analog output 1
AO1
ON
OFF
Current output (0...20) mA
0 mA
hex
Data 4
Data 5
Decimal
0
RO1 ON/OFF Repeat
ON
OFF
Input (ms)
Period (ms)
ON/OFF Repeates
0
RO2 ON/OFF Repeat
ON
OFF
Input (ms)
Period (ms)
ON/OFF Repeates
0
Digital input 1
DI1
ON
OFF
Boolean value
"1" > 15 V
"0" < 5 V
hex
Data 4
hex
Data 5
Decimal
0
Binary
00000000
Values OK?
Relay output 1/Relay output 2
RO1/RO2
ON
OFF
RO1
RO2
NATIONAL INSTRUMENTS
LabVIEW™ Evaluation Software

Moi

FIO-21:llä AO kytkentä on muuten sama mutta pwm:n referenssijännite on 3V3 kun se
FIO-11:sta on 2V5.

Mikan mailissa oli taannoin taulukkoa:

```
const iq16 AO2PWM_COEF[3] = {     IQ16(0.0),     IQ16(163.1),          IQ16(186.0) };
const iq16 AO2PWM_CONST[3] = { IQ16(0.0),     IQ16(0.88074-0.5),   IQ16(0.0) };
```

...josta vakiokerroin FIO-11:sta jota olit jo käyttänytkin:

*PWM = 163.1* I -0.88074*

Oisko tuo kerroin 186 olla sitten FIO-21:lle..eli

*PWM = 186.0* I -0.88074*

Analog in-kytkennät poikkeaa toisistaan, FIO-21:llä vain virtatulo

```
const iq15 AI_VOLT_COEF[3] =    { IQ15(0.0),     IQ15(-2.0/2904),     IQ15(-2.0/2904) };
const iq15 AI_VOLT_CONST[3] = { IQ15(0.0),     IQ15(32776.5),       IQ15(32776.5) };
const iq15 AI_CURR_COEF[3] =    { IQ15(0.0),     IQ15(-2.0/1479.5),   IQ15(2.0/2904) };
const iq15 AI_CURR_CONST[3] = { IQ15(0.0),     IQ15(32778.0),       IQ15(0.0) };
```

FIO-21:lle: ? I = CHx - ? / 2904

terveisin, Ari

---

Moikka,

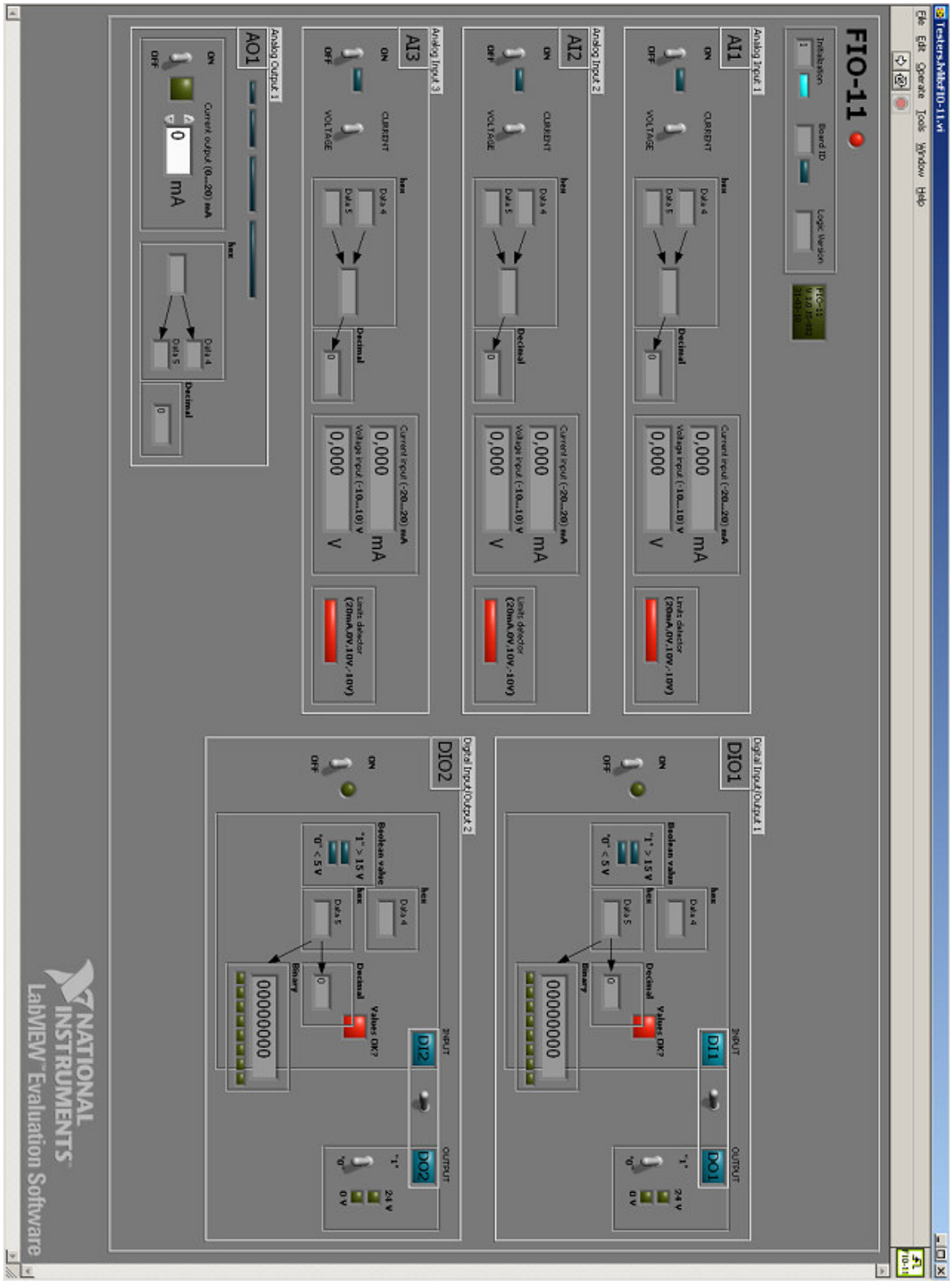
Olisiko sinulla jotain hyviä ideoita miten pystyisin skaalaamaan FIO-21?
Kun sain FIO-11 skaalattua kaavoilla jotka olivat siinä s-postissa minkä lähetit, mutta samat kaavat eivät päteneet FIO-21een. esim. analog input, analog output (virta tulo/lähtö).

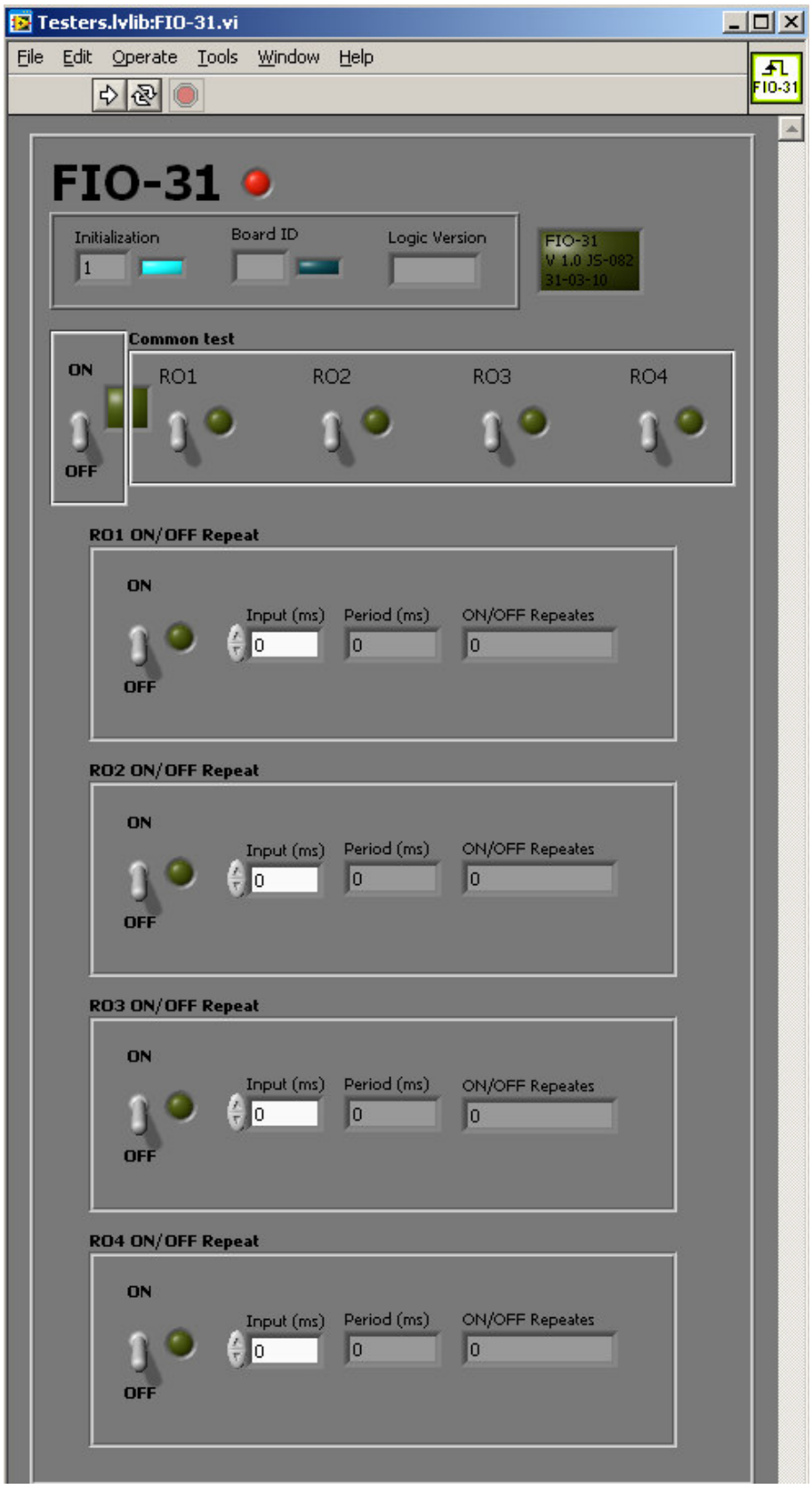
ohessa kaavat:

$$I = \frac{CHx - 32778}{-1479.5} \qquad (3b)$$

*PWM = 163.1* I -0.88074*

t.Jarmo


testers/kitit/FIO-11.vi
File Edit Operate Tools Window Help
FIO-11
Initialization
Board ID
Logic Version
Analog Input 1
AI1
ON
OFF
CURRENT
VOLTAGE
bus
Data 4
Data 5
Decimal
Current input (-20...20) mA
0,000
mA
Voltage input (-10...10) V
0,000
V
Limits detector (20mA,0V,10V,-10V)
Analog Input 2
AI2
Analog Input 3
AI3
Analog Output 1
AO1
Current output (0...20) mA
0
mA
Digital Input/Output 1
DIO1
INPUT
OUTPUT
DI1
DO1
Boolean value
"1" > 15 V
"0" < 5 V
Values OK?
Binary
00000000
"1"
"0"
24 V
0 V
Digital Input/Output 2
DIO2
DI2
DO2
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software


Testers.lvlib:FIO-31.vi
File Edit Operate Tools Window Help
FIO-31
FIO-31
Initialization
1
Board ID
Logic Version
FIO-31
V 1.0 JS-082
31-03-10
Common test
ON
OFF
RO1
RO2
RO3
RO4
RO1 ON/OFF Repeat
ON
OFF
Input (ms)
Period (ms)
ON/OFF Repeates
0
RO2 ON/OFF Repeat
ON
OFF
Input (ms)
Period (ms)
ON/OFF Repeates
0
RO3 ON/OFF Repeat
ON
OFF
Input (ms)
Period (ms)
ON/OFF Repeates
0
RO4 ON/OFF Repeat
ON
OFF
Input (ms)
Period (ms)
ON/OFF Repeates
0