SULAUTETUN JÄRJESTELMÄN PUOLIAUTOMAATTINEN TESTAUS

Teemu Karimerto

Thuy Nguyen

Opinnäytetyö

Joulukuu 2011

Tietotekniikan koulutusohjelma

Sulautettujen järjestelmien

suuntautumisvaihtoehto

Tampereen ammattikorkeakoulu

1 TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautettujen järjestelmien koulutusohjelma

KARIMERTO, TEEMU & NGUYEN, THUY: Sulautetun järjestelmän puoliautomaattinen testaus

Opinnäytetyö 73 s., liitteet 61 s.
Joulukuu 2011

Tämän työn tarkoituksena oli esitellä yksi tapa sekä kirjoittaa kokonainen ohjelmistokomponenttiketju aina rautatasolta ylimmälle C-kielisille tasolle että esitellä järjestelmällinen tapa testata ja simuloida kirjoitetut koodit. Työssä esiteltiin myös itse kehitetty simulaatioympäristö VHDL-kielisille komponenteille.

Uuden laitteiston kehittäminen on haastavaa työtä ja ilman testausta siitä ei tulisi mitään. Lopputulos ei myöskään olisi luotettava eikä missään tapauksessa markkinakelpoinen. Sekä testeri että VHDL-simulaatioympäristö ovat aktiivisessa käytössä erään uuden varastojärjeltelmän kehityksessä.

Testaus- ja simulaatioympäristö ovat edelleen kehityksen alla eivätkä ne ole lopullisia tuotteita. Ne ovat silti osoittautuneet täysin korvaamattomiksi työkaluiksi.

Asiasanat: testaus, nios, vhdl, simulointi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Computer Science
Option of Embedded Systems

KARIMERTO, TEEMU & NGUYEN, THUY: Semi-automatic Testing of an Embedded System

Bachelor's thesis 73 pages, appendices 61 pages
December 2011

The purpose of this thesis is to demonstrate a systematic, accurate and user-assisted testing of several different circuit boards. This thesis is divided into two separate, but mutually supportive sections. The first part introduces and describes a detailed case study of a single test component and testing architecture as well as thorough simulation testing using Python programming language. The second part deals with test modules in a more generic manner and also describes the advanced testing architecture.

The development of any new hardware/software project is always a challenge, and without proper testing and testing tools, would be a disaster waiting to happen. In response to this challenge, a solution was devised. This solution includes a simulation environment for VHDL modules and a tester application/environment for entire printed circuit boards.

Both the tester and the simulation environment have been in active use for only a short while, so improvement are still forthcoming. They have however proven themselves to be invaluable tools in the development process.

Keywords: testing, nios, vhdl, simulation

2 SISÄLLYS

3 TESTAUSJÄRJESTELMÄN YLEISKUVAUS

Testaus on aina ollut, ja tulee aina olemaan, olennainen osa mitä tahansa uuden laitteen kehitysprosessia. Testaus koskee niin laitteistoa kuin ohjelmistoakin. Tämän työn tarkoitus on esitellä erään kehitteillä olevan uuden laitejärjestelmän piirilevyjen ja eri ohjelmistokomponenttien testaus aina alimmalta tasolta ylöspäin. Samantyyppistä testausjärjestelmää voidaan käyttää lähes missä tahansa sulautetun järjestelmän testauksessa jonka kehitys tehdään Alteran Quartus-ympäristössä. Testausohjelmisto on kehitetty ainoastaan Linux-ympäristöön sen ilmaisten työkalujen ja ohjelmistojen sekä tiedostojärjestelmän vuoksi.

Testattava laitteisto koostuu neljästä erilaisesta, mutta pääkomponenteiltaan samankaltaisesta piirilevystä. Jokaiselta levyltä löytyy DDR-muisti, MSP430-piiriin perustuva bootloader, SD-kortin lukija, eri määrä moottoriohjaimia, AMR-sensoreita ja AD-muuntimia. Näiden lisäksi osassa piirilevyistä on vielä vaihtelevasti radio, viiva-antureita, etäisyysantureita, akkujen latauspiiri, valoverhot sekä kallistusanturi. Testausjärjestely on jokaiselle komponentille pohjimmiltaan samankaltainen. Alimmalla tasolla simuloidaan ja testataan VHDL-kirjasto (tai useampi), seuraavalla tasolla VHDL-kirjastoja ohjaavat C-kieliset kirjastot ja ylimpänä C-kielinen testikirjasto.

Suurin osa C-kielisistä kirjastoista on suunniteltu käännettäviksi sekä sulautetulle prosessorille että osittain emuloidusti PC:lle. Tästä syystä C-kieliset kirjastot voidaan pääosin testata suoraan PC:llä ilman erillistä simulointiympäristöä. VHDL-kirjastot puolestaan simuloidaan ja testataan niitä varten erikseen Python-kielellä kehitetyllä simulaatio-ohjelmistolla. Simulaattorin ja C-kieliset testikirjastot on pääosin kehittänyt Teemu Karimerto yhteistyössä Harry Flinkin kanssa. Varsinaiset simulaatiotestit on kehittänyt Thuy Nguyen.

4 KOKONAISEN KOMPONENTTIKETJUN KUVAUS

Koko testausjärjestelmä elektroniikasta ylöspäin on varsin laaja kokonaisuus, joten parhaiten siihen pääsee käsiksi kuvaamalla yhden kokonaisen komponenttiketjun aina ohjauselektroniikasta ylimmälle C-kieliselle tasolle. Esiteltäväksi komponentiksi valittiin akkujen langattomasta latauksesta huolehtiva *batterycharger*. Tämän komponentin tehtävä on kaksiosainen. Normaalissa käytössä se on vastuussa yksinkertaisesti akkujen latauksesta langattomasti muuntajan välityksellä. Pyyhkäisytilassa se mittaa parhaan taajuuden tehonsiirron kannalta.

4.1 Ohjauselektroniikka

4.1.1 Yleistä

Laitteen liikkuvan sensorilavan erillisellä piirilevyllä olevat akut tarvitsevat latausta, joka tapahtuu langattomasti. Lataus tapahtuu muuntajan välityksellä langattomasti. Itse latauspiiriä ohjaa kaksi erillistä ohjelmakirjastoa: *batterycharger* ja tätä ohjaava osa *batterychargercontrol*. Nämä kirjastot on toteutettu käyttäen laitteistonläheistä VHDL-ohjelmointikieltä. *Batterycharger*-kirjasto sisältää viisi tiedostoa, jotka ovat *batterycharger_filter.vhd*, *batterycharger_logic.vhd*, *batterycharger_avalon.vhd*, *batterycharger.vhd* ja *batterycharger_hw.tcl*. Näitä tarkastellaan yksitellen myöhemmin. Yksinkertaisempi *batterychargercontrol* sisältää vain yhden VHDL-tiedoston, *batterychargercontrol.vhd*. Ohjelmakirjastolla tarkoitetaan kaikkia yhteen loogiseen kokonaisuuteen eli toiminnalliseen yksikköön kuuluvia tiedostoja.

Ohjelmakirjastojen toimivuuden testaamiseksi on tehty erillinen simulointiympäristö. Suurin osa simulointiympäristöstä on toteutettu Python-ohjelmointikielellä, ja lopusta

huolehtii GHDL-simulaatio-ohjelmisto. Kaikki *batterycharger*-kirjaston tiedostot on testattu ensin yksitellen erillisillä testiohjelmilla, jonka jälkeen kokonaisuus testatiin vielä erikseen. *Batterychargercontrol*-kirjaston yksinkertaisuuden vuoksi on sen testikin varsin pieni.

4.1.2 Toiminta

Akkujen lataus piirilevyllä tapahtuu langattomasti. Akkujen lautauksen hoitaa *batterycharger*-kirjasto ja tätä ohjaava yksinkertaisempi kirjasto, *batterychargercontrol*. Laitteen rungossa on kaksi AMR-sensoria (anisotrooppinen magnetoresistiivinen sensori), jotka tunnistavat kun laitteen lapa on oikeassa asennossa akkujen lautausta varten. Lavan ollessa oikeassa asennossa AMR-sensorit välittävät tiedon *batterychargercontrol*-kirjastolle, joka sallii tai estää, *batterycharger*:n ladata akkuja. Normaalikäytössä akkujen lataus on mahdollista ainoastaan lavan ollessa oikeassa asennossa. Akkujen laturi voidaan testausta varten pakottaa päälle AMR-sensoreista huolimatta.

4.1.3 Muuntajan ohjauselektroniikka

Tässä osiossa tarkastellaan laitteen akkujen laturin ohjauselektroniikkaa, joka koostuu muuntajan ensiöpuolen ajurista (kuvio 1 sivulla 10 ja kuvio 2 sivulla 11) ja toisiopuolen jännitteensyöttöpiiristä (kuvio 3 sivulla 12). Ohjauselektroniikan on suunnitellut Tero Kultanen.

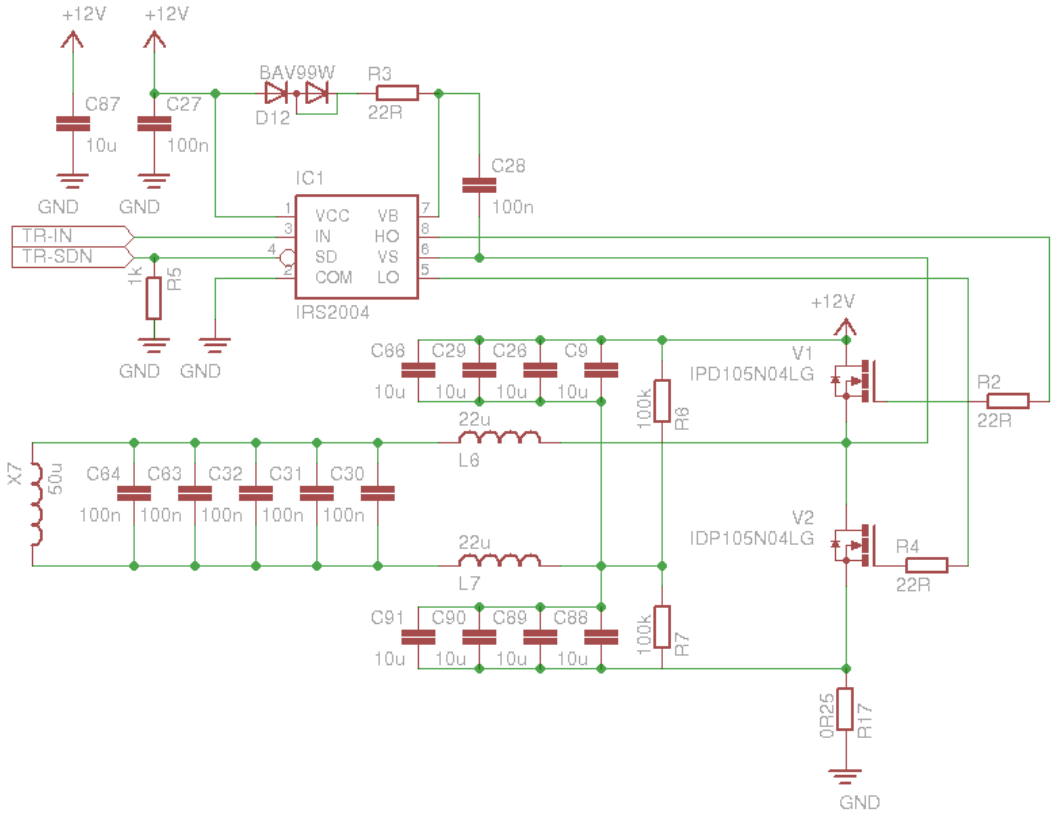
4.1.4 Ensiöpuolen ajuripiiri IRS2004

Kuviossa 1 muuntajan ensiöpuolen ajuripiirissä on käytetty International Rectifierin valmistama IRS2004-puolisiltaohjainta ohjaamaan jännitettä. Ohjainpiirin ulostulo ohjaa kahta IPD105-MOSFET:ia. Ulostulo riippuu kahdesta piiriin sisäänmenevistä signaaleista: `TR-IN` ja `TR-SDn`. Mikäli `TR-SDn` on alhaalla (tilassa `0`), piiri on shutdown-tilassa, eli molemmat ulostulot `HO` ja `LO` ovat ei-aktiivisia. Normaalikäytössä piirin `TR-SDn`-sisääntulo on oltava ylhäällä (tilassa `1`) ja riippuen `TR-IN`-signaalin tilasta, vain toinen `HO`/`LO`-ulostuloista on aktiivisessa tilassa. Ulostuloista `LO` on aktiivisena kun sisääntulo `TR-IN` on ylhäällä (tilassa `1`) tai `HO` on aktiivisena kun `TR-IN` on alhalla (tilassa `0`). [IRS2004]

4.1.5 Ensiöpuolen ajuri

IRS2004 ajuripiirin lisäksi ohjauselektroniikkaan kuuluu useat muut oheiskompomponentit. Kondensaattorit (C27) on kytketty käyttöjännitteen ja maan välille vakauttamaan käyttöjännitettä. Siltaohjaimen Low-puolen hilan jännite (V_{LO}) määräytyy suoraan käyttöjännitteen V_{CC} ja maapisteen mukaan, joka riittää avaamaan N-kanavaisen MOSFETin sen lähteen (Source) ollessa kytkettynä samaan maapisteeseen. Käyttöjännite V_{CC} ohjaimelle on 12 V. Siltaohjaimen High-puolen MOSFETin avaamiseksi vaaditaan yhtälailla suurempi jännite hilalla kuin on sen lähteessä, mutta koska lähteen jännite MOSFETin auetessa nousee käyttöjännitteeseen V_{CC}, tarvitaan erillinen bootstrap-kytkentä. Tämän kytkennän muodostavat yhdessä kondensaattori C28, latausvirran rajoitusvastus R3 sekä diodi D12. High-puolen ollessa kiinni, latautuu kondensaattori C28 lähes V_{CC}-jännitteeseen (diodin jännitehäviö rajoittaa tätä hieman). Kun High-puoli avataan, nousee hilan jännite (V_{HO}) yhteensä kondensaattorin ja V_{CC}:n jännitteeseen, joka riittää avaamaan ja pitämään MOSFETin auki. Kondensaattorin varaus ei riitä kovin kauaa, joten siltaohjaimen on pidettävä myös Low-puolta auki jonkin aikaa. Tästä syystä ohjauksessa käytetään 50% pulssisuhteella toimivaa PWM-ohjausta.

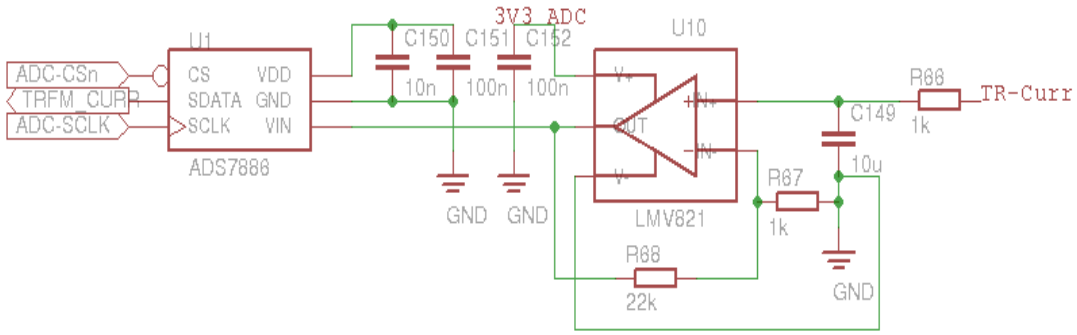
Pelkkä siltaohjain ja sen muodostama kanttimuotoinen lähtöjännite ei ole omiaan ohjaamaan muuntajaa tai siirtämään käytännössä minkäänlaista tehoa. Kanttiaalto on myös ongelmallinen sen sisältämien harmonisten taajuuksien vuoksi. Näistä syistä johtuen piirissä on myös kondensaattorien C30 – C32, C63 ja C64, kelojen L6 ja L7 sekä avomuuntajan ensiöpuolen muodostama LC-resonanssipiiri. Tällä kytkennällä on kaksi etua; suurempi avomuuntajan jännite ja aaltomuodon muuttaminen siniaalloksi. Resonanssipiiri ei olisi välttämätön mikäli kyseessä olisi tavallinen kiinteäsydäminen muuntaja, mutta koska kyseessä on avomuuntaja jonka magneettipiirissä on noin 1 mm ilmaväli, resonanssipiiri on toiminnan kannalta tarpeellinen. Suurempi jännite on suoraa seurausta resonanssipiirin toiminnasta ja samoin LC-kytkennän muodostama alipäästösuodatin karsii kanttiaallosta kaikki ylemmät taajuudet jättäen jäljelle lähes puhtaan siniaallon. Koska ohjaustaajuus on kohtuullisen matala (maksimissaan 100 kHz), ei tällainen muuntajaratkaisu myöskään aiheuta ongelmia EMC-mittauksissa.



KUVIO 1. Tehonsiirtomuuntajan ensiöpuolen ajuri (Kultanen 2010, muokattu).

4.1.6 Ensiöpuolen virran mittaus

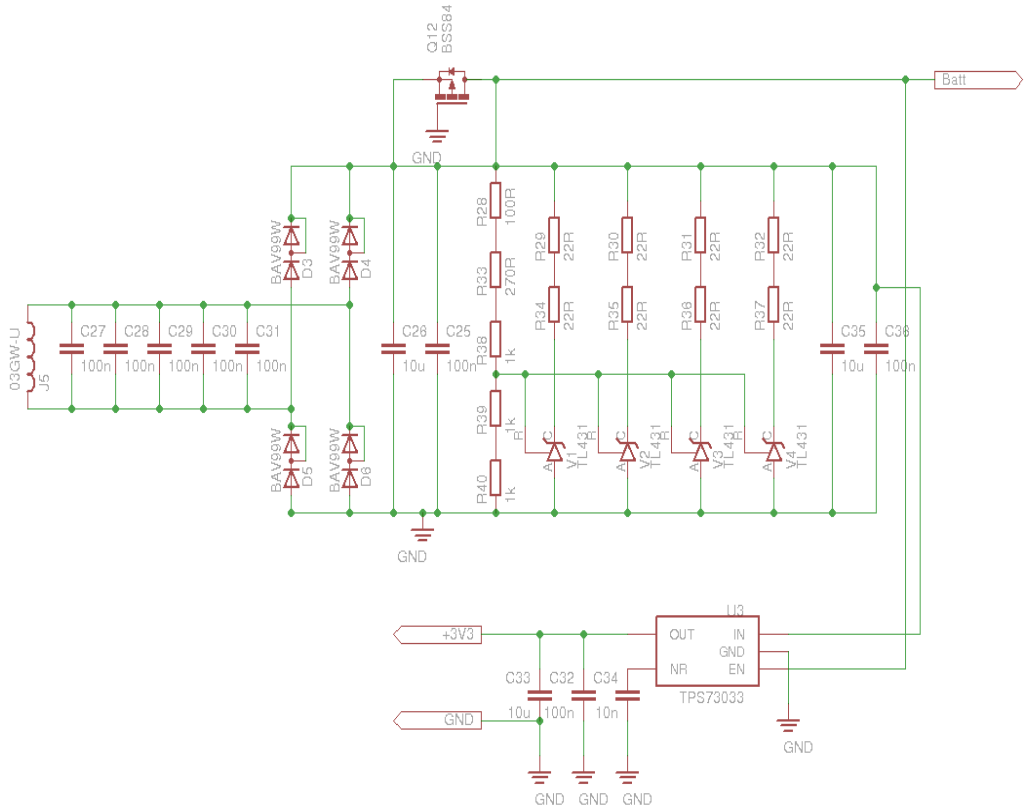
Muuntajan siirtämän energian maksimoinnin vuoksi on sen teho voitava mitata. Tehon mittausta varten tarvittaisiin normaalisti erillistä elektroniikkaa mittaamaan sekä jännitettä että virtaa, mutta jännitteen ollessa vakio, riittää pelkkä virran mittaus. Virran mittaus on toteutettu jännitteen mittauksena pienen (0.25 Ω) vastuksen yli. Ennen vahvistusta virta-arvo suodatetaan C149 ja R66 muodostamalla alipäästöpiirillä. Vahvistuksen jälkeen varsinaisesta virran mittauksesta huolehtii erillinen AD-muunnin ADS7886.



KUVIO 2. Tehonsiirtomuuntajan ensiöpuolen virran mittaus (Kultanen 2010, muokattu).

4.1.7 Toisiopuoli ja jännitteensyöttöpiiri

Muuntajan toisiopuoli (kuvio 3) muuntaa sisääntulevan vaihtojännitteen takaisin tasajännitteeksi. Myös vastaanottopuolella on avomuuntajan toisiopuolen sekä kondensaattorien C27 – C31 muodostama LC-resonanssipiiri. Vaihtojännite muunnetaan tasajännitteeksi diodien D3 – D6 muodostaman tasasuuntauskytkennän avulla. Neljä kappaletta rinnakkain olevia regulaattoreja (V1 – V4) kytkentöineen puolestaan pitää PLUS-pisteen jännitteen noin 4,2 voltissa, joka on kolmelle sarjaankytketylle Li-ion kennolle sopiva latausjännite. Regulaattorikytkentöjä on rinnakkain neljä siksi, että ne yhdessä kestävät suurimmatkin mahdolliset avomuuntajan jännitepiikit. MOSFET Q12 pitää huolen siitä, ettei akkujen varaus pääse koskaan tippumaan liian pieneksi kytkemällä piirin varsinaisen jännitteensyötön pois päältä.



KUVIO 3. Tehonsiirtomuuntajan toisiopuoli ja jännitteensyöttöpiiri (Kultanen 2010, muokattu).

4.2 VHDL

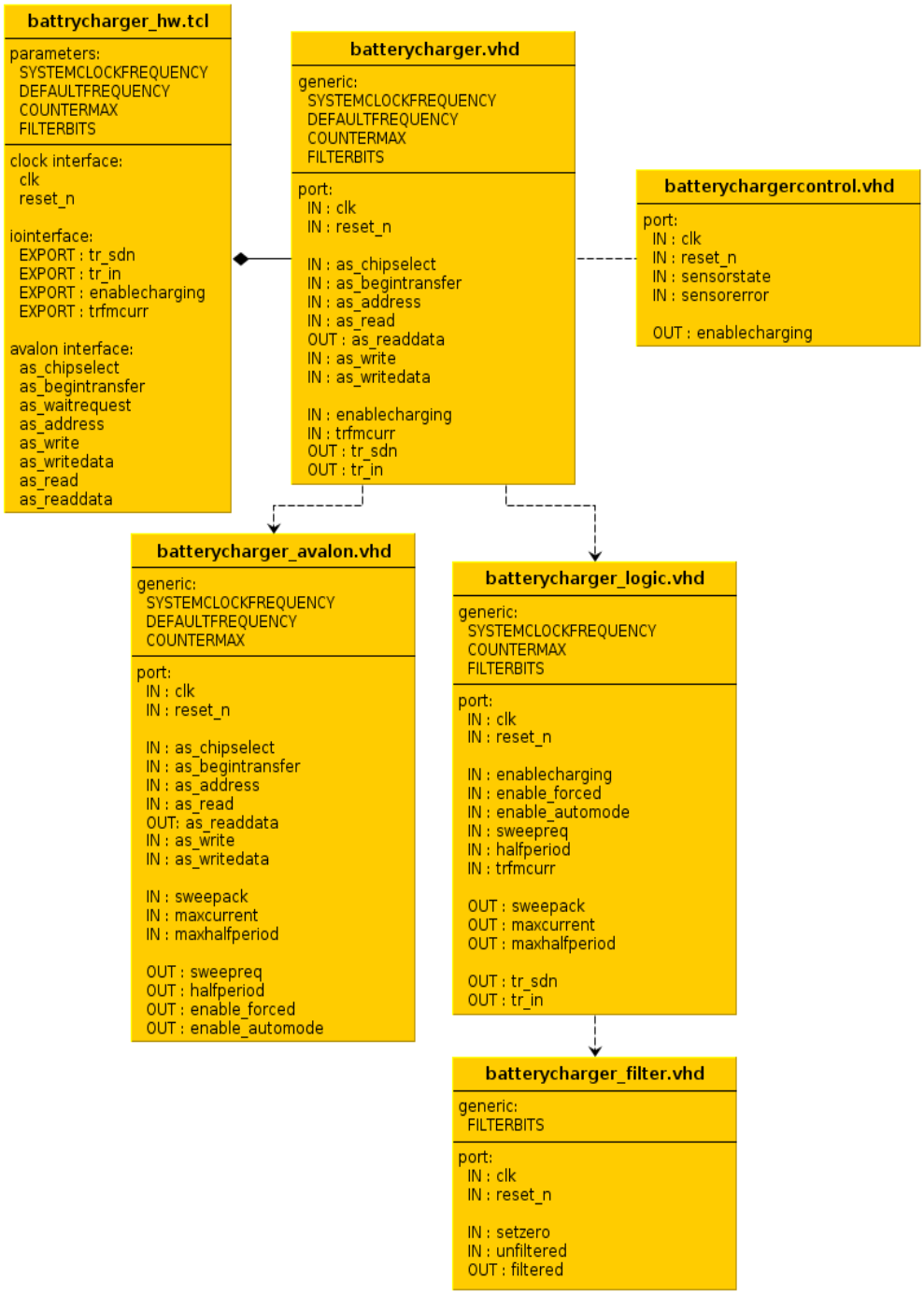
Laitteen akkujen latauspiiriä ohjaavat kaksi kirjastoa (*batterycharger* ja *batterychargercontrol*) on toteutettu VHDL-kielellä. VHDL-kieli (VHSIC Hardware Description Language) on yleisesti elektroniikkateollisuudessa käytetty laitteistokuvauskieli. Kieltä käytetään pääasiassa kuvaamaan erityisesti digitaalipiirien toimintaa ja rakennetta. [VHDL]

4.2.1 Batterycharger

Batterycharger-kirjasto koostuu viidestä eri tiedostosta:

- Päätason tiedosto *batterycharger.vhd*
- Avalon-väylään liittyvä kommunikointitiedosto (*batterycharger_avalon.vhd*)
- Logiikkatiedosto (*batterycharger_logic.vhd*) johon kuuluu alitiedostona:
  - Virta-arvon suodatustiedosto *batterycharger_filter.vhd*
- Laitteistokomponentin kuvaustiedosto (hardware component description file) *batterycharger_hw.tcl*.

Kokonaisuus on kuvattu alla olevassa UML-kaaviossa (kuvio 4). Kaaviossa on myös *batterycharger*-kokonaisuuteen liittyvä erillinen ja itsenäinen osa, *batterychargercontrol.vhd*. Molempien kokonaisuuksien kaikki lähdekoodit löytyvät kokonaisuudessaan liitteestä 1.



KUVIO 4. Batterycharger- ja batterychargercontrol -kirjastojen tiedostot UML-kaaviona.

Päätiedosto, *batterycharger.vhd*, sisältää Avalon-komponentin *batterycharger_avalon.vhd* ja logiikkakomponentin *batterycharger_logic.vhd*, sekä laitteistokomponentin kuvaustiedoston *batterycharger_hw.tcl*. Logiikkakomponentti sisältää lisäksi virransuodatukseen tarvittavan komponentin, *batterycharger_filter.vhd*.

Filter-komponentin tehtävänä on suodattaa alipäästösuodattimen tavoin mitatusta virrasta pois suuret muutokset ja hakea virran keskiarvo. Komponentti välittää mitatun virran ylemmälle tilankoneena toimivalle logiikkakomponentille (`batterycharger_logic`), joka asettaa mitatun virranarvon ja siihen liittyvän puolijakson maksimiarvoiksi. Logiikkatasolle välittyy myös akkujen latausmoodin tiedot käyttäjältä Avalon-komponentin avulla sekä control-komponentilta päätason (`batterycharger`) kautta. Näiden tietojen perusteella logiikkakomponentti välittää pääkomponentille akkujen latauksen sallivan tai estävän signaalin (`tr_sdn`).

4.2.2 Filter

Kaavion alimmaisena ohjelmayksikkönä on `batterycharger_filter` ja sen tiedosto *batterycharger_filter.vhd*. Ohjelmayksiköllä tarkoitetaan tässä dokumentissa VHDL-kielessä esiintyvää `entity`-kuvauksen ja sen `architecture`-toteutuksen muodostamaa kokonaista suunnitteluyksikköä. Yksikön tehtävä on suodattaa AD-muuntimelta tulevasta latausvirrasta kaikki suuret äkkinäiset muutokset. Ohjelmayksikkö `batterycharger_filter` on toiminnaltaan yksinkertainen digitaalinen 1. asteen alipäästösuodatin. Se aiheuttaa virran mittaukseen jonkin verran viivettä, mutta tämä on hyväksyttävä, sillä mittausjaksot ovat ylemmällä tasolla riittävän pitkiä. Suodattimen varsinainen toiminta kooditasolla on esitelty alla olevassa listauksessa (listaus 1).

```
if reset_n = '0' or setzero = '1' then
  filtered_s <= (others => '0');

elsif rising_edge(clk) then
  v_filtered       := unsigned(filtered_s);
  v_unfiltered     := resize(unsigned(unfiltered), FILTERBITS + 12);
  v_filtered_upper := resize(unsigned(filtered_s(FILTERBITS+11 downto
FILTERBITS)),FILTERBITS+12);

  filtered_s       <= std_logic_vector(v_filtered + v_unfiltered -
v_filtered_upper);
end if;
```

LISTAUS 1. Digitaalisen alipäästösuodattimen VHDL-kielinen toteutus

Suodattimen toiminta vastaa matemaattisesti kaavaa (1) alla:

$$\begin{aligned} y &= y + x - a \cdot y \\ y_{out} &= y \cdot a \end{aligned} \tag{1}$$

, missä y_{out} on suodatettu tulos, y on sisäinen, suuremman tarkkuuden arvo, x on sisäänmenevä arvo ja a on suodatuskerroin.

Tämä vastaa C-kielellä kaavaa (2) alla:

$$\begin{aligned} y &= y + x - (y \gg b) \\ yout &= y \gg b \end{aligned} \tag{2}$$

, missä b on suodatuskerroin ja muut ovat samat kuin yllä.

Yksikkö sisältää kello (`clk`) ja reset (`reset_n`) -signaalien lisäksi vain kaksi sisääntuloa (`unfiltered` ja `setzero`) ja yhden ulostulon (`filtered`). Sisääntulo (`unfiltered`) on suodattamaton virran signaali, josta suodatetaan suuret äkkinäiset muutokset pois ja tämä suodatettu virta asetetaan komponentin ulostuloon (`filtered`).

4.2.3 Logic

Logiikkayksikkö `batterycharger_logic` (tiedosto *batterycharger_logic.vhd*) pitää sisällään suodatuskomponentin `batterycharger_filter`. Logiikkayksikkö välittää sisääntulevan mittausvirran arvon suoraan suodatuskomponentille ja suodattimen ulostuloarvoa käytetään varsinaisessa vertauksessa.

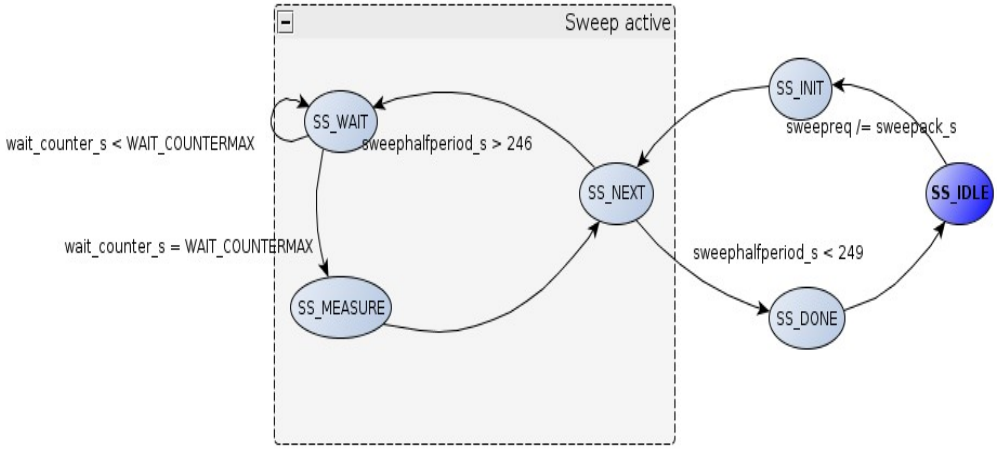
Logiikkayksikkö kykenee toimimaan kahdessa eri tilassa. Nämä ovat normaalitila ja automaattinen pyyhkäisymittaustila. Normaalitilassa yksikkö toimii yksinkertaisena PWM-generaattorina jonka taajuus riippuu sisääntulevasta `halfperiod`-arvosta. Kuten nimi sanoo, halfperiod viittaa 50% pulssisuhteella toimivan PWM:n taajuuden puolijaksoon. Sisääntuleva halfperiod-luku ei siis ilmoita suoraan kellotaajuutta, vaan pulssin puolikkaan kellojaksojen lukumäärän. Tämä luku voidaan muuntaa takaisin taajuudeksi kaavan (3) mukaisesti.

$$f = \frac{1}{2 \cdot (T_h + 1)} \tag{3}$$

, missä f on taajuus ja T_h on halfperiod-arvo.

Kaavan erikoisuus, yhden lisääminen T_h arvoon, johtuu suoraan VHDL-koodin toteutuksesta, sillä siellä tehokkuudesta johtuen kaikki laskurit alkavat arvosta 0 mikä puolestaan kasvattaa varsinaista taajuutta hieman. Muutoin kaava on normaali taajuuden ja jaksonpituuden välinen muunnoskaava.

Logiikkayksikön toinen toimintatila on automaattinen pyyhkäisymittaustila. Sen tarkoituksena on hakea maksimivirran ja siihen liittyvän puolijakson arvot `batterycharger_filter`-komponentin suodattaman latausvirran avulla. Yksikön pyyhkäisytilakoneen toiminta on kuvattu alla olevassa kuviossa (kuvio 5) ja sen tilasiirtymät myöhempänä olevassa taulukossa (taulukko 1).



KUVIO 5. Pyyhkäisytilakoneen tilat ja tilasiirtymät.

Pyyhkäisytilakone on alussa toimettomana idle-tilassa (`SS_IDLE`) , kunnes sille tulee pyyhkäisyn aloituspyyntö. Aloituspyyntö on voimassa kun `sweepreq`-sisääntulosignaalissa on erisuuri arvo kuin sisäisessä `sweepack_s`-signaalissa. Aloituspyynnön jälkeen tilakone siirtyy seuraavaan alustustilaan (`SS_INIT`) , jossa nollataan maksimipuolijakso, maksimivirta ja suodatuskomponentin nykyinen arvo sekä alustetaan pyyhkäisyssä käytettävän puolijakson lukema maksimiarvoonsa (1250). Tämän jälkeen siirrytään `SS_NEXT`-tilaan, jossa puolijaksoa pienennetään yhdellä. Mikäli puolijakso on pienentämisen jälkeen suurempi kuin minimiarvo (249) siirrytään odotustilaan (`SS_WAIT`).

Odotustilassa PWM-generaattori tuottaa uutta taajuuttaa nykyisen puolijakson arvon perusteella ja elektroniikka ”ottaa käyttöön” tämän uuden taajuuden. Muutokset taajuudessa heijastuvat pienellä viiveellä suoraan muuntajan tehokkuuteen. Tehokkuuden muutos puolestaan näkyy suoraan virran arvossa. Odotustilassa on laskuri joka kasvaa yhdellä jokaisella kellonjaksolla. Tilakone pysyy tässä tilassa niin kauan kunnes laskuri on saavuttanut halutun määrän, jonka jälkeen siirrytään mittaustilaan (`SS_MEASURE`).

Mittaustilassa asetetaan mitattu ja suodatettu virran arvo maksivirraksi sekä siihen liittyvä puolijakso maksimipuolijaksoksi mikäli suodatettu virta-arvo on suurempi kuin sen hetken maksimivirta-arvo. Mittaustilasta palataan aina `SS_NEXT`-tilaan, jossa jälleen

kerran pienennetään käytössä olevaa puolijaksoa ja siirrytään odotustilaan ja niin edelleen.

Tästä syntyy silmukka, joka päättyy ainoastaan silloin kun puolijakso on pienentynyt minimiarvoon (249). Tämän toteutuessa siirrytään valmistilaan (`SS_DONE`), jossa `sweepack_s`-signaali asetetaan `sweepreq`-signaalin arvoon, ilmentäen ylemmälle tasolle mittauksen olevan nyt valmis. Lopuksi siirrytään takaisin alkutilaan (`SS-IDLE`).

Alla oleva taulukko (taulukko 1) kuvaa selkeästi pyyhkäisytilakoneen toimintaa. Taulukosta nähdään mihin tilaan voidaan siirtyä tietystä tilasta sekä mihin tilaan siirtyminen ei ole mahdollista. Tilakoneen toiminta on kuvattu ohjelmointikoodin tasolla alla olevassa listauksessa (listaus 2).

TAULUKKO 1. Pyyhkäisytilakoneen tilasiirtymät.

Mistä \ Mihin	SS_IDLE	SS_INIT	SS_NEXT	SS_WAIT	SS_MEASURE	SS_DONE
SS_IDLE	**1**	**1**	0	0	0	0
SS_INIT	0	0	**1**	0	0	0
SS_NEXT	0	0	0	**1**	0	**1**
SS_WAIT	0	0	0	**1**	**1**	0
SS_MEASURE	0	0	**1**	0	0	0
SS_DONE	**1**	0	0	0	0	0

```
case sweep is
  when SS_INIT =>
    sweephalfperiod_s <= 1250;
    maxcurrent_s      <= (others => '0');
    maxhalfperiod_s   <= 0;
    sweep             <= SS_NEXT;
    setzero_s         <= '1';
  when SS_NEXT =>
    setzero_s <= '0';
   if sweephalfperiod_s > 249 then
      sweephalfperiod_s <= sweephalfperiod_s - 1;
      sweep             <= SS_WAIT;
    else
      sweep <= SS_DONE;
    end if;
   when SS_WAIT =>
    if wait_counter_s = WAIT_COUNTERMAX then
      wait_counter_s <= 0;
      sweep          <= SS_MEASURE;
    else
      wait_counter_s <= wait_counter_s + 1;
      sweep          <= SS_WAIT;
    end if;
  when SS_MEASURE =>
    if filtered_s > maxcurrent_s then
      maxcurrent_s    <= filtered_s;
      maxhalfperiod_s <= sweephalfperiod_s;
    end if;
   sweep <= SS_NEXT;
  when SS_DONE =>
    sweepack_s    <= sweepreq;
    sweep         <= SS_IDLE;
    sweepactive_s <= '0';
  when others =>
    - In case of an unknown state, go back to idle state
    sweep <= SS_IDLE;
end case;
```

LISTAUS 2. Pyyhkäisytilakoneen VHDL-toteutus.

Ylläolevassa listauksessa tilakoneen ollessa `SS_WAIT`-tilassa laskuria (`wait_counter_s`) kasvatetaan niin kauan kunnes se saavuttaa halutun odotusajan (`WAIT_COUNTERMAX`), jonka jälkeen siirrytään mittaustilaan (`SS_MEASURE`). Laskurin ollessa pienempi kuin `WAIT_COUNTERMAX`, siirtyy tilakone aina uudestaan samaan tilaan jatkamaan laskurin kasvattamista. Virran mittaustilassa tarkastellaan ensin suodatettua virtaa. Mikäli tämä virta on suurempi kuin tämänhetkinen maksimivirta, asetetaan maksimivirraksi suodatettu virta sekä siihen liittyvän puolijakson arvo maksimipuolijaksoksi. Tämän jälkeen siirrytään seuraavaan tilaan `SS_NEXT`.

4.2.4 Avalon

Alteran Quartus-ympäristössä kommunikointi eri rajapintojen, kuten ohjelmistoprosessorien, muistien, IO-väylien ja omien komponenttien, välillä tapahtuu Avalon® -nimisen väylän kautta. Väylän toteutus käyttäjän kannalta on melko yksinkertainen, mutta kommunikointi sen kautta on silti erittäin nopeaa ja vaivatonta. Quartus-kääntäjä ja väylän rakenne huolehtii automaattisesti kaikkien tarpeellisten signaalien reitittämisestä, vuorojen jaosta, ajastuksista ja muusta oleellisesta. [MNL-AVABUSREF-2.0]

Avalonyksikkö `batterycharger_avalon` (tiedosto *batterycharger_avalon.vhd*) mahdollistaa VHDL-ohjelmakirjaston ja C-koodin välisen kommunikoinnin. Yksinkertaisen Avalon-väylän avulla voidaan sekä kirjoittaa että lukea tietoa komponenttiin tai komponentista. Lisäksi Avalon-väylän avulla voidaan hallita siihen kytkettyjä laitteita ja komponentteja.

Ohjelmayksikön *`batterycharger_avalon`* avulla voidaan rekistereistä lukea seuraavia asioita:

- Akkujen latausmoodi (on/off/automatic).
- Nykyinen normaalitilan puolijakson arvo `halfperiod`.
- Pyyhkäisytila (aktiivinen/ei-aktiivinen).
- Pyyhkäisysyklin mittaamat maksimivirta sekä siihen liittyvä puolijakson arvo.

Yksikön avulla voidaan puolestaan kirjoittaa rekistereihin seuraavia asioita:

- Akkujen latausmoodi (on/off/automatic)
- Normaalitilan puolijakson arvo `halfperiod`.
- Pyyhkäisysyklin aloituspyyntö.

Avalonyksikkö sisältää muiden VHDL-tiedostojen tavoin ensin tarvittavat geneeriset muuttujat, sekä porttilistan sisään- ja ulostuloista. Avalonyksikkö sisältää vain muuta-

mat sisäiset signaalit. Luku- ja kirjoitusrekisterit on nimetty tarkoituksensa mukaisesti seuraavasti (listaus 3):

```
-- Avalon registers
constant AVALONADDR_RW_MODE               : integer := 16#00#;
constant AVALONADDR_RW_HALFPERIOD         : integer := 16#01#;
constant AVALONADDR_RW_SWEEP              : integer := 16#02#;
constant AVALONADDR_R_SWEEP_MAXHALFPERIOD : integer := 16#03#;
constant AVALONADDR_R_SWEEP_MAXCURRENT    : integer := 16#04#;
```

LISTAUS 3. Avalon-väylän rekisterien osoitteet.

Osoitteessa 0 on akkujen latausmoodin luku ja kirjoitus. Osoitteen käyttömoodi osoitetaan suoraan sen nimessä (`RW` = Read/Write eli osoitteesta voidaan lukea ja kirjoittaa). Samaan tyyliin osoitteessa 1 on nykyisen normaalitilassa käytössä olevan puolijakson arvon luku ja kirjoitus. Osoite 2 puolestaan toimii hieman eri tavalla, sillä sinne ei kirjoiteta mitään tiettyä arvoa, vaan kirjoitus aiheuttaa automaattisesti uuden pyyhkäisysyklin aloituksen (mikäli se ei tällä hetkellä ole aktiivisena). Luku tästä rekisteristä taas kertoo onko sykli tällä hetkellä aktiivisena vai ei.

Maksimipuolijakson nykyisen arvon luenta tehdään osoitteesta 3. Myös tässä osoitteen käyttömoodi (`R` = Read only eli osoitteesta voidaan ainoastaan lukea) näkyy suoraan sen nimessä. Kolmas vaihtoehto nimityksille olisi luonnollisesti `W` = Write only, eli osoitteeseen voitaisiin ainoastaan kirjoittaa. Nykyinen mitattu maksimivirran arvo puolestaan luetaan osoitteesta 4. Prosessin sisällä tiedon luku toimii esimerkkinä alla olevan listauksen (listaus 4) tapaan.

```
---------------------------------------
-- Set/get charger state and other variables via avalon

if as_chipselect = '1' then
  if as_read = '1' and as_begintransfer = '1' then
    as_readdata_s <= (others => '0');

    case to_integer(unsigned(as_address)) is

      when AVALONADDR_RW_HALFPERIOD =>
        -- Read current halfperiod
        as_readdata_s <= std_logic_vector(to_unsigned(halfperiod_s,
32));

… KOODIA POISTETTU …

    end case;
```

LISTAUS 4. Avalon-väylän osoitteen luennan VHDL-toteutus.

Mikäli piirin valinta on päällä (`as_chipselect='1'`), sekä lukuvalinta on päällä (`as_read='1'`) ja tiedonsiirtolupa annettu (`as_begintransfer='1'`), voidaan varsinainen luenta suorittaa. Aluksi nollataan lukurekisteri (`as_readdata_s`). Seuraavaksi tarkastellaan pyydetyn osoitteen arvo (`as_address`). Mikäli osoitteen arvo vastaa esimerkissä olevan puolijakson luku/kirjoitus -osoitteen arvoa (`AVALONADDR_RW_HALFPERIOD`), suoritetaan luenta puolijakson rekisterin (`halfperiod_s`) arvosta. Tässä tapauksessa luenta on yksinkertainen, ainoastaan vektorin pidennys lukurekisteriin sopivaksi. Kirjoitus vastaavasti toimii lähes samalla tavalla seuraavan listauksen (listaus 5) mukaisesti:

```
  elsif as_write = '1' and as_begintransfer = '1' then
    case to_integer(unsigned(as_address)) is

… KOODIA POISTETTU …

      when AVALONADDR_RW_HALFPERIOD =>
        -- Write new halfperiod value
        halfperiod_s <= to_integer(unsigned(as_writedata));

… KOODIA POISTETTU …

    end case;
```

LISTAUS 5. Avalon-väylän osoitteen kirjoituksen VHDL-toteutus.

Yllä olevasta koodista on jätetty pois piirin valinnan tarkistus, sillä se tehdään yhteisesti sekä luvulle että kirjoitukselle. Kun kirjoitusvalinta on päällä (`as_write='1'`) ja tiedonsiirto on sallittu (`as_begintransfer='1'`), kirjoitetaan haluttuun osoitteeseen ja signaalin C-koodista annettu arvo. Esimerkissä kirjoitetaan puolijakson arvoksi C-koodista annettu uusi puolijakson arvo. Kirjoitus tapahtuu käytännössä samalla tavalla kuin luentakin, tosin sillä erotuksella että kirjoituksessa muunnetaan kirjoitusrekisterissä (`as_writedata`) oleva arvo takaisin kokonaisluvuksi.

4.2.5 Laitteistokomponentin kuvaustiedosto

Laitteistokomponentin kuvaustiedosto b*atterycharger_hw.tcl* on VHDL-kirjaston, ja tarkemmin sen päätason sekä Avalon-moduulin, porttien määrittely- ja kuvaustiedosto. Tämä tiedosto määrittelee kaikki moduulin tarvitsemat Avalon-väylän ominaisuudet ja signaalien nimet (listaus 6), kello- ja reset-signaalien nimet (listaus 7) sekä reaalimaailmaan liittyvien signaalien nimet (listaus 8). Siinä annetaan myös lisätietoa kirjastosta kuten moduulin nimi, kirjoittaja, versio sekä geneeriset parametrit (listaus 9) Sen pääasiallinen sisältö koostuu erilaisista interface-, eli liittymämäärittelyistä.

```
# Avalon slave interface
# Properties
add_interface "AvalonSlaveInterface" "avalon" "slave" "clock_inter-
face"
set_interface_property "AvalonSlaveInterface" "isNonVolatileStorage"
"false"
set_interface_property "AvalonSlaveInterface" "burstOnBurstBoundarie-
sOnly" "false"
set_interface_property "AvalonSlaveInterface" "readLatency" "0"
set_interface_property "AvalonSlaveInterface" "holdTime" "0"
set_interface_property "AvalonSlaveInterface" "printableDevice" "fal-
se"
set_interface_property "AvalonSlaveInterface" "readWaitTime" "1"
set_interface_property "AvalonSlaveInterface" "setupTime" "0"
set_interface_property "AvalonSlaveInterface" "addressAlignment" "DY-
NAMIC"
set_interface_property "AvalonSlaveInterface" "writeWaitTime" "0"
set_interface_property "AvalonSlaveInterface" "timingUnits" "Cycles"
set_interface_property "AvalonSlaveInterface" "minimumUninterrup-
tedRunLength" "1"
set_interface_property "AvalonSlaveInterface" "isMemoryDevice" "false"
set_interface_property "AvalonSlaveInterface" "linewrapBursts" "false"
set_interface_property "AvalonSlaveInterface" "maximumPendingReadTran-
sactions" "0"

# Ports
add_port_to_interface "AvalonSlaveInterface" "as_chipselect" "chipse-
lect"
add_port_to_interface "AvalonSlaveInterface" "as_begintransfer" "be-
gintransfer"
add_port_to_interface "AvalonSlaveInterface" "as_address" "address"
add_port_to_interface "AvalonSlaveInterface" "as_write" "write"
add_port_to_interface "AvalonSlaveInterface" "as_writedata" "writeda-
ta"
add_port_to_interface "AvalonSlaveInterface" "as_read" "read"
add_port_to_interface "AvalonSlaveInterface" "as_readdata" "readdata"
```

LISTAUS 6. Avalon-väylän määritteet ja signaalinimet.

```
# Interface clock_sink
add_interface "clock_interface" "clock" "sink" "asynchronous"
add_port_to_interface "clock_interface" "clk" "clk"
add_port_to_interface "clock_interface" "reset_n" "reset_n"
```

LISTAUS 7. Kelloväylän määritteet ja signaalinimet.

```
# I/O
add_interface "iointerface" "conduit" "output" "clock_interface"
# Ports in interface iointerface
add_port_to_interface "iointerface" "tr_sdn" "export"
add_port_to_interface "iointerface" "tr_in" "export"
add_port_to_interface "iointerface" "enablecharging" "export"
add_port_to_interface "iointerface" "trfmcurr" "export"
```

LISTAUS 8. Reaalimaailman signaalien nimet.

```
set_source_file "batterycharger.vhd"
set_module "batterycharger"
set_module_description "batterycharger"
set_module_property "displayName" "batterycharger"
set_module_property "author" "Thuy Nguyen"
set_module_property "libraries" [ list "ieee.std_logic_1164.all"
"ieee.numeric_std.all" "std.standard.all" ]
set_module_property instantiateInSystemModule true
set_module_property version "2.0"
set_module_property group "ess"
set_module_property editable true

# Module parameters
add_parameter "SYSTEMCLOCKFREQUENCY" "integer" "50000000" ""
set_parameter_property "SYSTEMCLOCKFREQUENCY" AFFECTS_PORT_WIDTHS tr
add_parameter "DEFAULTFREQUENCY" "integer" "14000" ""
set_parameter_property "DEFAULTFREQUENCY" AFFECTS_PORT_WIDTHS true
add_parameter "COUNTERMAX" "integer" "50000000" ""
set_parameter_property "COUNTERMAX" AFFECTS_PORT_WIDTHS true
add_parameter "FILTERBITS" "integer" "5" ""
set_parameter_property "FILTERBITS" AFFECTS_PORT_WIDTHS true
```

LISTAUS 9. Moduulin parametrit ja muut määritteet.

4.2.6 Batterychargercontrol

Koska akkujen lataus voidaan normaaliolosuhteissa sallia ainoastaan tietyssä tapauksessa, tarvitaan erillinen `batterycharger`-yksikköä (tiedosto *batterycharger.vhd*) ohjaava `batterychargercontrol`-yksikkö. Ohjausyksikkö on hyvin yksinkertainen VHDL-kirjasto, joka koostuu ainoastaan yhdestä tiedostosta *batterychargercontrol.vhd*. `Batterychargercontrol` sisältää vain muutaman signaalin, joiden tilojen vaihtelun perusteella yksikkö välittää akkujen latauksen sallivan tai estävän signaalin (`enablecharging`) `batterycharger`-yksikölle.

Sisääntuleva signaalivektori `sensorstate` kertoo ohjausyksikölle AMR-anturien tilan ('1' = AMR-anturi tunnistaa magneettikentän, '0' = ei magneettikenttää). Näistä kahden sensorin täytyy tunnistaa magneettikenttä, jolloin tiedetään laitteessa olevan lavan olevan oikeassa asennossa ja lataus voidaan sallia. Laitteessa on myös muita AMR-antureja, joten tämän ohjausyksikön päätehtävä valita niistä oikeat signaalit. Signaalivektori `sensorerror` puolestaan kertoo AMR-anturien virhetilanteesta ('1' = anturi ei tunnis-

tettu/rikki/ei kytkettynä tai '0' = anturi kunnossa). Ulostulo `enablecharging` määräytyy näiden kahden sisääntulosignaalin perusteella seuraavan taulukon mukaisesti (taulukko 2). Taulukossa ei ole erikseen määritelty mitkä kaksi anturitilaa tulee olla '1', vaan niitä molempia kuvataan ainoastaan yhdellä arvolla. Yksi arvo riittää kuvaamaan molempia sensoreja, sillä niiden molempien on oltava '1' (looginen AND-operaatio). Myös ohjausyksikön koodin toiminnallinen osa on kuvattu seuraavassa listauksessa (listaus 10).

TAULUKKO 2. Ohjausyksikön ulostulon määräytyminen sisääntulosignaaleista.

sensorerror	sensorstate	enablecharging
0	1	salli lataus (1)
0	0	estä lataus (0)
1	ei väliä	estä lataus (0)

```
---------------------------------------
--! Set enablecharging signal
process(clk, reset_n)
begin
  if reset_n = '0' then
    enablecharging_s <= '0';
  elsif rising_edge(clk) then
    -- default is off
    enablecharging_s <= '0';
    -- if sensors are present
    if sensorerror(1) = '0' and sensorerror(2) = '0' then
      -- and sensor states are 1 then enable charging
      if sensorstate(1) = '1' and sensorstate(2) = '1' then
        enablecharging_s <= '1';
      end if;
    end if;
  end if;
end process;
```

LISTAUS 10. Ohjausyksikön batterychargerycontrol VHDL-toteutus.

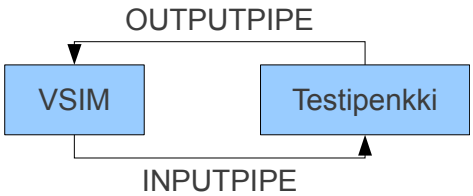
`Batterychargercontrol`-ohjausyksiköltä lähtevä salliva ulostulo mahdollistaa lapojen akkujen latauksen latausmoodin ollessa automaatti-tilassa. Akkujen latausmoodin ollessa manuaalitilassa (joko pakotettuna päälle tai pois), ei tällä ulostulolla ole merkitystä latauksen kannalta, vaan lataukseen vaikuttaa käyttäjä Avalon-väylän kautta.

4.3 Simulaatioympäristö

VHDL-kielellä toteutetut ohjelmakomponenti tarvitsevat tässä vaiheessa simulointia, jolla varmistetaan koodin toimivuus ennen sen rautatestausta. Tätä varten on kehitetty kokonaan oma VHDL-kielen simulaatioympäristö nimeltä VSIM. Oman simulaatioympäristön kehittämiseen päädyttiin markkinoilta löytyvien simulaatioympäristöjen puutteiden, kuten hitaus, kankeus ja hankala käytettävyys, vuoksi. Ympäristön on kehittänyt pääosin Teemu Karimerto.

VSIM-ympäristö on kirjoitettu lähes kokonaan Python-ohjelmointikielellä. Ainoastaan testipenkin päätason tiedosto *tb.vhdl* ja testipenkin ohjaustiedosto *ctrl.vhdl* ovat VHDL-kielisiä. Normaalisti näillä kahdella ohjelmointikielellä ei ole mitään tekemistä eikä minkäänlaista yhteyttä keskenään, mutta Linuxin tiedostojärjestelmän putkien (pipe) avulla yhteys saadaan luotua. Varsinainen testipenkki koostuu näistä kahdesta ohjaustiedostosta sekä yhdestä tai useammasta testattavasti VHDL-komponentista. Juuri ennen testausta käännetään testipenkki GHDL-ohjelman avulla ajettavaksi ohjelmaksi.

Kommunikointi Python-kielisen VSIM-ympäristön ja käännetyn testipenkin välillä tapahtuu kahden putkitiedoston avulla. Nämä tiedostot avataan molemmista ohjelmista ristiin toinen lukua ja toinen kirjoitusta varten kuvion (kuvio 6) mukaisesti. Simulaatioympäristön sisäinen toiminta on osittain suljettua koodia ja tämän dokumentin ulkopuolella, joten sitä ei tässä sen tarkemmin esitellä.



KUVIO 6. Putkikommunikaatio.

4.4 Simulaatiotestit

Testausta varten luodaan jokaista VHDL-koodia varten ensin pieni testimoduli, käyttäen yksinkertaista Python-ohjelmointikieltä. Testattua jokaisen koodin toimivuutta onnistuneesti, on seuraava tehtävänä testata kokonaisuus yhdellä testiohjelmalla. Testausohjelmilla pyritään testaamaan kaikkia `batterycharger`:n ja `batterychargercontrol`:n osia kaikilla mahdollisilla parametreillä. Näin varmistetaan mahdollisimman hyvin koodin toimivuus ennen sen siirtämistä oikeaan lopulliseen käyttöympäristöön.

VHDL-yksikön ylimmän tason signaalien arvot voidaan lukea ja kirjoittaa `self.mod` -Python-luokan (tyyppiä `vsim.vhdltester.Command`) avulla. Luokka osaa toimia automaattisesti sekä luku- että kirjoitustilanteessa joko asettamalla signaaliin uuden arvon tai vaihtoehtoisesti lukemalla signaalin nykyisen arvon. Signaalien uudet arvot eivät tosin automaattisesti välity VHDL-koodiin asti ennenkuin ajetaan yksi kellojakso. Signaalien nimet ja suunnat generoituvat automaattisesti suoraan testattavasta VHDL-koodista ja niihin viitataan Pythonin pistenotaatiolla (esim. `self.mod.enablecharging` viittaa suoraan samannimiseen signaaliin `batterycharger` -yksikössä).

Testimoduleita on yhteensä viisi, yksi jokaista VHDL-tiedostoa kohti. Testit on nimetty testattavan VHDL-tiedoston mukaisesti:

- *test_batterycharger_filter.py*, suodatinkomponentin testi
- *test_batterycharger_logic.py*, logiikkakomponentin testi
- *test_batterycharger_avalon.py*, Avalon-moduulin testi
- *test_batterycharger.py*, kokonaisuuden testi
- *test_batterychargercontrol.py*, ohjausyksikön testi

4.4.1 Filter-testi

Testi *test_batterycharger_filter.py* on hyvin yksinkertainen. Tällä halutaan testata `filter`-komponentin toimintaa, latausvirran suodatusta kaikilta suurilta virranmuutoksilta. Testi sisältää kaksi pientä funktiota, apufunktio `calc` ja varsinainen testi `TestFilter`. `calc`-funktio sisältää tiedostossa *batterycharger_filter.vhd* VHDL-kielellä toteutetun varsinaisen suodatuslogiikan Python-kielellä. `TestFilter`-funktio arpoo satunnaisia lukuja ja kutsuu silmukassa sekä `calc`-funktiota että ajaa yhden kellojakson suodattimen VHDL-koodia samalla parametrillä. Suodatetut tulokset on oltava yhtäläisiä sekä Python-toteutuksen että VHDL-koodin välillä. Suodatettu arvo ei myöskään saa vaihdella suuresti, mutta sitä tässä testissä ei erikseen huomioida. Testikoodi on esiteltynä alla (listaus 11).

```
def calc(self, y, x, bits = 5):
      shifty = y >> bits
      y = y + x - shifty
      return (y, (y >> bits))

def TestFilter(self):
      """ Test filter. """
      y = 0
      for i in xrange(1000):
            x = 1000 + random.randint(-50, 50)
            self.mod.unfiltered = x
            y, yval = self.calc(y, x)
            self.Tick()
            assert self.mod.filtered == yval
```

LISTAUS 11. Suodatinkomponentin testauskoodi test_batterycharger_filter.py.

4.4.2 Logic-yksikkö ja sen kietoja

Logiikkatiedosto sisältää muutaman geneerisen muuttujan joilla on testauksen kannalta erittäin suuret arvot, ja testaus siis kestäisi aivan liian kauan. Tästä syystä *batterycharger_logic.vhd* -tiedostolle on tehty *batterycharger_logic_wrapper.vhd* -kietojatiedosto. Tämä on läpinäkyvä kietoja *batterycharger_logic.vhd* -tiedostolle, jolla saadaan kätevästi vaihdettua VHDL-komponentin `generic`-määritteitä sopivammiksi nopeampaa testaamista varten, muokkaamatta kuitenkaan alkuperäistä lähdekoodia. Logiikkayksik-

kö sisältää `generic`-määritteitä, jotka ovat arvoiltaan liian suuria käytettäväksi testauksessa. Ilman arvojen muokkaamista tämä tarkoittaisi sitä, että testit tulevat kestämään liian pitkään eikä pitkä kesto ole oleellinen komponentin toiminnan varmistamisessa. Kietojatiedostolla *batterycharger_logic_wrapper.vhd* korvataan määritteet `SYSTEMCLOCKFREQUENCY` ja `COUNTERMAX` pienemmillä arvoilla seuraavasti (taulukko 3):

TAULUKKO 3. Logiikkakomponentin kietojan muuttamat generic-arvot.

	batterycharger_logic.vhd	*batterycharger_logic_wrapper.vhd*
`SYSTEMCLOCKFREQUENCY`	50000000	2000
`COUNTERMAX`	50000000	2000

Varsinainen VHDL-kielinen toteutus kietojatiedostoille on hyvin yksinkertainen. Se sisältää alkuperäisen yksikön kanssa täysin identtisen esittelyn josta se muuttaa ainoastaan `generic`-arvoja (listaus 12), komponenttina alkuperäisen yksikön (listaus 13) sekä suoran signaalien reitityslistan (listaus 14). Mitään muuta toimintaa tai logiikkaa kietojassa ei ole, joten se ei testaajalle näy muutoin kuin hieman eri nimisenä yksikkönä (alkuperäisen yksikön nimi `_wrapper` -loppupäätteellä).

```
entity batterycharger_logic_wrapper is
  generic (
    SYSTEMCLOCKFREQUENCY : integer := 2000;  -- [Hz] System clock
frequency
    COUNTERMAX           : integer := 2000;  -- Maximum count value
    FILTERBITS           : integer := 5);
  port (
    clk     : in std_logic;             --! System clock
    reset_n : in std_logic;             --! Reset (active low)

    enablecharging  : in std_logic;     --! Enable charging by automa-
tic logic
    enable_forced   : in std_logic;     --! Enable charging via avalon
    enable_automode : in std_logic;     --! Mode is automatic
    sweepreq        : in std_logic;     --! Sweep request
    halfperiod      : in integer range 0 to COUNTERMAX;
    trfmcurr        : in std_logic_vector(11 downto 0);  --! Input for
transformed current

    sweepack      : out std_logic;      --! Sweep acknowledge
    maxcurrent    : out std_logic_vector(11 downto 0);  --! Maximum
filtered current
    maxhalfperiod : out integer range 0 to 2047;

    tr_sdn : out std_logic;             --! Output for charging state
    tr_in  : out std_logic);            --! Output for enabling trans-
former
end entity batterycharger_logic_wrapper;
```

LISTAUS 12. Logiikkakomponentin kietojan entity-esittely.

```
architecture rtl of batterycharger_logic_wrapper is
  component batterycharger_logic
    generic (
      SYSTEMCLOCKFREQUENCY : integer;
      COUNTERMAX           : integer;
      FILTERBITS           : integer);
    port (
      clk     : in std_logic;
      reset_n : in std_logic;

      enablecharging  : in std_logic;
      enable_forced   : in std_logic;
      enable_automode : in std_logic;
      sweepreq        : in std_logic;
      halfperiod      : in integer range 0 to COUNTERMAX;
      trfmcurr        : in std_logic_vector(11 downto 0);

      sweepack      : out std_logic;
      maxcurrent    : out std_logic_vector(11 downto 0);
      maxhalfperiod : out integer range 0 to 2047;

      tr_sdn : out std_logic;
      tr_in  : out std_logic);
  end component;
```

LISTAUS 13. Logiikkayksikön komponenttiesittely.

```
begin
  batterycharger_logic_inst : batterycharger_logic
    generic map (
      SYSTEMCLOCKFREQUENCY => SYSTEMCLOCKFREQUENCY,
      COUNTERMAX           => COUNTERMAX,
      FILTERBITS           => FILTERBITS)

    port map (
      clk             => clk,
      reset_n         => reset_n,
      enablecharging  => enablecharging,
      enable_forced   => enable_forced,
      enable_automode => enable_automode,
      sweepreq        => sweepreq,
      halfperiod      => halfperiod,
      trfmcurr        => trfmcurr,
      sweepack        => sweepack,
      maxcurrent      => maxcurrent,
      maxhalfperiod   => maxhalfperiod,
      tr_sdn          => tr_sdn,
      tr_in           => tr_in);
end;
```

LISTAUS 14. Logiikkakomponentin kietojan signaalien reitityslista.

4.4.3 Logic testi

Logiikkatesti *test_batterycharger_logic.py* sisältää pyyhkäisytilakoneen ja virransuodatuksen toiminnan yhteisen testauksen, sillä *batterycharger_logic.vhd* sisältää komponenttina suodatinyksikön `batterycharger_filter`. Näin samalla testillä voidaan testata molempien komponenttien toimintaa yhdessä. Testi sisältää kaksi testiä: `TestHalfperiodSweep` ja `TestChargingMode`. Ensimmäisen testin (`TestHalfperiodSweep`) alussa alustetaan kaksi taulukkoa sisäänmeneville (`values_in = []`) ja ulostuleville (`values_out = []`) virroille, sekä kaksi muuttujaa: toinen maksimivirralle (`max_x = 1000`) ja toinen virran rippelille, eli häiriölle (`ripple = 15`). Pyyhkäisytilakoneen toiminta aloitetaan antamalla pyyhkäisyn aloituspyyntö (`self.mod.sweepreq = not self.mod.sweepack`). Testifunktio on esiteltynä alla olevassa listauksessa (listaus 15).

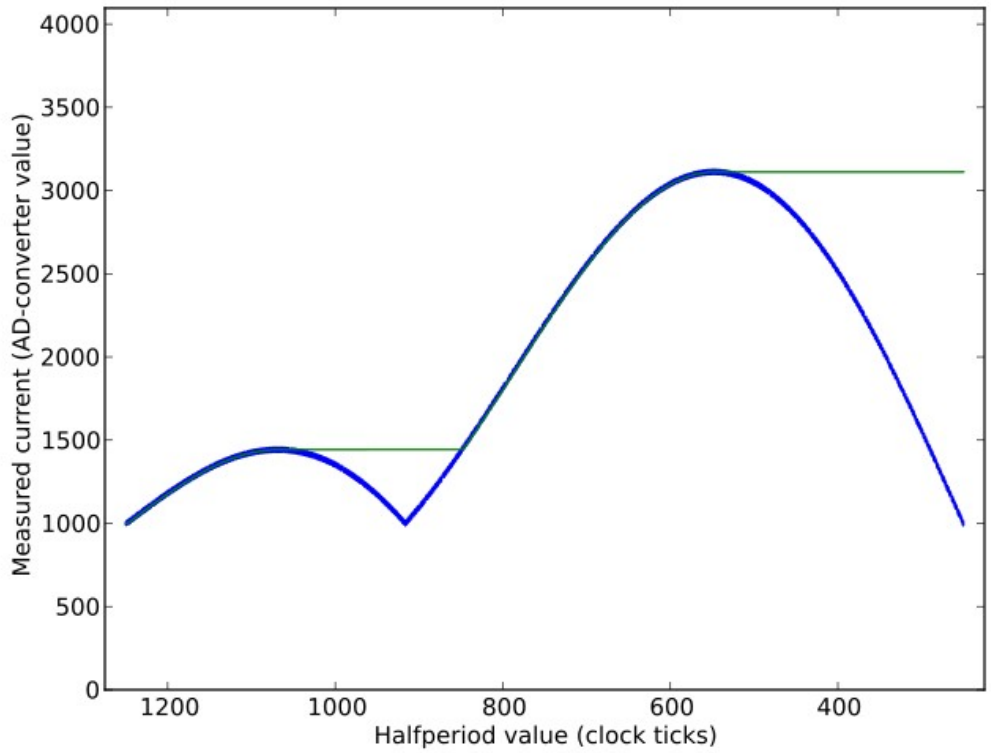
```
def TestHalfperiodSweep(self):
      """ Test logic module sweep state machine """
      values_in = []
      values_out = []
      self.mod.halfperiod = 750
      self.mod.sweepreq = not self.mod.sweepack # Generate sweep request
      self.Tick(3) # Run a few clock cycles to get state machine going
      max_x = 1000 # Start at 1000 because current never goes below this
      ripple = 15
      for loops in xrange(1000):
            f = float(loops) / 999.0 * math.pi
            x = int(1000 + 1500 * 0.6 * abs(math.sin(f) + math.sin(2 * f - math.pi)))
            max_x = max(x, max_x)
            for t in xrange(300):
                  y = x + random.randint(-ripple, ripple)
                  self.mod.trfmcurr = y
                  self.Tick()
                  values_in.append(y)
                  values_out.append(self.mod.maxcurrent or 1000)
            self.Tick(2)
            assert abs(self.mod.maxcurrent - max_x) <= 4, "Found maximum current too far
from average."
            self.Tick()
            Out(".")
      # Plot results
      p.plot(p.arange(250, 1250, 1. / 300.), values_in)
      p.plot(p.arange(250, 1250, 1. / 300.), values_out)
      p.xlabel("Halfperiod value (clock ticks)")
      p.ylabel("Measured current (AD-converter value)")
      p.savefig("sweep.svg", dpi = 150)
      p.show()
```

LISTAUS 15. Pyyhkäisytilakoneen ja virtasuodattimen testi.

Testin päätoiminto on for-silmukan muodossa. Silmukassa lasketaan halutuilla arvoilla sinikäyrää (katso kaava (4) alla), johon satunnaisesti lisätään jokaiseen arvoon pieni poikkeama. Sinikäyrässä on pienempi ja suurempi huippu. Lisättävä poikkeama voi olla -15 ja +15 välillä (`ripple`-arvo). Sinikäyrän maksimipiste haetaan muuttujaan `max_x` ja tätä vertaillaan suodatettuun virranarvoon ja ero näiden kahden arvon välillä saa olla korkeintaan neljä yksikköä, eikä suurempi ero kuulu suodattimen normaaliin toimintaan. Sinikäyrän arvot häiriöineen tallennetaan taulukkoon (`values_in`) ja mitatut puolijakson maksimivirran arvot omaan taulukkoonsa (`values_out`). Taulukoiden arvoista voidaan piirtää alla olevan kuvion (kuvio 7) mukainen käyrä.

$$D + A \cdot \left| \sin(\omega \cdot t) + \sin(2\omega \cdot t + \varphi) \right| \tag{4}$$

, missä $D = 1000,$ $A = 1500,$ $0 \leqslant \omega \cdot t \leqslant \pi$ ja $\varphi = -\pi$.



KUVIO 7. Virran arvo ja maksimiarvo puolijakson mukaan.

Kuvassa sininen käyrä kuvaa mitattavaa suodattamatonta sisääntulovirtaa. Virtakäyrässä esiintyy lukuisia pieniä häiriöitä, joita tässä kuvassa ei kovin hyvin näy suuren näytemäärän vuoksi. Vihreä käyrä kuvaa mitattua suodatettua suurinta ulostulovirtaa. Käyrä myötäilee sisääntulovirtaa suodattaen häiriöitä sisääntulovirrasta ja hakien virralle sen maksimiarvoa. Maksimiarvon löydettyä käyrä pysyy samassa arvossa kunnes se löytää uuden maksimiarvon sisääntulovirrasta. Tämä näkyy parhaiten sisääntulovirtakäyrän alku- ja loppupuoliskolla, jolloin virtakäyrä kääntyy hetkeksi laskuun ja lopussa kokonaan laskuun. Näissä kohdissa vihreä käyrä pysyy muuttumattomana edellisessä löydetyssä maksimiarvossa kunnes sisääntulovirtakäyrä lähtee taas nousemaan kohti uusia maksimiarvoja. Mitatun virtakäyrän huipusta voidaan lukea virran maksimiarvon y-akselista (Measured current) sekä siihen liittyvän maksimipuolijakson x-akselista (Halfperiod value).

Toinen testi (`TestChargingMode`) on yksinkertaisempi testi (listaus 16 alla), jolla varmistetaan akkujen latausmoodin tilan asetus. Akkujen lataustila voi olla automaatti-

sesti sallittu/kielletty, pakotettu päälle/pois tai pyyhkäisytilakoneen toimiessa aina sallittu. Mikäli tilakone ei ole saanut pyyhkäisypyyntöä, tarkistaa se akkujen lataustilan asetuksen. Tilan ollessa automaattinen (`enable_automode='1'`), välittää ohjelma akkujen latauksen sallivan/estävän `batterychargercontrol`-yksiköltä tuleva signaalin (`enablecharging`) ulostuloon (`tr_sdn`). Lataustilan ollessa pakotettu päälle (`enable_automode='0'`), välittää ohjelma ulostuloon käyttäjän määräämän sallivan/estävän bitin (`enable_forced`).

Testin ensimmäisessä silmukassa asetetaan satunnaisesti akkujen latauskäskyt: sallitaan, estetään, pakotetaan päälle tai pakotetaan pois. Lisäksi lataustila asetetaan myös satunnaisesti automaattiseksi tai manuaaliseksi. Seuraavaksi tarkistetaan mikäli lataustila on automaattinen, on ulostulon seurattava `enable`-signaalia. Mikäli lataustila on manuaalinen, on ulostulon vastaavasti oltava sama kuin `forced`-signaali.

Testin toisessa silmukassa asetetaan edellisen silmukan tavoin akkujen latauskäskyt sekä lataustilan. Seuraavaksi annetaan tilakoneelle pyyhkäisypyyntö (`self.mod.sweepreq = not self.mod.sweepack`) ja tämä on erona tämän ja edellisen silmukan välillä. Tilakoneen saatua pyyhkäisypyynnön, ei annetuilla latauskäskyillä ja -tilalla ole enää merkitystä, vaan ohjelma asettaa aina automaattisesti ulostuloon sallivan signaali (`tr_sdn = '1'`).

```
def TestChargingMode(self):
      """ Test charging mode (automatic, forced or sweep). """
      for i in xrange(1000):
            enable = random.choice([True, False])
            forced = random.choice([True, False])
            automode = random.choice([True, False])
            self.mod.enablecharging = enable
            self.mod.enable_forced = forced
            self.mod.enable_automode = automode
            self.Tick()
            if automode:
                  assert self.mod.tr_sdn == enable
            else:
                  assert self.mod.tr_sdn == forced
      for i in xrange(1000):
            enable = random.choice([True, False])
            forced = random.choice([True, False])
            automode = random.choice([True, False])
            self.mod.enablecharging = enable
            self.mod.enable_forced = forced
            self.mod.enable_automode = automode
            self.mod.sweepreq = not self.mod.sweepack
            self.Tick()
            assert self.mod.tr_sdn == True
```

LISTAUS 16. Lataustilojen asetustesti.

4.4.4 Avalon-testi

Avalon-kommunikoinnin testauksesta huolehtii *test_batterycharger_avalon.py*. Tällä halutaan varmistaa, että datan kirjoitus ja luku Avalon-väylän kautta onnistuu ongelmitta. Testit ovat varsin yksinkertaisia, mutta tärkeitä, sillä Avalon-väylä mahdollistaa VHDL-kirjastojen ja C-koodin välisen kommunikoinnin. Ensin kirjoitetaan dataa Avalon-väylän kautta rekisteriin ja luetaan kirjoitettu data Avalon-väylän kautta rekisteristä. Luettu data täytyy täsmätä kirjoitettuun dataan. Tällöin voidaan todeta että Avalon-kommunikointi on oikein tehty.

Testeissä Avalon-väylän kirjoitus ja luku tapahtuu `self.avalon` -Python-luokan avulla. Tämä luokka (`vsim.tools.AvalonRegisterMapping`) huolehtii automaattisesti Avalon-väylän signaalien asetuksista sekä luku- että kirjoitustilanteessa, ja se osaa myös automaattisesti valita oikean toiminnan tilanteesta riippuen. Luokka on suunniteltu siten, että lukutilanteessa asetetaan signaalit `_cs`, `_begintransfer` ja `_read` aktiivisiksi (tilaan `'1'`) ja `_address`-signaalille haluttu rekisterin osoite.

Tämä vastaa aiemmin esitellyssä listauksessa (listaus 4) olevaa tilannetta. Tämän jälkeen tuotetaan yksi kellojakso ja luetaan vastaus `_readdata`-signaalista.

Kirjoitustilanne on lähes identtinen. Myös siinä asetetaan `_cs`, `_begintransfer` ja `_write` aktiivisiksi (tilaan `'1'`) ja `_address`-signaalille haluttu osoite. Näiden lisäksi asetataan kirjoitettava data `_writedata`-signaaliin. Tilanne on vastaava kuin aiemmin esitelty Avalon-väylän osoitteen kirjoittaminen (listaus 5). Jälleen ajetaan yksi kellojakso ja kirjoitus on näin saatu valmiiksi.

Avalon-testimoduuli sisältää viisi yksinkertaista testiä:

- `TestWriteReadHalfperiod`
- `TestWriteReadMode`
- `TestWriteReadSweep`
- `TestReadSweepMaxHalfperiod`
- `TestReadMaxCurrent`

Ensimmäisessä testissä (`TestWriteReadHalfperiod`) testataan puolijakson kirjoitus ja luku (listaus 17). Ensin tarkistetaan puolijakson oletustilaa alussa, jonka jälkeen muutetaan puolijakson arvoa satunnaisesti sallituissa rajoissa (0-50000000) kirjoittamalla se Avalon-kirjoitusrekisteriin (`self.avalon.WriteHalfperiod`) ja luetaan kirjoitettu puolijakson arvoa Avalon-lukurekisterillä (`self.avalon.ReadHalfperiod`).

```
def TestWriteReadHalfperiod(self):
      """ Test write and read halfperiod. """
      defaultcounter = (self.mod.SYSTEMCLOCKFREQUENCY/2/self.mod.DEFAULTFREQUENCY) - 1
      assert self.avalon.ReadHalfperiod == defaultcounter
      assert self.mod.halfperiod == defaultcounter
      self.Tick()
      for i in xrange(1000):
            halfperiod = random.randint(0, 50000000)
            self.avalon.WriteHalfperiod = halfperiod
            self.Tick()
            assert self.avalon.ReadHalfperiod == halfperiod
            assert self.mod.halfperiod == halfperiod
```

LISTAUS 17. Puolijakson kirjoitus- ja lukutesti.

Seuraavassa testissä (`TestWriteReadMode`), listattuna alla (listaus 18), testataan käyttömoodin asetus. Avalonin kautta asetetaan akkujen latausmoodi satunnaisesti päälle, pois tai automaatti-tilaan. Tämän jälkeen tarkistetaan latauksen tila lukemalla Avalon-väylän kautta hetki sitten kirjoitettu tila. Lisäksi tarkastellaan suoraan Avalon-komponentin ulostuloista onko lataus pakotettu käyntiin vai onko lataus automaatti-tilassa.

```
def TestWriteReadMode(self):
      """ Test write and read mode. """
      for i in xrange(1000):
            mode = random.choice([0, 1, 2])
            self.avalon.WriteMode = mode
            self.Tick()
            if mode == 0:
                  assert self.avalon.ReadMode == 0
                  assert self.mod.enable_forced == 0
                  assert self.mod.enable_automode == 0
            if mode == 1:
                  assert self.avalon.ReadMode == 1
                  assert self.mod.enable_forced == 1
                  assert self.mod.enable_automode == 0
            if mode == 2:
                  assert self.avalon.ReadMode == 2
                  assert self.mod.enable_forced == 0
                  assert self.mod.enable_automode == 1
```

LISTAUS 18. Latausmoodin asetustesti.

Testillä `TestWriteReadSweep` (listaus 19) tarkistetaan pyyhkäisytila alussa, joka on oltava `'0'` eli epätosi. Tämän jälkeen annetaan pyyhkäisyn aloituspyyntö kirjoittamalla mikä tahansa luku (tässä tapauksessa 1) pyyhkäisyn aloituspyynnön rekisterin osoitteeseen (`self.avalon.WriteSweep = 1`). Jokaisen pyynnön jälkeen pyyhkäisytila on oltava eri kuin ennen pyynnön antamista.

```
def TestWriteReadSweep(self):
      """ Test write and read sweep. """
      assert self.avalon.ReadSweep == False
      for i in xrange(1000):
            self.mod.sweepack = random.choice([True, False])
            self.avalon.WriteSweep = 1
            self.Tick()
            assert self.mod.sweepreq == (not self.mod.sweepack)
```

LISTAUS 19. Pyyhkäisytilan testi.

Maksimipuolijakson lukutesti `TestReadSweepMaxHalfperiod` (listaus 20) testaa yksinkertaista signaaliarvon lukemista Avalon-väylän kautta. Maksimipuolijakson arvoksi (`self.mod.maxhalfperiod`) asetetaan satunnaisesti sallituissa rajoissa ar-

vottu luku. Asetettu luku luetaan Avalon-väylän lukurekisterillä (`self.avalon.ReadSweepMaxHalfperiod`) ja verrataan luettua tulosta aiemmin asetettuun arvoon. Näiden kahden luvun on vastattava toisiaan. Viimeinen testi `TestReadMaxCurrent` on tehty ja toimii täysin samalla periaatteella.

```
def TestReadSweepMaxHalfperiod(self):
      """ Test reading sweep max halfperiod. """
      for i in xrange(1000):
            maxhalfperiod = random.randint(0, 2047)
            self.mod.maxhalfperiod = maxhalfperiod
            self.Tick()
            assert self.avalon.ReadSweepMaxHalfperiod == maxhalfperiod

def TestReadMaxCurrent(self):
      """Test reading max current. """
      for i in xrange(1000):
            maxcurrent = random.randint(0, 4095)
            self.mod.maxcurrent = maxcurrent
            self.Tick()
                  assert self.avalon.ReadSweepMaxCurrent == maxcurrent
```

LISTAUS 20. Maksimipuolijakson ja maksimivirran lukutestit.

4.4.5 Batterycharger-testi

Edellisillä testeillä on voitu varmistaa vain yksittäisten VHDL-yksiköiden toimintaa. Seuraavaksi testataan kaikkien komponenttien yhteistoiminta. Testillä halutaan varmistaa, että kaikki kirjaston ohjelmayksiköt toimivat oikein yhdessä. Testi toteutetaan pääkomponentille tehdylle *wrapper*-tiedostolle. Pääyksikön kautta tiedot on vietävä kaikille muille komponenteille. Näin toimii ohjelma kokonaisuudessaan todellisuudessakin. Avalon-väylän rekisteillä kirjoitetaan sisäänmenot, jotka välittyvät ylimmän tason pääyksikön (`batterycharger`), eli *top level entity*n, kautta niitä käyttäville yksiköille. Myös ulostulot tulevat pääkomponentin kautta Avalon-lukurekistereille käyttäjän luettavaksi. Pääyksiköllä on itsellään muutamia sisään- ja ulostuloja, jotka välittyvät suoraan muille komponentille ilman Avalon-kommunikointia.

Yhdistetty testi sisältää kaksi testiä, jolla testataan ensin virransuodatusta ja tilakoneen toimintaa (`TestHalfperiodSweep`). Testi on käytännössä samanlainen kuin logic-komponentin testauksessakin, tosin sillä erotuksella että nyt välituloksia ei kerätä erilli-

siin taulukoihin eikä kuvaajaa näinollen piirretä. Seuraava testi on yksinkertaisempi pyyhkäisyn aloituspyyntötesti (`TestWriteReadSweep`), jolla testataan pyyhkäisyn aloituksen pyytäminen Avalon-rekisterillä. Viimeisellä testillä (`TestWriteReadMode`) testataan akkujen latausmoodin asetuksen onnistumista Avalon-rekisterin avulla ja asetetun tilan luku sekä Avalon-rekisterin kautta, että suoraan pääyksikön ulostuloista. Kaikki testit löytyvät kokonaisuudessaan dokumentin lopusta liitteestä 2.

Testit ovat hyvin pitkälle varsin samoja kuin yksittäisten yksiköiden testauksessa käytetyt testit. Tässä yhdistetyssä testissä suurimpana erona on se, että yksittäisen yksiköiden signaaleja ei voida suoraan hyödyntää testissä, vaan signaalien käyttömahdollisuudet rajoittuvat pääyksikön signaaleihin sekä Avalon-rekistereihin. Koska testit muistuttavat pääosin jo edellä esitettyjä testejä, ei tässä osiossa esitellä tarkemmin kaikkia pääyksikön testejä. Alla olevassa listauksessa (listaus 21) on esiteltynä virransuodatusta ja tilakoneen toimintaa käsittelevä testi.

```
def TestHalfperiodSweep(self):
      """ TestHalfperiodSweep """
      assert not self.avalon.ReadSweep
      self.mod.enablecharging = 0
      self.avalon.WriteMode = 2
      self.Tick()
      assert self.avalon.ReadMode == 2
      assert self.mod.tr_sdn == 0
      self.avalon.WriteSweep = 1
      self.Tick()
      assert self.avalon.ReadSweep
      assert self.mod.tr_sdn == 1
      assert self.avalon.ReadMode == 2
      max_x = 1000 # Start at 1000 because current never goes below this
      ripple = 15
      for loops in xrange(1000):
            f = float(loops) / 999.0 * math.pi
            x = int(1000 + 1500 * 0.6 * abs(math.sin(f) + math.sin(2 * f - math.pi)))
            max_x = max(x, max_x)
            for t in xrange(300):
                  y = x + random.randint(-ripple, ripple)
                  self.mod.trfmcurr = y
                  self.Tick()
            self.Tick(2)
            maxcurrent = self.avalon.ReadSweepMaxCurrent
            assert abs(maxcurrent - max_x) <= 4, "Found maximum current too far from
average."
      print self.avalon.ReadSweepMaxHalfperiod
      print self.avalon.ReadSweep
```

LISTAUS 21. Päätason tilakoneen ja virransuodatuksen testi.

4.4.6 Makefile

Tiedosto *makefile* eli niin sanottu käännösohjetiedosto, sisältää testitiedostojen kääntämistä ja ajoa varten tarvittavat tiedot. Sisällytettävät tiedot ovat muun muassa polku testattaviin VHDL-tiedostoihin, ja vsim-hakemistoon, testiin kuuluvien lähdetiedostojen nimet, käännöskäskyjen määrittelyt sekä testitiedostojen nimet. *Makefile*-tiedoston avulla testit voidaan käynnistää ja ajaa yksinkertaisella komennolla ”`make test`” ja testissä syntyneet väliaikaiset käännöstiedostot voidaan poistaa ”`make clean`”-käskyllä. Nämä käskyt on määritelty makefile-tiedoston lopussa. *Makefile* ei ole pakollinen osa testauksen kannalta, mutta se helpottaa testien kääntämistä. Ilman tätä tiedostoa käännös ja testien suorittaminen sekä muut määritykset olisi tehtävä manuaalisesti. Esimerkkinä on yksinkertaisempi `batterychargercontrol`-yksikön *makefile* (listaus 22).

```
# ============================================================================
VHDLPATH=../../vhdllib/batterychargercontrol
VSIMPATH=../vsim
GHDLARGS=-g --workdir=work --warn-unused --warn-binding --warn-body --warn-specs --warn-
error
VHDLPP=../../bin/vhdlpp
ENTITYSRCFILE=./batterychargercontrol.vhd
SRCFILES=$(ENTITYSRCFILE)
PIPENAME=$(basename $(notdir $(ENTITYSRCFILE)))
INPUTPIPE=/tmp/$(PIPENAME)_inputpipe
OUTPUTPIPE=/tmp/$(PIPENAME)_outputpipe
clean: defaultclean
test:
      +make clean
      +make compile
      INPUTPIPE=$(INPUTPIPE) OUTPUTPIPE=$(OUTPUTPIPE) PYTHONPATH=$(VSIMPATH)/.. python
-B    test_batterychargercontrol.py
include ../vsim/vsim.mk
# ============================================================================
```

LISTAUS 22. Batterychargercontrol-yksikön makefile.

4.4.7 Batterychargercontrol-testi

Python-tiedosto, *test_batterychargercontrol.py*, on hyvin yksinkertainen testimoduuli, jolla testataan *batterychargercontrol.vhd* toimivuutta. Testissä `batterychargercontrol`-yksikön sisääntuloihin laitetaan `sensorstate`- ja `sensorerror` -tiloiksi vuorotellen kaikki mahdolliset vaihtoehdot ja tarkastellaan ulostuloa. Mikäli ulos-

tulot täsmää oikeisiin tuloksiin, voidaan todeta että batterychargercontrol-yksikkö toimii tarkoituksen mukaisesti.

Testitiedosto sisältää kaksi pientä testiä, TestSensorsError ja TestSensorsNotError. Molemmat testit on esiteltynä alla (listaus 23). Ensimmäisellä pyritään testaamaan ulostuloa, kun errostate-signaalin tila on vähintään toisessa AMR-sensorissa päällä (tila 1). Tässä tapauksessa ulostulon on oltava kielteinen latauskäsky, riippumatta sensorin toiminnan, sensorstate-signaalin, tilasta.

Jälkimmäisellä testillä, TestSensorsNotError, testataan ulostuloa kahdessa eri tapauksessa. Signaalin, sensorerror, ollessa 0 eli virhettä ei esiinny kummassakaan sensorissa, mutta sensorien tila, sensorstate, on päällä (1) korkeintaan vain toisessa sensorissa. Tällöin ulostulon on oltava kielteinen latauskäsky. Toisessa tapauksessa batterychargercontrol sallii latauksen, kun virhettä ei esiinny kummassakaan sensorissa ja molempien sensorien toimintatila, sensorstate, on päällä (1).

```
def TestSensorsError(self):
      """ Test sensors error, charging shoul not be enabled. """
      for i in xrange(10):
            for x in xrange(100):
                  errorstate = random.choice([2, 4, 6])
                  self.mod.sensorerror = errorstate
                  self.mod.sensorstate = 0
                  self.Tick()
                  assert self.mod.enablecharging == 0
                  self.mod.sensorstate = 1
                  self.Tick()
                  assert self.mod.enablecharging == 0
                  self.mod.sensorstate = 2
                  self.Tick()
                  assert self.mod.enablecharging == 0
                  self.mod.sensorstate = 6
                  self.Tick()
                  assert self.mod.enablecharging == 0

def TestSensorsNotError(self):
      """ Test sensors not error, charging should be enabled. """
      for i in xrange(10):
            for x in xrange(100):
                  self.mod.sensorerror = 0
                  sensorstate = random.choice([0, 2, 4, 6])
                  self.mod.sensorstate = sensorstate
                  self.Tick()
                  if sensorstate == 6:
                        assert self.mod.enablecharging == 1
                  else:
                        assert self.mod.enablecharging == 0
```

LISTAUS 23. Ohjausyksikön testit.

4.5 NIOS-kirjasto

Edellä esitelty laitteisto, siihen liittyvä VHDL-kirjasto Avalon-väylöineen sekä sen testaus ovat jo itsessään varsin kattava kokoelma elektroniikkaa, ohjausta, kommunikointia ja simulointia. Mutta yksistään pelkkä VHDL-komponenttikokonaisuus ei ole vielä kovin käytännöllinen, vaan sitä ohjaamaan tarvitaan korkeamman tason C-kielistä logiikkaa. Lisäksi pelkkä Avalon-väylä yksinään ei ole vielä erityisen käytännöllinen kaikkine osoitteineen ja signaaleineen, vaan tarvitaan jonkinlainen selkeä tapa, tässä tapauksessa ohjelmakirjasto, kommunikoida korkean tason C-kielen ja VHDL-komponentin välillä.

Tähän on vastauksena NIOS-kirjasto. Nimitys kirjastolle tulee FPGA:lla toteutetuista, puhtaasti logiikkakomponenteista koostetuista NIOS II -ohjelmistoprosessoreista [NII5V1-11.0]. NIOS-kirjastot ovat yksinkertaisimmillaan “vastakappale” VHDL-kirjaston Avalon-komponentille, joka pitää sisällään tarpeelliset funktiokutsut yksinkertaista kommunikointia varten. Monipuolisimmillaan NIOS-kirjastot osaavat hoitaa esimerkiksi kokonaisen UDP/IP/Ethernet -pinon toiminnan tai radioverkon yli tapahtuvan kommuninkoinnin yhteydessä VHDL-kirjastojen kanssa. Batterycharger-kirjaston tapauksessa kyseessä on melko yksinkertainen kirjasto jonka päätehtävänä on kommunikointi VHDL-kirjaston kanssa sen Avalon-komponentin mukaisesti.

4.5.1 Kirjaston tiedostorakenne

Kaikki NIOS-kirjastot on kirjoitettu puhtaalla C-kielellä ja ne koostuvat hyvin pitkälle saman reseptin mukaisesti. Tämä tarkoittaa käytännössä kolmea tiedostoa jotka on nimetty seuraavasti:

- *kirjastonimi*.c – Kirjaston funktioiden toteutustiedosto
- *kirjastonimi*.h – Kirjaston funktioiden esittelytiedosto

- *kirjastonimi*_inline.h – Kirjaston struktuurien, tyyppimääritteiden, sisäisten määritteiden ja inline-funktioiden toteutustiedosto. Tämä tiedosto ei aina ole välttämätön.

Näiden lisäksi useat kirjastot on jaettu useampaan tiedostoon rakenteen selkeyden ja modulaarisuuden vuoksi. Esimerkiksi laajemmat, vain kirjaston sisäiseen käyttöön tarkoitetut, staattiset funktiot määritellään usein *kirjastonimi*_internal.h -nimiseen tiedostoon. Lisäksi käytössä saattaa olla erilaisia laajempia määrittely- ja konfiguraatiotiedostoja. Myös itse koodien toteutukset saattavat olla jaettuna useampaan .c -tiedostoon. Olipa tiedostoja kuinka monta tahansa, alkaa niiden nimi joka tapauksessa aina kirjaston nimellä. Tällä varmistetaan omalta osaltaan tiedostojen yksilöllisyys.

4.5.2 Funktioiden nimeämiset

Sen lisäksi että kaikki yhteen kirjastoon kuuluvat tiedostot nimetään aina saman kaavan mukaan, myös kaikki kirjaston ulkopuolelta käytettävät funktiot nimetään yhtenäisesti. Jokainen funktionimi alkaa aina isolla kirjoitetulla `KIRJASTONIMI`-alkuliitteellä. Näin voidaan varmistua siitä, ettei vahingossakaan pääse syntymään nimiavaruuden ristiriitoja. Lisäksi tällä rakenteella pyritään mallintamaan C++ -kielen luokkatyyppisiä rakenteita (vertaa `Luokannimi::` nimiavaruusmäärittely), mutta kuitenkin ilman C++ -kielen mukanaan tuomaa ylimääräistä taakkaa. Tällä ylimääräisellä taakalla tarkoitetaan C++ -kielen vaatimaa hieman suurempaa muistimäärää per luokan instanssi sekä hieman monimutkaisempaa kutsurakennetta.

Yhtenäinen funktioiden nimeämistapa myös kertoo suoraan ja helposti että mistä kirjastosta on kyse. Lähes jokainen kirjasto liittyy läheisesti vastaavan nimiseen VHDL-kirjastoon, joten käyttäjä pystyy helposti löytämään molempien osien koodit ja vertaamaan niiden toimintaa. Samanlaiset nimeämiskäytännöt sekä NIOS- että VHDL-kirjastoissa tuo koodin luettavuuden ja helpon käytettävyyden lisäksi myös erittäin selkeää modulaarisuutta ja uudelleenkäytettävyyttä.

4.5.3 Sisäinen rakenne

Alteran Quartus-käännösympäristössä jokaiselle yksittäiselle Avalon-moduulille määritellään automaattisesti (tai käyttäjän halutessa myös manuaalisesti) niin kutsuttu pohjaeli perusosoite (Base Address). Nämä määrittelyt, ja niihin liittyvä järjestelmän määrittelytiedosto, luodaan automaattisesti projektin käännösvaiheessa ja ne löytyvät järjestelmän kuvaustiedostosta nimeltään *system.h*. Kuten jo edellä mainittiin, suuri osa NIOS-kirjastoista liittyvät läheisesti vastaaviin VHDL-kirjastoihin ja erityisesti niiden Avalon-moduuleihin. Tästä seuraa se, että NIOS-kirjastot jotka ovat läheisesti tekemisissä vastaavien VHDL-kirjastojen kanssa tarvitsevat tiedon näistä perusosoitteista, ja sen vuoksi kaikki tällaiset NIOS-kirjastot määrittelevät oman struktuurinsa (C-kielen `struct`). Käytännössä ensimmäinen asia jokaisessa luokan sisäisessä struktuurissa onkin nimenomaan tämä perusosoite. Asiasta seuraa lisää tarkennusta hieman myöhemmin.

Aiemmin selitettiin jo hieman yhtäläisyyksiä näiden puhtaasti C-kielisten kirjastojen ja vastaavien C++ -kielen luokkarakenteiden välillä. Molemmissa löytyy nippu julkisia ja piilotettuja funktioita, ja molemmissa data myös “piilotetaan” (kapseloidaan) luokan sisälle ilman että luokan käyttäjällä olisi varsinaista suoraa pääsyä siihen. C-kielessä tällainen aito piilottaminen ei ole mahdollista, mutta sitä voidaan tehokkaasti mallintaa struktuurien (`struct`) avulla, varsinkin kun struktuurin varsinainen toteutus on erillisessä tiedostossa. Kirjaston käyttäjälle ainoa tarpeellinen tiedosto sen määrittelytiedosto, eli .h -tiedosto. Alla on esimerkkilistaus (listaus 24) yksinkertaisesta NIOS-kirjaston sisäisestä struktuurista.

```
struct MYLIBRARY {
  uint32_t baseaddr; // VHDL module base address

  uint32_t internalstate; // Variable to hold internal library state
  uint8_t data[64]; // Array for data values
};
```

LISTAUS 24. Esimerkki NIOS-kirjaston sisäisestä struktuurista.

Kuten yllä olevasta esimerkistä nähdään, on “luokan” piilotetun datarakenteen ensimmäinen jäsen nimenomaisesti edellä mainittu perusosoite, joka on tyypiltään `uint32_t` eli 32-bittinen etumerkitön kokonaisluku. Sen lisäksi datarakenteesta löytyy

kaks muuta arvoa joita kirjasto voi käyttää tallentamaan erilaisia tarpeellisia tietoja. Osa kirjastoista ei välttämättä sisällä perusosoitteen lisäksi mitään muuta dataa, ja osa taas sisältää suuret määrät erilaisia tietoja. C++ -tyylisesti lähes jokaisesta kirjastosta voi olla olemassa monta ilmentymää (instanssia) joilla kaikilla on käytössä luokan samat funktiot mutta omat struktuurinsa, yleensä vähintään eri perusosoitteilla varustettuna.

Yhtäläisyydet C++ -kielen kanssa jatkuvat edelleen datan piilotuksen lisäksi. Kuten C++ -kielessä luokalla on yksi tai useampi rakentaja (constructor), löytyy myös NIOS-kirjastoista yksi tai useampi alustusfunktio. Nämä funktiot nimetään pääsääntöisesti aina `KIRJASTONNIMIInit` -mallisesti. Parametrien tyypit ja arvot vaihtelevat aina käyttökohteesta riippuen, mutta tyypillisin erimerkki on alla olevassa listauksessa b. Näiden alustusfunktioiden perimmäisenä tarkoituksena on alustaa kirjaston sisäisen struktuurin arvot sopiviksi. Ne voivat tehdä myös muita tarpeellisia alustustoimenpiteitä kuten esimerkiksi NIOS-kirjastoon liittyvän VHDL-kirjaston sisäisten rekisterien alustus.

```
void MYLIBARYInit( MYLIBRARY* mylib, uint32_t baseaddr ) {
  // Initialize structure memory
  memset( (void*) mylib, 0, sizeof(MYLIBARY) );

  // Set base address
  mylib->baseaddr = baseaddr;
}
```

LISTAUS 25. Esimerkki NIOS-kirjaston alustusfunktiosta.

Edellä oleva esimerkki (listaus 25) on erittäin tyypillinen NIOS-kirjaston alustusfunktio, parametreja myöten. Tämä rakenne vastaa hyvin pitkälti C++ -kielen rakentajafunktiota jonka parametrinä on `baseaddr` -muuttuja. Tällainen alustusfunktio nollaa kirjaston struktuurin muistialueen ja asettaa sen jälkeen perusosoitteen talteen samaiseen piilotettuun datarakenteeseen. Todellisuudessa edellä esitetty rakenne on käytännössä identtinen C++ -kielen kanssa, sillä funktiokutsun ensimmäinen parametri on osoitin luokan sisäiseen struktuuriin. Sama rakenne on käytössä C++ -kielessä, ainoastaan sillä erotuksella, että kyseinen parametri (vertaa `this` -avainsanaan) on automaattinen ja käyttäjältä piilotettuna.

Rakentaja- eli alustusfunktion lisäksi kirjastolla voi mahdollisesti olla myös C++ -kielen tapaisesti hajottajafunktio (destructor), mikäli sellaista tarvitaan esimerkiksi vapauttamaan dynaamisesti varattua muistia tai muita kirjaston varaamia resursseja. Koska tällaisten funktioiden tarve on harva, ei niille ole sovittua yhteistä nimeämistapaa, mutta esimerkiksi *KIRJASTONNIMI*`Uninit` ja *KIRJASTONNIMI*`Destroy` ovat käytössä muutamassa kirjastossa.

Ollakseen käyttökelpoisia, täytyy kirjastojen tietysti sisältää muitakin funktioita kun pelkkiä rakentaja- ja hajottajafunktioita. Näistä yksinkertaisimpina esimerkkeinä on erilaiset niin kutsutut getter/setter -funktiot, joilla joko haetaan luokan jonkin sisäisen muuttujan arvo, tai vaihtoehtoisesti asetetaan luokan sisäiselle muuttujalle joku arvo. Tällaiset funktiot esitellään C-kielessä yleensä inline-funktioiksi, eli käännösvaiheessa ne korvataan (tavallisesti) kopioilla funktion koodista. Tämä tehdään siksi, että funktiot olisivat sekä nopeita kutsua, että myös tilaa säästäviä, kuitenkaan päästämättä käyttäjää varsinaisesti ”suoraan” käsittelemään kirjaston struktuurin sisäisiä arvoja.

Luonnollisesti kirjastoilla on normaalisti myös laajempia toiminnallisia funktioita, joiden nimet ja parametrit esitellään .h-tiedostossa ja toteutus .c-tiedostossa. Esimerkiksi kaikki alustusfunktiot ovat tällaisia, vaikka niiden sisäinen toiminta saattaakin olla suppea. Pääsääntöisesti hyvin lyhyet (1-2 riviä) funktiot tehdään inline-funktioina ja kaikki vähänkin laajemmat funktiot tavallisina funktioina. Seuraavalla sivulla olevat kolme listausta (listaus 26, listaus 27 ja listaus 28) näyttävät esimerkin kaikista edellä luetelluista rakenteista ja funktioista. Niistä selviää myös tarkemmin kirjaston varsinainen koko rakenne.

```
#ifndef MYLIBRARY_H
#define MYLIBRARY_H

#include <stdint.h>

// Internal structure
struct MYLIBRARY;
typedef struct MYLIBRARY MYLIBRARY;
// Initializer function
void MYLIBRARYInit( MYLIBRARY* mylib, uint32_t baseaddr );
// Inline getter and setter functions
static inline uint32_t MYLIBRARYGetState( MYLIBRARY* mylib );
static inline void MYLIBRARYSetState( MYLIBRARY* mylib, uint32_t newstate );
// Communications function for VHDL library
void MYLIBRARYFetchData( MYLIBRARY* mylib );
// Include last!
#include ”mylibrary/mylibrary_inline.h”

#endif // MYLIBRARY_H
```

LISTAUS 26. Esimerkkikirjaston mylibrary.h esittelytiedosto.

```
#ifndef MYLIBRARY_INLINE_H
#define MYLIBRARY_INLINE_H

#include <stdint.h>
#include <io.h>

// Internal structure contents
struct MYLIBRARY {
  uint32_t baseaddr; // VHDL module base address
  uint32_t internalstate; // Variable to hold internal library state
  uint8_t data[64]; // Array for data values
};

static inline uint32_t MYLIBRARYGetState( MYLIBRARY* mylib ) {
  return mylib->internalstate;
}
static inline void MYLIBRARYSetState( MYLIBRARY* mylib, uint32_t newstate ) {
   mylib->internalstate = newstate;
}

#endif // MYLIBRARY_INLINE_H
```

LISTAUS 27. Esimerkkikirjaston mylibrary_inline.h inline-funktioiden toteutustiedosto.

```
#include <string.h>
#include <stdint.h>

#include ”mylibrary/mylibrary.h”

void MYLIBARYInit( MYLIBRARY* mylib, uint32_t baseaddr ) {
  // Initialize structure memory
  memset( (void*) mylib, 0, sizeof(MYLIBARY) );
  // Set base address
  mylib->baseaddr = baseaddr;
}

void MYLIBRARYFetchData( MYLIBRARY* mylib ) {
  int i;
  uint8_t tmp;
  for( i = 0; i < 64; i++ ) {
      tmp = IORD( mylib->baseaddr, i ); // Get data from VHDL module
      // Process here if necessary
      mydata[i] = tmp; // Save result
  }
}
```

LISTAUS 28. Esimerkkikirjaston mylibrary.c funktioiden toteutustiedosto.

Edellä esitellyt kolme listausta ovat varsin yksinkertainen esimerkki tyypillisestä NIOS-kirjaston toteutuksesta, mutta siitä voidaan hyvin nähdä miten C-kielellä voidaan toteuttaa käytännössä C++ -tyyppinen luokka. Sisäinen struktuuri esitellään *mylibrary.h* -tiedostossa ainoastaan nimeltä, eikä kirjaston käyttäjän muuta tarvitse tietääkään, eli käyttäjällä ei ole tietoa luokan sisäisistä ”piilotetuista” tietorakenteista. Tiedosto *mylibrary_inline.h* liitetään (`#include`) *mylibrary.h* -tiedostoon vasta lopuksi, ja näin kääntäjä saadaan kuitenkin tietoiseksi kaikesta sen tarvitsemista asioista. Kolmannessa listauksessa (listaus 28) on eräs Alteran omista funktioista, Avalon-väylän kommunikoinnissa käytetty `IORD`-funktio.

4.5.4 Batterycharger NIOS-kirjasto

Edellä esitelty rakenne on käytännössä suoraan sellaisenaan käytössä tarkasteltavassa *batterycharger* NIOS-kirjastossa, tosin siinä funktioita on huomattavasti enemmän. Toisaalta sen tietorakenne on edellä mainittu yksinkertaisin esimerkki, eli se sisältää ainoastaan VHDL-kirjaston Avalon-moduulin perusosoitteen. Muuta tietoa ei tässä tapauksessa tarvitse sisäisesti säilyttää, vaan ne ovat VHDL-kirjaston sisäisissä rekistereissä, joista ne saadaan haluttaessa luettua ja kirjoitettua nopeasti. Kirjaston esittelytiedoston alku on kuvattuna alla olevassa listauksessa (listaus 29). Koodi kokonaisuudessaan löytyy dokumentin lopusta liitteestä 3.

```
//
// batterycharger.h
// Battery charger module for using gripper board's light curtain battery charger
//
#include <stdint.h>
//---------------------------------------------------------------------------
#ifndef BATTERYCHARGER_H
#define BATTERYCHARGER_H
//---------------------------------------------------------------------------
struct BATTERYCHARGER;
typedef struct BATTERYCHARGER BATTERYCHARGER;

// Initialize interface.
void BATTERYCHARGERInit(BATTERYCHARGER *batterycharger,
                        uint32_t baseaddr); // Base address of VHDL-module

//---------------------------------------------------------------------------
// Enable batterycharger
static inline void BATTERYCHARGERSetModeForceOn(BATTERYCHARGER *batterycharger);

//---------------------------------------------------------------------------
// Disable batterycharger
static inline void BATTERYCHARGERSetModeForceOff(BATTERYCHARGER *batterycharger);

//---------------------------------------------------------------------------
// Set batterycharger to automatic mode
static inline void BATTERYCHARGERSetModeAuto(BATTERYCHARGER *batterycharger);
```

LISTAUS 29. NIOS-kirjasto batterycharger.h esittelytiedoston alku.

Yllä olevasta listauksesta nähdään suoraan yhtäläisyys aiemmin esiteltyyn ja selitettyyn esimerkkikirjastoon. Ainoastaan alustusfunktio (`BATTERYCHARGERInit`) on normaali funktio ja loput funktiot ovat hyvin lyhyitä inline-funktioita. Alla (listaus 30) on esitelty kokonaisuudessaan funktioiden toteutustiedosto *batterycharger.c*. Tämä on käytännössä identtinen esimerkkikirjaston toteutustiedoston kanssa, eli siinä aluksi nollataan kirjaston sisäinen struktuuri ja sen jälkeen laitetaan perusosoite muistiin.

```
//
// batterycharger.c
// Battery charger module for using gripper board's light curtain battery charger
//
#include <string.h>
#include "batterycharger/batterycharger.h"
//---------------------------------------------------------------------------
// Initialize interface.
void BATTERYCHARGERInit(BATTERYCHARGER *batterycharger,
                        uint32_t baseaddr) { // Base address of VHDL-module
  memset( batterycharger, 0, sizeof(*batterycharger) );
  batterycharger->baseaddr = baseaddr;
}
//---------------------------------------------------------------------------
```

LISTAUS 30. NIOS-kirjasto batterycharger.c funktioiden toteutustiedosto.

Viimeisenä alla olevassa listauksessa (listaus 31) on tämän NIOS-kirjaston ”tärkein” tiedosto, eli tässä tapauksessa inline-funktioiden toteutustiedosto *batterycharger_inline.h*. Lähes kaikki kirjaston toiminta toteutetaan erittäin lyhyinä, vain yhden rivin mittaisina funktioina, ja ovat tästä nimenomaisesta syystä inline-funktioita.

Uutena asiana tiedostossa ovat esikääntäjämäärittelyt, eli `#define` -lauseet. Nämä määrittelyt ovat identtisiä, osin nimityksiä lukuunottamatta, aiemmin esitellyn VHDL-kirjaston `constant` -osoitemäärittelyiden kanssa. Näin voidaan helposti verrata sekä VHDL-kirjaston että NIOS-kirjaston yhtäläisyyksiä ja varmistua osaltaan siitä että omituisia virheitä ei esiinny ja että kaikki toimii oikein. Näillä `#define` -määrittelyillä vältytään myös siltä, että koodin seasta joutuisi etsimään niin kutsuttuja ”taikanumeroita”, eli numeroita joiden selitys ei ole millään muotoa ilmeinen. Myös kirjaston muokkaaminen ja korjaaminen jälkikäteen on huomattavasti helpompaa kun kaikki kirjastoa koskevat oleelliset osoitteet, numerot ja muut vastaavat on kerätty yhteen paikkaan. Tätä ideaa olisi mahdollista viedä jopa pidemmälle sopivan esikäsittelijän kanssa, joka osaisi lukea osoitemääritteet esimerkiksi suoraan VHDL-koodista.

```
//
// batterycharger_inline.h
// Battery charger module for using gripper board's light curtain battery charger
//
#include <io.h>
#include "batterycharger/batterycharger.h"
//------------------------------------------------------------------------------
#ifndef BATTERYCHARGER_INLINE_H
#define BATTERYCHARGER_INLINE_H
//------------------------------------------------------------------------------
// Internals
//------------------------------------------------------------------------------
struct BATTERYCHARGER {
  uint32_t baseaddr;
};
//------------------------------------------------------------------------------
// Batterycharger register-addresses
#define BATTERYCHARGER_REG_RW_MODE             0 // Mode register
#define BATTERYCHARGER_REG_RW_HALFPERIOD       1 // Set/get halfperiod
#define BATTERYCHARGER_REG_RW_CALIB            2 // Start/query calibration status
#define BATTERYCHARGER_REG_R_CALIB_HALFPERIOD  3 // Get halfperiod after calibration
#define BATTERYCHARGER_REG_R_CALIB_CURRENT     4 // Get current after calibration

// Batterycharger mode register values
#define BATTERYCHARGER_REG_MODE_FORCEOFF  0 // Disable charger
#define BATTERYCHARGER_REG_MODE_FORCEON   1 // Enable charger
#define BATTERYCHARGER_REG_MODE_AUTO      2 // Set charger to automatic mode

//------------------------------------------------------------------------------
static inline uint32_t BATTERYCHARGER_ReadReg(BATTERYCHARGER *batterycharger,
                                              uint32_t reg) {
  return IORD(batterycharger->baseaddr,reg);
}
//------------------------------------------------------------------------------
static inline void BATTERYCHARGER_WriteReg(BATTERYCHARGER *batterycharger,
                                           uint32_t reg,
                                           uint32_t value) {
  IOWR(batterycharger->baseaddr,reg,value);
}
//------------------------------------------------------------------------------
// Public API
//------------------------------------------------------------------------------
// Force batterycharger on
static inline void BATTERYCHARGERSetModeForceOn(BATTERYCHARGER *batterycharger) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_MODE,
BATTERYCHARGER_REG_MODE_FORCEON);
}
//------------------------------------------------------------------------------
// Force batterycharger off
static inline void BATTERYCHARGERSetModeForceOff(BATTERYCHARGER *batterycharger) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_MODE,
BATTERYCHARGER_REG_MODE_FORCEOFF);
}
//------------------------------------------------------------------------------
// Set batterycharger to automatic mode
static inline void BATTERYCHARGERSetModeAuto(BATTERYCHARGER *batterycharger) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_MODE,
BATTERYCHARGER_REG_MODE_AUTO);
}
//------------------------------------------------------------------------------
```

LISTAUS 31. NIOS-kirjasto batterycharger_inline.h inline-funktioiden toteutustiedosto.

Yllä olevassa listausksessa (listaus 31) on alkuosaa inline-funktioiden toteutustiedostosta. Tiedosto kokonaisuudessaan löytyy lopusta liitteestä 3. Kuten jo aiemmin todettiin, on batterycharger-kirjaston sisäinen struktuuri erittäin pieni ja yksinkertainen, eli se sisältää ainoastaan VHDL-kirjaston perusosoitteen. Seuraavana on `#define` -määritteet VHDL-kirjaston eri osoitteille sekä arvot MODE-rekisterin arvoille. Kaikki nämä arvot

ovat niitä ”taikalukuja” joita koodia luettaessa pitäisi ilman nimeä muulla tavoin ymmärtää, ja niiden muokkaaminen ilman tarkempaa tietoa olisi vähintäänkin virhealtista.

Seuraavana on kaksi kappaletta kirjaston vain sisäiseen käyttöön tarkoitettuja funktioita, ja sitä selventämään käytetään funktioiden nimessä alaviiva (_). Näiden funktioiden tarkoituksena on jälleen selventää koodin lukemista ja erityisesti korjaamista/muokkaamista jälkikäteen. Kaikki Alteran Avalon-väylän liikenne tapahtuu käytössä olevassa versiossa `IORD` ja `IOWR` -käskyillä, mutta mitään takuita ei ole käytännön jatkuvuudesta. Tästä syystä nämä kaksi käskyä on niin sanotusti kiedottu ylempiin `BATTERYCHARGER_ReadReg` ja `BATTERYCHARGER_WriteReg` -käskyihin. Käännettäessä nämä käskyt ovat inline-tyyppimäärityksen mukaan täysin läpinäkyviä eivätkä ne aiheuta viiveitä eikä sen enempää koodin koon kasvuakaan.

Tämän jälkeen on käytännön toteutuksen aiemmin listauksessa f esitellyistä muutamasta funktiosta. Kuten listauksesta nähdään, ei koko funktiossa ole käytössä ainuttakaan selittämätöntä numeroa eikä näinollen virheitä pääse syntymään kovinkaan helposti. Koodin lukeminen ainoastaan kerran riittää selittämään riittävän yksityiskohtaisesti mitä se tekee, eikä esimerkiksi lisäkommentteja juuri tarvita.

4.6 Testmod – testimoduuli

4.6.1 Moduulin perusrakenne

Testauksen tärkeimmät osiot, eli testausyksiköt koostuvat jokainen yhdestä testmod-kirjastosta sekä nollasta tai useammasta NIOS-kirjastosta. Kaikki testmod-kirjastot ovat myös edellä esitellyn NIOS-kirjaston mallisia ja koostuvat samoista kolmesta tiedostosta. Niiden toteutus on osittain määritelty tarkemmin kuin normaalien NIOS-kirjastojen

toiminta, sillä jokaisesta testimoduulista täytyy löytyä tietyt ennaltamääritetyn näköiset funktiot. Nämä funktiot ovat seuraavia:

- `TESTMOD`*`KIRJASTONNIMI`*`IStart`, funktio jota kutsutaan kun testiyksikkö käynnistetään pääohjelmasta.
- `TESTMOD`*`KIRJASTONNIMI`*`Stop`, funktio jota kutsutaan testiyksikön lopussa.
- `TESTMOD`*`KIRJASTONNIMI`*`Update`, funktio jota pääohjelma toistuvasti kutsuu testiyksikön ollessa aktiivisena.
- `TESTMOD`*`KIRJASTONNIMI`*`GetTestName`, funktio jolla pääohjelma kysyy testiyksiköltä sen nimen.

Funktioiden nimet voisivat tosiasiassa olla muutakin kuin nämä ennaltamäärätyt, kunhan jokainen niistä on muodoltaan oikean näköinen. Tämä siitä syystä että kaikki funktiokutsut perustuvat funktio-osoittimiin. Käytännössä kuitenkin jatketaan NIOS-kirjastoissa esiintyvää linjaa, eli funktiot nimetään kirjaston nimen mukaan. Alla olevassa listauksessa (listaus 32) on näkyvissä `TESTMODBATTERYCHARGER`-testimoduulin kyseisten funktioiden alut.

```
//------------------------------------------------------------------------------
static void TESTMODBATTERYCHARGERStart(void *con)
{
  TESTMODBATTERYCHARGER *bccon = (TESTMODBATTERYCHARGER *)con;
  bccon->curState  = TESTMODBATTERYCHARGER_STATE_QUERY;
}
//------------------------------------------------------------------------------
static void TESTMODBATTERYCHARGERStop(void *con)
{
}
…
//------------------------------------------------------------------------------
static uint32_t TESTMODBATTERYCHARGERUpdate(void *con) {
  int32_t doTest;
  int gettest;
  TESTMODBATTERYCHARGER *bccon = (TESTMODBATTERYCHARGER *)con;

  // Always print mode and halfperiod
  TESTMODBATTERCHARGER_INTERNAL_PrintMode(bccon);
  uint32_t halfperiod = BATTERYCHARGERGetHalfPeriod(bccon->bc);
  TESTIOAPIPrintf( bccon->io, "Current halfperiod: %d\n", halfperiod );

…
//------------------------------------------------------------------------------
static const char* TESTMODBATTERYCHARGERGetTestName(void *con) {
  return testname;
}
//------------------------------------------------------------------------------
```

LISTAUS 32. TESTMODBATTERYCHARGER-testimoduulin pääfunktiot.

Kuten listauksesta voidaan nähdä, on `Start`-funktio varsin yksinkertainen. Sen ainoa tehtävä on asettaa testimoduulin sisäinen tila oikeaksi. `Stop`-funktio ei tässä moduulissa tee mitään. Myöskin kaikki `GetTestName`-funktiot ovat täysin identtisiä keskenään, sillä niiden ainoa tehtävä on palauttaa osoitin moduulin sisäiseen, staattiseen, ja näinollen muutoin nimiavaruudesta piilotettuun, muuttujaan. Update-funktiosta on näkyvissä vain alkuosa, sillä sen toiminta on monimutkaisempi ja esitellään tarkemmin myöhemmin. Sen paluutyyppi on myös `uint32_t`, mutta muutoin se ei eroa kutsultaan muista funktioista sen enempää.

Näiden neljän funktion lisäksi jokaiseen testimoduuliin kuuluu koko joukko erilaisia alustus- ja asetusfunktioita. Näitä kaikkia ei kaikista testimoduuleista löydy testien erilaisuuden vuoksi. Alustus- ja asetusfunktioiden tarkoitus on ”piilottaa” pääfunktioiden osoitteet käännösvaiheessa sen vuoksi etteivät ne vie muutoinkin kallisarvoista muistitilaa. Tyypillisesti näihin funktioihin kuuluu seuraavat:

- `TESTMOD`*`KIRJASTONNIMI`*`SetupOnline`, funktio jolla asetetaan testi-instanssin sisäisiin muuttujiin funktio-osoittimet testimoduulin pääfunktioihin.
- `TESTMOD`*`KIRJASTONNIMI`*`SetupContext`, funktio jolla kerrotaan testimoduulille sen tarvitsemien NIOS-kirjastojen struktuurit.
- `TESTMOD`*`KIRJASTONNIMI`*`SetupMenu`, funktio jolla rakennetaan testimoduulin testivalikko.
- `TESTMOD`*`KIRJASTONNIMI`*`Init`, testimoduulin alustusfunktio.

```
//---------------------------------------------------------------------------
void TESTMODBATTERYCHARGERSetupOnline( TESTINSTANCE *ti ) {
  TESTINSTANCESetStart(  ti, TESTMODBATTERYCHARGERStart );
  TESTINSTANCESetStop(   ti, TESTMODBATTERYCHARGERStop );
  TESTINSTANCESetUpdate( ti, TESTMODBATTERYCHARGERUpdate );
}
//---------------------------------------------------------------------------
void TESTMODBATTERYCHARGERSetupContext( TESTMODBATTERYCHARGER *con, BATTERYCHARGER *bc )
{
  con->bc = bc;
}
//---------------------------------------------------------------------------
static const char *strenable          = "Set mode Force on";
static const char *strautomode        = "Set mode Automode";
static const char *strdisable         = "Set mode Off";
static const char *strwritehalfperiod = "Write new halfpediod value";
static const char *strcalibrate       = "Run calibration sweep";

void TESTMODBATTERYCHARGERSetupMenu( TESTMODBATTERYCHARGER *con ) {
  TESTIOAPIMENUInit( &con->menu, con->io );

  con->mienable.desc  = strenable;
  con->mienable.value = 1;
  TESTIOAPIMENUAddItem( &con->menu, &con->mienable );
…
//---------------------------------------------------------------------------
void TESTMODBATTERYCHARGERInit( TESTMODBATTERYCHARGER *con,
                                TESTINSTANCE *ti,
                                TESTIOAPI *ioapi )
{
  TESTIOAPIPrintf( ioapi, "Test <%s>: Initialize\n", testname );
  TESTINSTANCESetContext( ti, con );
  TESTINSTANCESetName( ti, TESTMODBATTERYCHARGERGetTestName );

  con->io             = ioapi;
  con->bc             = 0;
  con->prevhalfperiod = 1784;
  con->curState       = TESTMODBATTERYCHARGER_STATE_QUERY;
}
//---------------------------------------------------------------------------
```

LISTAUS 33. TESTMODBATTERYCHARGER-testimoduulin alustus- ja asetusfunktiot.

Edellä olevassa listauksessa (listaus 33) on esitetty pääosin `TESTMODBATTERYCHARGER`-testimoduulin alustus- ja asetusfunktiot. Niiden tehtävät ovat melko yksinkertaisia, erityisesti `SetupContext`-funktion, mutta muiden tehtävät selitetään tarkemmin myöhemmissä osioissa.

4.6.2 Update-funktion toiminta

Aiemmin mainittu testmoduulin toistuvasti ajettava Update-funktio on vastuussa varsinaisten testien suorittamisesta. Sen paluuarvoista päätellään vieläkö saman testimoduulin suorittamista jatketaan, onko siinä tapahtunut virhe, vai haluaako käyttäjä lopettaa

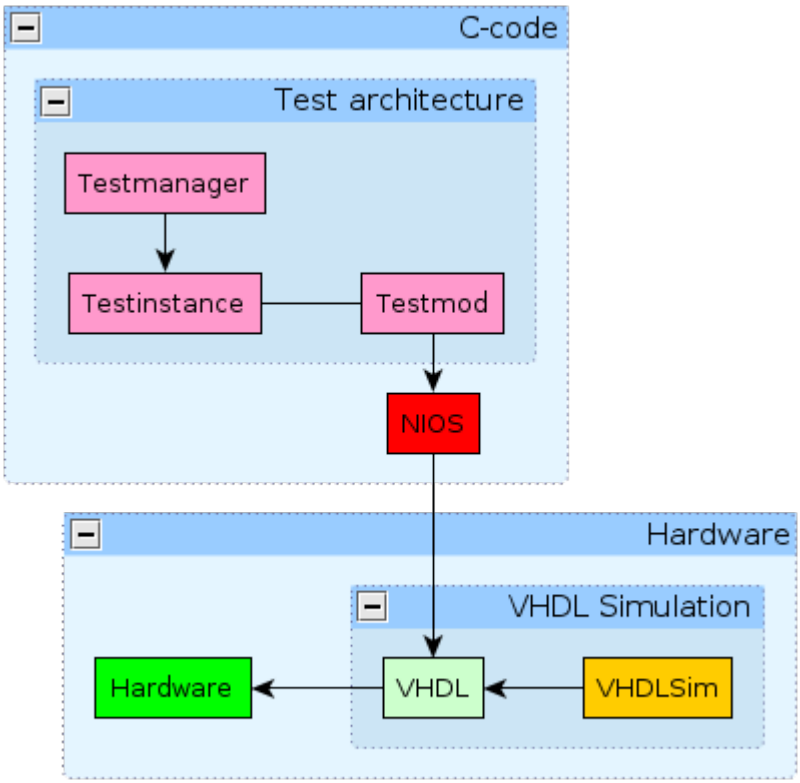
kyseisen testimoduulin käytön. Update-funktio sisältää käytännössä yhden `switch` -rakenteen, joka joko näyttää testimoduulin päävalikon tai jatkaa aktiivisen testin suorittamista. `TESTMODBATTERYCHARGER`-testimoduulin Update-funktio on esiteltynä alla (listaus 34):

```
static uint32_t TESTMODBATTERYCHARGERUpdate(void *con) {
  int32_t doTest;
  int gettest;
  TESTMODBATTERYCHARGER *bccon = (TESTMODBATTERYCHARGER *)con;

  // Always print mode and halfperiod
  TESTMODBATTERCHARGER_INTERNAL_PrintMode(bccon);
  uint32_t halfperiod = BATTERYCHARGERGetHalfPeriod(bccon->bc);
  TESTIOAPIPrintf( bccon->io, "Current halfperiod: %d\n", halfperiod );

  switch( bccon->curState )
  {
        case TESTMODBATTERYCHARGER_STATE_QUERY:
          gettest = TESTIOAPIMENUDoQuery( &bccon->menu, &doTest );
          if( gettest == TESTIOAPI_QUERY_CANCEL )
                return TESTINSTANCE_UPDATE_CANCEL;

          if( doTest == 0 )
                return TESTINSTANCE_UPDATE_DONE;
          else if( doTest == 1 ) {
                bccon->curState = TESTMODBATTERYCHARGER_STATE_ENABLE;
          } else if( doTest == 2 ) {
                bccon->curState = TESTMODBATTERYCHARGER_STATE_AUTOMODE;
          } else if( doTest == 3 ) {
                bccon->curState = TESTMODBATTERYCHARGER_STATE_DISABLE;
          } else if( doTest == 4 ) {
                bccon->curState = TESTMODBATTERYCHARGER_STATE_WRITEHALFPERIOD;
          } else if( doTest == 5 ) {
                bccon->curState = TESTMODBATTERYCHARGER_STATE_CALIBRATE;
          }
          return TESTINSTANCE_UPDATE_CONTINUE;
          break;

        case TESTMODBATTERYCHARGER_STATE_ENABLE:
          return TESTMODBATTERYCHARGEREnable(bccon);
          break;

        case TESTMODBATTERYCHARGER_STATE_AUTOMODE:
          return TESTMODBATTERYCHARGERAutomode(bccon);
          break;
        case TESTMODBATTERYCHARGER_STATE_DISABLE:
          return TESTMODBATTERYCHARGERDisable(bccon);
          break;

        case TESTMODBATTERYCHARGER_STATE_WRITEHALFPERIOD:
          return TESTMODBATTERYCHARGERWriteHalfPeriod(bccon);
          break;

        case TESTMODBATTERYCHARGER_STATE_CALIBRATE:
          return TESTMODBATTERYCHARGERCalibrate(bccon);
          break;

        default:
          return TESTINSTANCE_UPDATE_DONE;
          break;
  }

  return TESTINSTANCE_UPDATE_DONE;
}
```

LISTAUS 34. TESTMODBATTERYCHARGER-testimoduulin Update-funktio.

4.7 Testausarkkitehtuuri

Hyvin pitkälti jokainen testitapaus koostuu muutamasta perusosasta. Nämä ovat elektroniikka (Hardware), sitä ohjaava VHDL-koodi, VHDL-koodin kanssa kommunikoiva NIOS-kirjasto, kirjastoa käyttävä testimoduuli ja ylimmällä tasolla testejä ohjaava pääohjelma. Yleinen arkkitehtuurikuvaus on esitetty alla (kuvio 8).



KUVIO 8. Yleinen testausarkkitehtuuri.

4.8 Pääohjelma

Varsinainen pääohjelma on kaikessa lyhykäisyydessään erittäin yksinkertainen. Sen ainoat tehtävät ovat alustaa tulostuksessa käytetty `UARTIRQ`-moduuli (UART-pohjainen keskeytyksiin perustuva kommunikointikirjasto PC:n ja laitteiston välillä), alustaa itse testit ja ajaa testien päähallintaohjelmaa `TESTMANAGER`ia loputtomassa silmukassa.

Tämän lisäksi pääohjelma tulostaa erinäisiä alustustietoja kuten käytössä olevan bootloaderin versionumeron. Ohjelman kokonainen listaus (listaus 35) on esitetty alla.

```
//
// testermain.c
//
#include "stdint.h"
#include "sys/alt_cache.h"
#include "tester/tester.h"
#include "uart.h" // PRINTF()
#include "system.h" // DEBUGIO_BASE
#include "msp430/msp430.h"
//-----------------------------------------------------------------------------
static TESTER tester;
static UARTIRQ uartirq;
//-----------------------------------------------------------------------------
#ifdef IRQUART_PRINTF
int write(int fd, char *buf, int len) {
  if( !UARTIRQSendBuffer( &uartirq, buf, (uint32_t)len ) )
        return -1; // Out of memory
  return len;
}
#endif
//-----------------------------------------------------------------------------
int main() {
  // Turn on led
  IOWR(DEBUGIO_BASE,0,1);

  alt_dcache_flush_all();
  alt_icache_flush_all();

  UARTIRQInit( &uartirq, UART_TESTER_BASE, UART_TESTER_IRQ );
  UARTIRQSendBuffer( &uartirq, "TesterCPU: Start\n", 17 );
  //PRINTSTR("TesterCPU: Start\n");

  TESTERInit( &tester, &uartirq );

  // PRINTSTR("TesterCPU: Initialized\n");
  UARTIRQSendBuffer( &uartirq, "TesterCPU: Initialized\n", 23 );

  uint32_t bootloadermajor = MSP430GetBootloaderVersionMajor(&(tester.msp430));
  uint32_t bootloaderminor = MSP430GetBootloaderVersionMinor(&(tester.msp430));
  TESTIOAPIPrintf( &(tester.testioapi), "Bootloader version: %d.%d\n", bootloadermajor,
bootloaderminor );

  while(1)
        TESTERRun( &tester );
}
//-----------------------------------------------------------------------------
```

LISTAUS 35. Pääohjelma.

Koska käytössä on FPGA-tekniikalla toimiva piiri, tyhjäytyy sen konfiguraatio joka kerta virran katketessa. Tästä syystä tarvitaan erillinen niin sanottu bootloader-piiri, jonka tehtävänä on käynnistyksessä ladata SD-kortilta oikea ohjelma, eli konfiguraatio, FPGA-piirille. Bootloader-piiri toimii myös ulkopuolisena valvojana siltä varalta että laitteistossa tapahtuu jokin olettamaton virhe. Tämä bootloader on toteutettu C-kielellä ohjelmoitavalle MPS430-piirille. Pääohjelma käyttää MSP430 -NIOS-kirjastoa kysyäkseen itse bootloader-ohjelmiston nykyisen versionumeron.

Pääohjelman oleellisin osa on TESTER-moduuli, joka on varsinaisesti vastuussa testien käynnistyksestä. Se ei juurikaan poikkea muista NIOS-kirjastoista, kuten edellä olevasta listauksesta voidaan nähdä. Moduuli alustetaan TESTERInit-funktiolla ja varsinainen käyttö tapahtuu kutsumalla TESTERRun-funktiota toistuvasti. TESTER-moduulin listaus löytyy liitteestä 4.

4.9 Testmanager – testien hallinta

Kuten aiemmin on jo mainittu, jokainen testimoduuli on pääosin samanlainen ja tästä syystä niiden käyttö on myös hyvin pitkälle identtistä. Jokaisella moduulilla on omanlaisensa alustus- ja asetusfunktiot, mutta niiden lisäksi moduuleilla on edellä esitellyt neljä täysin identtistä, joskin erinimistä, funktiota. Nämä funktiot ovat siis Start, Stop, Update ja GetTestName.

Nämä identtisyydet ovat ennaltamäärättyjä siitä syystä että testejä voidaan hallinnoida erillisen TESTMANAGER-moduulin avulla. Tämän moduulin tärkeimmät tehtävät ovatkin listata sen sisältämät testiyksiköt, käynnistää ja pysäyttää testejä, sekä kutsua varsinaista testimoduulin pääkoodia ajavaa Update-funktiota. Lisäksi testien hallintamoduulin avulla lisätään pääohjelman nähtäväksi uudet testimoduulit, tulostetaan testien nimet ja asetetaan tällä hetkellä käytössä oleva testimoduuli.

Testien hallintamoduuli ei varsinaisesti itse ole tietoinen sen sisältämistä testeistä, vaan ainoastaan kahdesta osoittimesta testiyksiköihin. Nämä osoittimet ovat linkitetyn listan ylin linkki (head) sekä nykyisen, käytössä olevan testiyksikön osoitin (current). Lisäksi se on tietoinen testien kokonaislukumäärästä ja tulostuksessa käytettävästä TESTIOAPI-moduulista. TESTMANAGER-moduuli löytyy kokonaisuudessaan liitteestä 5.

4.10 Testinstance – testiyksiköt

Jokainen varsinainen testimoduuli näkyy hallinnointimoduulille vain pienenä, erittäin yksinkertaisena `TESTINSTANCE`-moduulin sisäisen struktuurin osoittimena. Tällä saavutetaan se etu, että hallinnointimoduuli pysyy mahdollisimman yksinkertaisena, eikä sen tarvitse tietää ainoastakaan testistä yksityiskohtia. Tällä saavutetaan myös toinen etu joka on muistin säästö. Pääohjelma voi sisältää kymmeniä eri testejä, mutta hallintointimoduulin kannalta jokainen niistä on vain pieni `TESTINSTANCE`-moduulin instanssi.

Testiyksikön eli `TESTINSTANCE`-moduulin tärkein sisältö onkin juuri aiemmin mainittujen `Start`, `Stop`, `Update` ja `GetTestName` funktio-osoittimien prototyyppien esittely ja niiden säilöminen omaan struktuuriinsa. Tämän lisäksi struktuuri sisältää osoittimen linkitetyssä listassa seuraavaan `TESTINSTANCE`-struktuuriin. Funktio-osoittimien prototyypit ovat alla olevassa listauksessa (listaus 36) nähtävän näköiset. Varsinainen sisäinen struktuuri on esitelty myöskin alla olevassa listauksessa (listaus 37). Kokonaiset listaukset on nähtävissä liitteessä 6.

```
typedef void        STARTFUNCTION(void*);
typedef void        STOPFUNCTION(void*);
typedef uint32_t    UPDATEFUNCTION(void*);
typedef const char* NAMEFUNCTION(void*);
```

LISTAUS 36. Funktio-osoittimien prototyypit.

```
struct TESTINSTANCE
{
  void* context;

  // Callback functions
  STARTFUNCTION  *start;
  STOPFUNCTION   *stop;
  UPDATEFUNCTION *update;
  NAMEFUNCTION   *name;

  // Test master version number, from #defines
  uint32_t masterversion;

  // Pointer to extra description after test name (to differentiate between multiple
instances)
  // By default this is set to zero and will not show up.
  char* extradescription;

  // Linked list pointer
  TESTINSTANCE *next;
};
```

LISTAUS 37. TESTINSTANCE-moduulin sisäinen struktuuri.

Edellä esitellyt funktioiden prototyypit (listaus 36) ovat jokaisen testimoduulin neljän pääfunktion prototyyppejä. Aiemmin mainittu `SetupOnline`-funktio (listaus 33, sivu 57) käyttää nimenomaisesti näitä prototyyppejä tallentaessaan testimoduulin funktioiden osoitteet testi-instanssin sisäiseen rakenteeseen.

4.11 Testioapi - Tulostus

Useaan kertaan mainittu FPGA-piirin sisäisen muistin vähyys on pakottanut käyttämään myös omia tulostusfunktioita, sillä C-kielen standardifunktiot ovat liian laajat ja monimutkaiset hyvittääkseen niiden vaatiman muistitilan. Tästä syystä onkin kehitetty oma kirjasto myös tulostusta varten. Käytännössä kirjasto toimii hyvin pitkälle standardien tulostusfunktioiden kaltaisesta, tosin sillä erotuksella että se tukee vain osaa varsinaisessa standardissa määritellyistä ominaisuuksista.

`TESTIOAPI`-moduuli sisältää useita erilaisia tulostuksessa ja käyttäjän syötteen luvussa tarvittavia funktioita. Näitä funktioita ovat esimerkiksi kokonaislukujen tulostus eri muodoissa (normaali, heksaluku ja binääriluku), yksittäisten merkkien tulostus ja lukeminen, lähes standardimallinen printf-funktio, kokonaislukujen kysely ja erilaisten vakiokysymysten kysyminen käyttäjältä. Moduulin esittelytietosto on esitelty alla olevassa listauksessa (listaus 38). Kokonainen listaus löytyy liitteestä 7.

```
//-------------------------------------------------------------------------
// Query function return values
#define TESTIOAPI_QUERY_ERROR  0x00
#define TESTIOAPI_QUERY_OK     0x01
#define TESTIOAPI_QUERY_CANCEL 0x02
// Possible choices for confirmation query
#define TESTIOAPI_CONFIRMCHOICES_YESNO       0x10
#define TESTIOAPI_CONFIRMCHOICES_YESNOCANCEL 0x11
// Possible return values from confirmation query
#define TESTIOAPI_CONFIRM_YES    0x20
#define TESTIOAPI_CONFIRM_NO     0x21
#define TESTIOAPI_CONFIRM_CANCEL 0x22
//-------------------------------------------------------------------------
struct TESTIOAPI;
typedef struct TESTIOAPI TESTIOAPI;
typedef TESTIOAPI* TESTIOAPIPTR;
//-------------------------------------------------------------------------
// Function pointers used by PutChar and GetChar
typedef int  (*TESTIOAPIISCHARREADYCALLBACK)(void); // Returns non-zero if new data is
readable.
typedef char (*TESTIOAPIGETCHARCALLBACK)(void);     // Returns next letter from receive
buffer (0 if new data was not available)
typedef int  (*TESTIOAPIPUTCHARCALLBACK)(char ch);  // Returns zero if buffer is full,
non-zero if data was sent.
//-------------------------------------------------------------------------
void TESTIOAPIInit( TESTIOAPI *ioapi,
                    TESTIOAPIISCHARREADYCALLBACK chrready,
                    TESTIOAPIGETCHARCALLBACK getchr,
                    TESTIOAPIPUTCHARCALLBACK putchr );
//-------------------------------------------------------------------------
// Print functions
// Return zero if buffer is full, non-zero if data was sent?
void TESTIOAPIPrintStr( TESTIOAPI *ioapi, const char *str );
void TESTIOAPIPrintInt( TESTIOAPI *ioapi, int32_t num );
void TESTIOAPIPrintHex( TESTIOAPI *ioapi, int32_t num );
void TESTIOAPIPrintHexSmall( TESTIOAPI *ioapi, int32_t num );
void TESTIOAPIPrintBin( TESTIOAPI *ioapi, int32_t num );
void TESTIOAPIPrintf( TESTIOAPI *ioapi, const char *fmt, ... );
//-------------------------------------------------------------------------
// Read or write a single character
void TESTIOAPIPutChar( TESTIOAPI *ioapi, char ch );
char TESTIOAPIGetChar( TESTIOAPI *ioapi ); // Non-blocking in altera
//-------------------------------------------------------------------------
// Query functions return two values
//   Function return value is one of TESTIOAPI_QUERY_xxx
//   And actual value is returned in last parameter (integer pointer)
//   query parameter can be null, in which case default is used.
// Integer
int TESTIOAPIQueryInteger( TESTIOAPI *ioapi, const char *query, int32_t *number );
int TESTIOAPIQueryRangeInteger( TESTIOAPI *ioapi, const char *query, int32_t min_val,
int32_t max_val, int32_t *number );
int TESTIOAPIQueryIntegerDefault( TESTIOAPI *ioapi, const char *query, int32_t def,
int32_t *number );
int TESTIOAPIQueryRangeIntegerDefault( TESTIOAPI *ioapi, const char *query, int32_t
min_val, int32_t max_val, int32_t def, int32_t *number );
//-------------------------------------------------------------------------
// Confirmation query
// Third parameter must be one of TESTIOAPI_CONFIRMCHOICES_xxx
// Return value is one of TESTIOAPI_QUERY_xxx
// Actual return value is in last parameter, it is one of TESTIOAPI_CONFIRM_xxx
int TESTIOAPIQueryConfirmation( TESTIOAPI *ioapi, const char *query, int choices, int
*select );
int TESTIOAPIQueryConfirmationDefault( TESTIOAPI *ioapi, const char *query, int choices,
int def, int *select );
//-------------------------------------------------------------------------
// String query
// Maximum buffer size (pointed to by buffer) is in the fourth parameter
// Third parameter returns the number of characters read
// Return value is one of TESTIOAPI_QUERY_xxx
int TESTIOAPIQueryString( TESTIOAPI *ioapi, const char *query, char *buffer, uint32_t
*buffersize, uint32_t maxbuffersize );
//-------------------------------------------------------------------------
#include "testioapi/testioapi_inline.h"
//-------------------------------------------------------------------------
```

LISTAUS 38. TESTIOAPI-moduulin funktioiden esittelytiedosto testioapi.h.

5 TESTIMODUULIT

5.1 DDR-muisti

FPGA-piirien tila on muistin suhteen yleensä hyvin rajallinen ja tästä syystä laitteissa onkin erillinen DDR-muisti jota käytetään muun muassa ohjelmistoprosessorien suoritusmuistina, ethernet-liikenteen puskurina ja säilyttämään moottorien ohjauksessa tarvittavia virtaohjetaulukoita sekä muita vastaavia tietoja. DDR-muisti on koko laitteiston oikean toiminnan kannalta oleellisin osa ja sen testaus on siten tärkeysjärjestyksessä ensimmäisiä.

DDR-muistin testaus poikkeaa edellä esitellystä arkkitehtuurista jonkin verran. Suurin ero on erillisten NIOS- ja VHDL-kirjastojen puuttuminen. Tämä johtuu siitä syystä, että Alteran Quartus-ympäristö sisältää valmiin, maksullisen ja varmasti testatun IP-lohkon (Intellectual property) DDR-muistien käsittelyä varten. Muistin testaus rajoittuu näinollen ainoastaan testimoduuliin.

Muistin testausta varten on tosin kehitetty oma näennäissatunnaislukugeneraattori VHDL-kielellä. Sen toiminta perustuu lineaariseen takaisinkytkettyyn siirtorekisteriin (LFSR, eli Linear Feedback Shift Register), joka tuottaa näennäisesti satunnaisen, pitkän sarjan numeroita. Oikein suunniteltu LFSR tuottaa tietynmittaisen bittijonon kaikki mahdolliset yhdistelmät (lukuunottamatta jompaa kumpaa ääripäätä), joten se on muistin testauksessa ideaaliratkaisu. Sen avulla DDR-muisti pakotetaan vaihtamaan sivuja tiuhaan tahtiin, ja toivon mukaan sillä löydetään mahdolliset viat. Samaa LSFR:iä voidaan käyttää sekä muistipaikkojen että sisällön arpomiseen. VHDL-kielellä toteutettu LSFR testataan lisäksi erikseen Python-simulaatiotestillä.

Testaus perustuu nimenomaan siihen, että muistiin kirjoitetaan näennäisesti satunnaisiin paikkoihin näennäisesti satunnaisia arvoja, siten että samat satunnaisluvut voidaan luoda

yhä uudestaan ja uudestaan. Näin sekä muistin kirjoitus että lukeminen ovat toisistaan riippumattomia osia ja voidaan suorittaa kerta toisensa jälkeen. Kirjoitus on tosin tehtävä vähintään kertaalleen ennen lukemista. Koko 128 MB kokoinen muisti kirjoitetaan kerralla täyteen ja sen sisältö voidaan lukea myöhemmin kokonaan, verraten jokaisen muistipaikan sisältöä satunnaislukugeneraattorin tuottamiin arvoihin. Mikäli arvot täsmäävät koko muistin sisällössä, voidaan todeta testin olleen onnistunut.

5.2 MSP430-bootloader

Koska laitteissa on käytössä nimenomaan FPGA-piirit, joiden sisältö haihtuu aina virran katkeamisen yhteydessä, tarvitaan myös erillinen bootloader. Sen tehtävänä on laitteen käynnistyessä lukea SD-kortilta FPGA-piirin konfiguraatio ja ladata se piirille. Sama bootloader-piiri toimii myös vahtikoirana (watchdog) koko FPGA-piirille, tuottaa niinsanotun Safe-signaalin PWM-pulssin toisen puolikkaan ja sisältää lisäksi piirilevyn sarjanumeron. Bootloader-piiriksi valittiin MSP430F223. Bootloader myös päättää mikä ohjelmistoversio SD-kortilta ladataan.

MSP430-piirin ohjelmisto on kirjoitettu kokonaan C-kielellä ja se kommunikoi FPGA-piirin kanssa SPI-väylän avulla. Testaus on siis käytännössä lähinnä SPI-väylän kautta tapahtuvan kommunikoinnin testausta. Bootloaderin on joka tapauksessa toimittava vähintään SD-kortin latauksen ja watchdogin osalta, sillä muutoin varsinaiseen testiohjelmaan ei olisi mahdollista edes päästä.

MSP430-piirin testaus noudattaa hyvin pitkälti edellä esiteltyä kaavaa. Se sisältää melko laajan VHDL-kirjaston ja sen simulaatiotestit, pienehkön NIOS-kirjaston ja testimoduulin. Näiden lisäksi testi pitää periaatteessa sisällään myös MSP430-piirin C-kielisen ohjelmiston testauksen, sillä molempien puolien on toimittava yhdessä oikein.

5.3 SD-kortti

Kuten edellisissä osiossa mainittiin, täytyy FPGA-piirin konfiguraation lisäksi SD-kortilta ladata kaikenlaisia taulukoita. Myös SD-kortin on siis toimittava moitteettomasti että laitteisto koskaan saadaan käyntiin. Testaus noudattaa jälleen samaa kaavaa kuin muutkin normaalit testikokonaisuudet. SD-korttia ohjaa VHDL-kirjasto, VHDL-kirjaston kanssa kommunikoi tässä tapauksessa melko laaja NIOS-kirjasto ja testimoduuli puolestaan käyttää NIOS-kirjastoa erilaisten testien ajamiseen.

SD-kortin testaus on periaatteessa hyvin yksinkertainen kirjoitus- ja lukutesti. Tietylle sektorille kirjoitetaan näennäissatunnaislukugeneraattorilla tuotettu testidata joka myöhemmin luetaan sieltä ja verrataan oletettuun sisältöön. Kokonaisen SD-kortin (2 GB) testaus olisi laitteistolla erittäin hidasta, joten tässä luotetaan PC:llä olevaan valmiiseen SD-kortin testeriin. Laitteistotestauksessa riittää muutaman satunnaissektorin testaus.

5.4 Moottoriohjain

Laitteiston ehdottomasti monimutkaisin kokonaisuus on moottoriohjain ja siihen kuuluvat alikomponentit. Moottorien ohjaukseen ja testaukseen kuuluu monta erillistä, toinen toistaan monimutkaisempaa VHDL- ja NIOS-kirjastoa. Jokaiseen moottoriin kuuluu virtaohjaimen ja varsinaisen moottoriohjaimen lisäksi myös kulma-anturi ja sen testaus. Testimoduuli on tästä syystä jonkin verran normaalia monimutkaisempi ja laajempi. Simulaatiotestit on tehty vain osalle VHDL-kirjastoja ja komponentteja, sillä niiden lähes satunnaisiin fysikaalisiin muutoksiin perustuvien mittausarvojen simulointi on monimutkaisuuden vuoksi käytännössä mahdotonta.

Moottorien testaus on laajasta VHDL- ja NIOS-kirjastokokoelmasta huolimatta kohtuullisen suoraviivaista. Kulma-anturi testataan käsin pyörittämällä testattavaa moottoria ja lukemalla nopeaan tahtiin sen lähettämiä arvoja. Mikäli arvot muuttuvat noudattaen lähes sinimuotoista käyrää, voidaan anturin todeta olevan ehjä. Itse moottorin pyörimisen

testaus puolestaan voidaan toteuttaa kahdella tapaa. Toinen vaihtoehto on ohittaa C-kielinen moottoriohjainkirjasto kokonaan ja generoida virtaohjeet manuaalisesti eri vaiheille, tai sitten käyttää hyväksi moottoriohjainta ja asettaa ainoastaan nopeuden (momentti) sen avulla.

Näiden lisäksi moottorille voidaan ohjata suoraan kohtuullisen suuret, pysyvät virta-arvot ja testata kuinka vahvasti moottori kykenee pysymään paikallaan. Osassa moottoreista on myös jarrut, ja näitä varten on myös olemassa oma testinsä. Jarrujen testi on hyvin yksinkertainen auki/kiinni-testaus.

5.5 AMR-anturit

Laitteistosta löytyy myös kohtuullisen lukuisa joukko AMR-antureita joilla mitataan lähinnä erilaisten mekaanisten liikkeiden tilaa. AMR-antureiden testaukseen ei liity varsinaisesti mitään VHDL-kirjastoa, sillä elektroniikka hoitaa antureilta suoraan kaksi signaalia. Nämä ovat tunnistus- ja virhesignaalit, joten testaus on myös hyvin yksinkertaista. Virhetila esiintyy mikä anturi on joko rikki tai irti ja tunnistussignaali puolestaan kertoo tunnistaako anturi sillä hetkellä magneettikentän vai ei. Testaus hoidetaan siis käsin irroittamalla ja kiinnittämällä anturi sekä tuomalla pieni magneetti anturin lähelle.

5.6 AD-muuntimet

Lähes jokaisesta laitteistosta löytyy enemmän tai vähemmän AD-muuntimia erilaisten jännitearvojen mittausta varten, eikä tämäkään laitteisto ole poikkeus. AD-muuntimien data tulee FPGA-piirille SPI-väylää pitkin, jota puolestaan ohjaa oma VHDL-kirjastonsa. Testaus on siis käytännössä vain SPI-kommunikoinnin testaus. Testimoduuli sisältää tässä tapauksessa vain yhden testin, eli AD-muuntimien arvojen lukemisen. Mikäli arvot ovat lähellä oletettuja arvoja, voidaan testin todeta olleen onnistunut.

5.7 Radio

Kuten *batterycharger*-kokonaisuuden yhteydessä kerrottiin, on laitteessa kaksi liikkuvaa sensorilapaa. Kommunikointi näiden lapojen ja piirilevyjen välillä hoidetaan radioteitse. Radion kirjastot ovat kohtuullisen monimutkaisia, sillä ne hoitavat liikennöinnin lähes kokonaan itse. Testaus vaatii luonnollisestikin kaksi erillistä piirilevyä jotka kykenevät lähettämään viestejä keskenään. Muutoin radion testaus noudattaa normaalia kaavaa, eli siinä on VHDL-kirjasto ja sen simulointitestit, muutama NIOS-kirjasto sekä testimoduuli.

Radion kanssa kommunikoinnin hoitaa kokonaan erillinen prosessori, joten sen käyttö on hieman monimutkaisempaa kuin muiden kirjastojen kanssa. Kaikki viestit testiprosessorin ja radioprosessorin välillä kulkevat jaetun muistin kautta. Muistin käyttövuorojen jako on hoidettu omalla mutex-kirjastolla (mutual exclusion) joka on toteutettu VHDL-kirjastossa kellojakson mittaisena atomisena operaationa. Koska kaikki prosessorit toimivat samassa FPGA-prosessorissa, saadaan tällaisen mutexin avulla estettyä muiden prosessorien pääsy tiettyyn muistiin. Sillä taas vältytään yllättäviltä tilanteilta kun kaksi prosessoria yrittää kirjoittaa samanaikaisesti samaan muistiosoitteeseen. Radion testauksen yhteydessä tulee siis myös osittain testattua mutex-kirjastojen toiminta.

Varsinainen radion testaus hoidetaan kohtuullisen yksinkertaisesti asettamalla kahdelle testattavalle radiopiirille sama taajuus ja sama osoite, ja lähettämällä viestejä näiden välillä. Mikäli suurin osa viesteistä tulee onnistuneesti perille, voidaan testin todeta onnistuneen. Radioliikenteessä törmäyksiä ja signaalin korruptoitumista tapahtuu väistämättäkin, joten testauksessa ei voida olettaa joka ainoan paketin päätyvän perille kokonaisuudessaan. Radiokirjastot eivät myöskään ota kantaa siihen menikö joku paketti perille, vaan ainoastaan siihen että onnistuiko lähetys tai saatiinko ehjä paketti. Ylemmän tason tehtävänä on huolehtia uudelleenlähetyksestä.

5.8 Viiva-anturi

Koska koko laitteisto kulkee kiskoa pitkin paikasta toiseen, tarvitaan sen paikkatieto. Tähän ongelmaan ratkaisuna on viiva-anturi. Laitteisto kulkee mustavalkokoodattua kiskoa pitkin ja viiva-anturit lukevat tätä koodausta. Sen perusteella voidaan päätellä missä kohtaa laitteisto on. Viiva-anturien testaus noudattaa jälleen samaa peruskaavaa, eli VHDL-kirjasto, NIOS-kirjasto ja testimoduuli.

Testaus on viiva-anturien tapauksessa melko yksinkertaista, sillä testimoduuli ei varsinaisesti yritä tulkita paikkakoodausta, vaan lukee ainoastaan anturin tunnistamat kolme raitaa. Näistä tuloksena saadaan kolme arvoa jotka kertovat mitkä värit tunnistettiin. Mikäli anturi tunnistaa onnistuneesti sekä mustan että valkoisen, voidaan testin todeta onnistuneen.

5.9 Etäisyysanturi

Viiva-anturien lisäksi laitteistossa on etäisyysantureja joiden avulla se paikoitetaan. Niiden testaus on hyvin pitkälti samankaltainen viiva-anturien kanssa, ainoastaan sillä poikkeuksella että lukuarvot kertovat tunnistetun värin sijasta etäisyyden. Muutoin testiin kuuluu täysin samat asiat kuin viiva-anturin testiinkin.

5.10 Kallistusanturi

Etäisyys- ja viiva-anturien lisäksi laitteistosta löytyy vielä kallistusantureita joita käytetään liikkeen vakauttamiseen ja laitteiston suorassa pitämiseen. Jälleen kerran testaus on hyvin samankaltainen kuin muissakin antureissa. Anturilta luettavat arvot kertovat kolmen akselin (X, Y ja Z) lukemat ja mikäli ne piirilevyä kallistettaessa muuttuvat oikein, testi on onnistunut.

5.11 Valoverhot

Viimeisenä laitteistosta löytyy kaiken lisäksi valoverhoja erilaisten muotojen tunnistusta varten. Nämä valoverhot koostuvat kymmenistä pienistä infrapuna-LED-lähettimistä ja -vastaanottimista. VHDL-kirjasto ohjaa vuorotellen jokaiseen lähetin-vastaanotin-pariin tietynlaisen bittipurskekuvion suurella nopeudella. Mikäli vastaanotin näkee samat bitit mitkä lähetin lähettää, on valoverhon yhteys siltä kohtaa vapaa, ja mikäli ei, on lähettimen ja vastaanottimen välillä este. Jokainen LED-pari käydään läpi vuorotellen siksi, että infrapuna-LEDien valo ei ole erityisen suunnattua ja ne häiritsisivät muutoin toisiaan. VHDL-kirjasto käy tätä sykliä läpi melko suurella nopeudella, joten pienimmätkin esteet havaitaan nopeasti.

Testaukseen kuuluu jälleen samat peruskirjastot ja -testit, eli VHDL- ja NIOS-kirjastot, VHDL-simulaatiotesti sekä testimoduuli. Moduulista löytyy ainoastaan yksi testi jolla luetaan valoverhon LEDien tila. Testaus suoritetaan sekä ilman esteitä, jolloin LEDien tilan on oltava kokonaan vapaa että asettamalla sopivia esteitä valoverhon väliin ja vertaamalla niiden muotoa LEDien tilaan. Mikäli ne vastaavat toisiaan, on testaus onnistunut.

6 POHDINTOJA

Tässä dokumentissa esitelty tapa simuloida ja testata elektroniikkaa ja VHDL-koodia on vain yksi monista tavoista simuloida ja testata. Käytännössä kuitenkin VSIM-ympäristö on osoittautunut yhdeksi helpoimmista ja tehokkaimmista tavoista VHDL-koodin testaamiseksi. Myös NIOS-kirjastot ja testimoduulit ovat selkeydessään erittäin helposti käytettäviä ja tehokkaita menetelmiä testauksessa.

Tämän ympäristökokonaisuuden avulla saadaan helposti ja nopeasti testattua jokaisen piirilevyn jokainen toiminnallinen kokonaisuus. Ja kuten alussa mainittiin, on testaus käytännössä kaikkien projektien sydän ja se tärkein osuus. Testeriä on käytetty aktiivisesti noin puolen vuoden ajan ja se on osoittautunut korvaamattomaksi työkaluksi sekä vikojen löytymisessä että diagnosoinnissa.

LÄHTEET

[IRS2004]: Internal Rectifier, 2006. IRS2004(S)PbF HALF-BRIDGE DRIVER http://www.irf.com/product-info/datasheets/data/irs2004pbf.pdf

[VHDL] Wikipedia, 2011. VHDL http://en.wikipedia.org/wiki/VHDL

[MNL-AVABUSREF-2.0]: Altera Corporation, 2011. Avalon Interface Specifications http://www.altera.com/literature/manual/mnl_avalon_spec.pdf

[NII5V1-11.0]: Altera Corporation, 2011. Nios II Processor Reference Handbook http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf

LIITTEET

LIITE 1

BATTERYCHARGER VHDL-kirjastot

```
--!
--! @file batterycharger_avalon.vhd
--! @brief Battery charger using Avalon interface
--!
--! Charges the batteries of the gripper light curtains

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
-------------------------------------------------------------------------------
entity batterycharger_avalon is
  generic (
    SYSTEMCLOCKFREQUENCY : integer := 50000000;  -- [Hz] System clock frequency
    DEFAULTFREQUENCY     : integer := 14000;  -- [Hz] Default generated frequency
    COUNTERMAX           : integer := 50000000);  -- Maximum count value

  port (
    clk     : in std_logic;             --! System clock
    reset_n : in std_logic;             --! Reset (active low)

    -- Avalon slave interface
    as_chipselect    : in  std_logic;   --! Avalon slave interface: Chip select
    as_begintransfer : in  std_logic;  --! Avalon slave interface: Begin transfer
    as_address       : in  std_logic_vector(2 downto 0);  --! Avalon slave interface:
Address
    as_read          : in  std_logic;   --! Avalon slave interface: Read enable
    as_readdata      : out std_logic_vector(31 downto 0);  --! Avalon slave interface:
Read data
    as_write         : in  std_logic;  --! Avalon slave interface: Write enable
    as_writedata     : in  std_logic_vector(31 downto 0);  --! Avalon slave interface:
Write data

    sweepack      : in std_logic;       --! Input for sweep acknowledge
    maxcurrent    : in std_logic_vector(11 downto 0);  --! Input for max current
    maxhalfperiod : in integer range 0 to 2047;  --! Input for max halfperiod

    sweepreq        : out std_logic;    --! Output for sweep request
    halfperiod      : out integer range 0 to COUNTERMAX;  --! Value which is written to
the counter signal after it goes down to 0.
    enable_forced   : out std_logic;  --! Output for enable charging via avalon
    enable_automode : out std_logic);   --! Output for automatic charging mode

end batterycharger_avalon;
-------------------------------------------------------------------------------
architecture rtl of batterycharger_avalon is

  constant DEFAULTCOUNTER : integer range 0 to COUNTERMAX := (SYSTEMCLOCKFREQUENCY/2/DE-
FAULTFREQUENCY)-1;

  signal as_readdata_s : std_logic_vector(31 downto 0);

  signal halfperiod_s : integer range 0 to COUNTERMAX := DEFAULTCOUNTER;  --! Value
which is written to the counter signal after it goes down to 0.

  signal ena_s      : std_logic;         --! Signal for forced enable
  signal automode_s : std_logic := '0';  --! Signal for automatic charging mode
  signal sweepreq_s : std_logic := '0';  --! Signal for sweep request

  -------------------------------------------------------------------------------
----
  -- Avalon registers

  constant AVALONADDR_RW_MODE               : integer := 16#00#;
  constant AVALONADDR_RW_HALFPERIOD         : integer := 16#01#;
  constant AVALONADDR_RW_SWEEP              : integer := 16#02#;
  constant AVALONADDR_R_SWEEP_MAXHALFPERIOD : integer := 16#03#;
  constant AVALONADDR_R_SWEEP_MAXCURRENT    : integer := 16#04#;

  constant MODEREG_OFF  : integer := 0;
  constant MODEREG_ON   : integer := 1;
  constant MODEREG_AUTO : integer := 2;
```

```
begin
  ----------------------------------------
  --! Avalon slave, read and write

  process(clk, reset_n)
  begin
    if reset_n = '0' then
      as_readdata_s <= (others => '0');
      ena_s         <= '0';             -- Disabled by default
      halfperiod_s  <= DEFAULTCOUNTER;
      automode_s    <= '0';
      sweepreq_s    <= '0';
    elsif rising_edge(clk) then

      ----------------------------------------
      -- Set/get charger state and other variables via avalon

      if as_chipselect = '1' then
        if as_read = '1' and as_begintransfer = '1' then
          as_readdata_s <= (others => '0');

          case to_integer(unsigned(as_address)) is
            when AVALONADDR_RW_MODE =>
              -- Mode register
              -- 0: Turn off
              -- 1: Turn on
              -- 2: Automatic mode, controlled by input signal enablecharging.
              if automode_s = '1' then
                as_readdata_s <= std_logic_vector(to_unsigned(MODEREG_AUTO, 32));
              else
                if ena_s = '1' then
                  as_readdata_s <= std_logic_vector(to_unsigned(MODEREG_ON, 32));
                else
                  as_readdata_s <= std_logic_vector(to_unsigned(MODEREG_OFF, 32));
                end if;
              end if;

            when AVALONADDR_RW_HALFPERIOD =>
              -- Read current halfperiod
              as_readdata_s <= std_logic_vector(to_unsigned(halfperiod_s, 32));

            when AVALONADDR_RW_SWEEP =>
              -- Check if sweep is active
              as_readdata_s(0) <= sweepack xor sweepreq_s;

            when AVALONADDR_R_SWEEP_MAXHALFPERIOD =>
              -- Read found maximum halfperiod, after sweep
              as_readdata_s <= std_logic_vector(to_unsigned(maxhalfperiod, 32));

            when AVALONADDR_R_SWEEP_MAXCURRENT =>
              -- Read found maximum current, after sweep
              as_readdata_s(11 downto 0) <= maxcurrent;

            when others =>

          end case;

        elsif as_write = '1' and as_begintransfer = '1' then
          case to_integer(unsigned(as_address)) is

            when AVALONADDR_RW_MODE =>
              -- Mode register
              -- 0: Turn off
              -- 1: Turn on
              -- 2: Automatic mode, controlled by input signal enablecharging.
              case to_integer(unsigned(as_writedata)) is
                when MODEREG_OFF =>
                  ena_s      <= '0';
                  automode_s <= '0';
                when MODEREG_ON =>
                  ena_s      <= '1';
                  automode_s <= '0';
                when MODEREG_AUTO =>
                  ena_s      <= '0';
                  automode_s <= '1';
                when others =>
              end case;
```

```
            when AVALONADDR_RW_HALFPERIOD =>
              -- Write new halfperiod value
              halfperiod_s <= to_integer(unsigned(as_writedata));

            when AVALONADDR_RW_SWEEP =>
              -- Start new sweep (only if one is not active now)
              if sweepreq_s = sweepack then
                sweepreq_s <= not sweepreq_s;
              end if;

            when others =>

          end case;

        end if;

      end if;

    end if;

  end process;

  ----------------------------------------
  -- Outputs

  as_readdata     <= as_readdata_s;
  halfperiod      <= halfperiod_s;
  enable_forced   <= ena_s;
  enable_automode <= automode_s;
  sweepreq        <= sweepreq_s;

end rtl;
--------------------------------------------------------------------------------
--!
--! @file batterycharger_filter.vhd
--! @brief Battery charger
--!
--! Charges the batteries of the gripper light curtains
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
--------------------------------------------------------------------------------
--! @brief Battery charger
entity batterycharger_filter is
  generic (
    FILTERBITS : integer := 5  -- Number of filter bits for input voltage
    );
  port (
    clk     : in std_logic;             --! System clock
    reset_n : in std_logic;             --! Reset (active low)

    setzero : in std_logic;

    unfiltered : in  std_logic_vector(11 downto 0);  --! Unfiltered current input
    filtered   : out std_logic_vector(11 downto 0));  --! Filtered current output

end entity batterycharger_filter;
--------------------------------------------------------------------------------
architecture rtl of batterycharger_filter is

  signal filtered_s : std_logic_vector(FILTERBITS + 11 downto 0);  --! Signal for filte-
red current

begin

  ----------------------------------------
  -- Filter process
  -- Filter is a simple 1st order digital low pass filter, using the formula below:
  --   y = y + (x - y) >> b
  -- where y is the filtered result, x is the new input value and b is the amount of
filtering bits.

  process(clk, reset_n)
    variable v_filtered_upper : unsigned(FILTERBITS + 11 downto 0);
    variable v_unfiltered     : unsigned(FILTERBITS + 11 downto 0);
    variable v_filtered       : unsigned(FILTERBITS + 11 downto 0);
```

```
  begin
    if reset_n = '0' or setzero = '1' then
      filtered_s <= (others => '0');

    elsif rising_edge(clk) then
      v_filtered       := unsigned(filtered_s);
      v_unfiltered     := resize(unsigned(unfiltered), FILTERBITS + 12);
      v_filtered_upper := resize(unsigned(filtered_s(FILTERBITS + 11 downto
FILTERBITS)), FILTERBITS + 12);

      filtered_s <= std_logic_vector(v_filtered + v_unfiltered - v_filtered_upper);
    end if;
  end process;

  ----------------------------------------
  -- Outputs

  filtered <= filtered_s(FILTERBITS + 11 downto FILTERBITS);
  ----------------------------------------

end rtl;
--------------------------------------------------------------------------------
# TCL File Generated by Component Editor 7.2 on:
# Thu Oct 04 12:02:37 EEST 2007
# DO NOT MODIFY
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
set_source_file "batterycharger.vhd"
set_module "batterycharger"
set_module_description "batterycharger"
set_module_property "displayName" "batterycharger"
set_module_property "author" "Thuy Nguyen"
#set_module_property "className" "AvalonMII"
#set_module_property "displayName" "MII for Avalon bus"
#set_module_property "group" "ess"
set_module_property "libraries" [ list "ieee.std_logic_1164.all" "ieee.numeric_std.all"
"std.standard.all" ]

set_module_property instantiateInSystemModule true
set_module_property version "2.0"
set_module_property group "ess"
set_module_property editable true

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Module parameters
add_parameter "SYSTEMCLOCKFREQUENCY" "integer" "50000000" ""
set_parameter_property "SYSTEMCLOCKFREQUENCY" AFFECTS_PORT_WIDTHS true
add_parameter "DEFAULTFREQUENCY" "integer" "14000" ""
set_parameter_property "DEFAULTFREQUENCY" AFFECTS_PORT_WIDTHS true
add_parameter "COUNTERMAX" "integer" "50000000" ""
set_parameter_property "COUNTERMAX" AFFECTS_PORT_WIDTHS true
add_parameter "FILTERBITS" "integer" "5" ""
set_parameter_property "FILTERBITS" AFFECTS_PORT_WIDTHS true
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Interface clock_sink

add_interface "clock_interface" "clock" "sink" "asynchronous"

add_port_to_interface "clock_interface" "clk" "clk"
add_port_to_interface "clock_interface" "reset_n" "reset_n"

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# I/O
add_interface "iointerface" "conduit" "output" "clock_interface"

# Ports in interface iointerface
add_port_to_interface "iointerface" "tr_sdn" "export"
add_port_to_interface "iointerface" "tr_in" "export"

add_port_to_interface "iointerface" "enablecharging" "export"

add_port_to_interface "iointerface" "trfmcurr" "export"
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Avalon slave interface

# Properties
add_interface "AvalonSlaveInterface" "avalon" "slave" "clock_interface"
```

```
set_interface_property "AvalonSlaveInterface" "isNonVolatileStorage" "false"
set_interface_property "AvalonSlaveInterface" "burstOnBurstBoundariesOnly" "false"
set_interface_property "AvalonSlaveInterface" "readLatency" "0"
#"1"
set_interface_property "AvalonSlaveInterface" "holdTime" "0"
set_interface_property "AvalonSlaveInterface" "printableDevice" "false"
set_interface_property "AvalonSlaveInterface" "readWaitTime" "1"
set_interface_property "AvalonSlaveInterface" "setupTime" "0"
set_interface_property "AvalonSlaveInterface" "addressAlignment" "DYNAMIC"
#NATIVE"
set_interface_property "AvalonSlaveInterface" "writeWaitTime" "0"
set_interface_property "AvalonSlaveInterface" "timingUnits" "Cycles"
set_interface_property "AvalonSlaveInterface" "minimumUninterruptedRunLength" "1"
set_interface_property "AvalonSlaveInterface" "isMemoryDevice" "false"
set_interface_property "AvalonSlaveInterface" "linewrapBursts" "false"
set_interface_property "AvalonSlaveInterface" "maximumPendingReadTransactions" "0"
#set_interface_property "AvalonSlaveInterface" "addressSpan" "16384"

# Ports
add_port_to_interface "AvalonSlaveInterface" "as_chipselect" "chipselect"
add_port_to_interface "AvalonSlaveInterface" "as_begintransfer" "begintransfer"
#add_port_to_interface "AvalonSlaveInterface" "as_waitrequest" "waitrequest"
add_port_to_interface "AvalonSlaveInterface" "as_address" "address"
add_port_to_interface "AvalonSlaveInterface" "as_write" "write"
add_port_to_interface "AvalonSlaveInterface" "as_writedata" "writedata"
add_port_to_interface "AvalonSlaveInterface" "as_read" "read"
add_port_to_interface "AvalonSlaveInterface" "as_readdata" "readdata"

#set_port_direction_and_width "as_chipselect" "input" 1
#set_port_direction_and_width "as_waitrequest" "output" 1
#set_port_direction_and_width "as_address" "input" 16
#set_port_direction_and_width "as_write" "input" 1
#set_port_direction_and_width "as_writedata" "input" 32
#set_port_direction_and_width "as_read" "input" 1
#set_port_direction_and_width "as_readdata" "output" 32

# ============================================================================
--!
--! @file batterycharger.vhd
--! @brief Battery charger
--!
--! Charges the batteries of the gripper light curtains
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
-------------------------------------------------------------------------------
--! @brief Battery charger
entity batterycharger_logic is
  generic (
    SYSTEMCLOCKFREQUENCY : integer := 50000000;  -- [Hz] System clock frequency
    COUNTERMAX           : integer := 50000000;  -- Maximum count value
    FILTERBITS           : integer := 5);
  port (
    clk     : in std_logic;                      --! System clock
    reset_n : in std_logic;                      --! Reset (active low)

    -- Control signals to other VHDL modules
    enablecharging  : in std_logic;     --! Enable charging by automatic logic
    enable_forced   : in std_logic;     --! Enable charging via avalon
    enable_automode : in std_logic;     --! Mode is automatic
    sweepreq        : in std_logic;     --! Sweep request
    halfperiod      : in integer range 0 to COUNTERMAX;  --! Value which is written to
the counter signal after it goes down to 0.
    trfmcurr        : in std_logic_vector(11 downto 0);  --! Input for transformed cur-
rent

    sweepack      : out std_logic;      --! Sweep acknowledge
    maxcurrent    : out std_logic_vector(11 downto 0);  --! Maximum filtered current
    maxhalfperiod : out integer range 0 to 2047;  --! Maximun halfperiod used to maximum
filtered current

    -- Hardware interface
    tr_sdn : out std_logic;             --! Output for charging state
    tr_in  : out std_logic);            --! Output for enabling transformer
end entity batterycharger_logic;
```

```
--------------------------------------------------------------------------------
architecture rtl of batterycharger_logic is

  ----------------------------------------
  -- Component declarations

  component batterycharger_filter
    generic (
      FILTERBITS : integer);
    port (
      clk        : in  std_logic;
      reset_n    : in  std_logic;
      setzero    : in  std_logic;
      unfiltered : in  std_logic_vector(11 downto 0);
      filtered   : out std_logic_vector(11 downto 0));
  end component;

  ----------------------------------------

  constant WAIT_TIMEMS     : integer := 150;  --! Waiting time 150ms
  constant WAIT_COUNTERMAX : integer := (SYSTEMCLOCKFREQUENCY / 1000) * WAIT_TIMEMS;
--! Counter that counts 150ms

  signal hz_s       : std_logic := '0';  --! Signal for enabling transformer output
  signal charging_s : std_logic := '0';  --! Signal for charging state output
  signal counter_s  : integer range 0 to COUNTERMAX := 0;  --! Counter signal for coun-
ter process
  signal filtered_s : std_logic_vector(11 downto 0) := (others => '0');  --! Signal for
filtered current

  signal sweepack_s    : std_logic := '0';  -- Signal for sweep acknowledge
  signal setzero_s     : std_logic := '0';  -- Manual filter reset
  signal sweepactive_s : std_logic := '0';  -- Sweep is active?

  signal sweephalfperiod_s : integer range 0 to 2047 := 1250;  --! Value which is writ-
ten to the counter signal after it goes down to 0

  signal maxcurrent_s    : std_logic_vector(11 downto 0) := (others => '0');  --! Signal
for max filtered current
  signal maxhalfperiod_s : integer range 0 to 2047 := 0;  --! Signal for max halfperiod
used to filter max current

  signal wait_counter_s : integer range 0 to WAIT_COUNTERMAX := 0;  --! Counter signal
for waiting counter process

  type sweep_state is (SS_IDLE, SS_INIT, SS_NEXT, SS_WAIT, SS_MEASURE, SS_DONE);  --!
Sweep states: idle, initialize, next, wait, measure and done
  signal sweep : sweep_state := SS_IDLE;           --! Signal for sweep states

begin

  ----------------------------------------
  -- Batterycharger filter

  batterycharger_filter_inst : batterycharger_filter
    generic map (
      FILTERBITS => FILTERBITS)
    port map (
      clk        => clk,
      reset_n    => reset_n,
      setzero    => setzero_s,
      unfiltered => trfmcurr,
      filtered   => filtered_s);

  ------------------------------------------------------------------------------------
----
  -- Set charging
  process(clk)
  begin
    if reset_n = '0' then
      counter_s         <= 0;
      hz_s              <= '0';
      sweephalfperiod_s <= 1250;
      sweepack_s        <= '0';
      setzero_s         <= '0';
      sweepactive_s     <= '0';
      wait_counter_s    <= 0;
```

```
      sweep             <= SS_IDLE;

    elsif rising_edge(clk) then
      ----------------------------------------
      -- If sweep requested
      if sweepreq /= sweepack_s then
        sweepactive_s <= '1';
        sweep         <= SS_INIT;
      end if;

      ----------------------------------------
      -- If sweep was requested and is active
      if sweepactive_s = '1' then

        if counter_s = 0 then
          counter_s <= sweephalfperiod_s;
          hz_s      <= not hz_s;
        else
          counter_s <= counter_s-1;
        end if;
        charging_s <= '1';

        case sweep is

          when SS_INIT =>
            ----------------------------------------
            -- Sweep init
            -- Set halfperiod maximum, reset max current, max halfperiod and filter
            -- Move to next state
            sweephalfperiod_s <= 1250;
            maxcurrent_s      <= (others => '0');
            maxhalfperiod_s   <= 0;
            sweep             <= SS_NEXT;
            setzero_s         <= '1';

          when SS_NEXT =>
            ----------------------------------------
            -- Next state
            -- Check if halfperiod not minimum, then reduce halfperiod by 1 and move to
waiting state
            -- Otherwise move to done state
            setzero_s <= '0';

            if sweephalfperiod_s > 249 then
              sweephalfperiod_s <= sweephalfperiod_s - 1;
              sweep             <= SS_WAIT;
            else
              sweep <= SS_DONE;
            end if;

          when SS_WAIT =>
            ----------------------------------------
            -- Wait
            -- Wait until counter is maximum, then reset counter and move to measuring
state
            if wait_counter_s = WAIT_COUNTERMAX then
              wait_counter_s <= 0;
              sweep          <= SS_MEASURE;
            else
              wait_counter_s <= wait_counter_s + 1;
              sweep          <= SS_WAIT;
            end if;

          when SS_MEASURE =>
            ----------------------------------------
            -- Measure current
            -- Check if filtered current is higher than maximum current so far
            -- Set max current to filtered current and used halfperiod to max halfperiod
            -- Move back to state Next
            if filtered_s > maxcurrent_s then
              maxcurrent_s    <= filtered_s;
              maxhalfperiod_s <= sweephalfperiod_s;
            end if;

            sweep <= SS_NEXT;

          when SS_DONE =>
```

```
            ---------------------------------------
            -- Sweep done when halfperiod is reduced to its minimum (249)
            -- Set sweep acknowledge
            sweepack_s    <= sweepreq;
            sweep         <= SS_IDLE;
            sweepactive_s <= '0';

          when others =>
            ---------------------------------------
            -- In case of unknown state, go back to idle state
            sweep <= SS_IDLE;

        end case;

        ---------------------------------------
        -- If sweep not requested, regular functionality
      else

        if counter_s = 0 then
          counter_s <= halfperiod;
          hz_s      <= not hz_s;
        else
          counter_s <= counter_s-1;
        end if;

        if enable_automode = '1' then
          charging_s <= enablecharging;
        else
          charging_s <= enable_forced;
        end if;

      end if;

    end if;

  end process;

---------------------------------------
-- Outputs

  tr_in         <= hz_s;
  tr_sdn        <= charging_s;
  sweepack      <= sweepack_s;
  maxcurrent    <= maxcurrent_s;
  maxhalfperiod <= maxhalfperiod_s;

---------------------------------------

end rtl;
--------------------------------------------------------------------------------
--!
--! @file batterycharger.vhd
--! @brief Battery charger
--!
--! Charges the batteries of the gripper light curtains
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
--------------------------------------------------------------------------------
--! @brief Battery charger
entity batterycharger is
  generic (
    SYSTEMCLOCKFREQUENCY : integer := 50000000;  -- [Hz] System clock frequency
    DEFAULTFREQUENCY     : integer := 14000;     -- [Hz] Default generated frequency
    COUNTERMAX           : integer := 50000000;  -- Maximum count value
    FILTERBITS           : integer := 5);        -- Number of filter bits for input vol-
tage
  port (
    clk     : in std_logic;             --! System clock
    reset_n : in std_logic;             --! Reset (active low)

    -- Avalon slave interface
    as_chipselect    : in  std_logic;   --! Avalon slave interface: Chip select
    as_begintransfer : in  std_logic;   --! Avalon slave interface: Begin transfer
    as_address       : in  std_logic_vector(2 downto 0);   --! Avalon slave interface:
Address
    as_read          : in  std_logic;   --! Avalon slave interface: Read enable
```

```
    as_readdata      : out std_logic_vector(31 downto 0);  --! Avalon slave interface:
Read data
    as_write         : in  std_logic;   --! Avalon slave interface: Write enable
    as_writedata     : in  std_logic_vector(31 downto 0);  --! Avalon slave interface:
Write data

    -- Control signals to other VHDL modules
    enablecharging : in std_logic;      --! Enable charging
    trfmcurr       : in std_logic_vector(11 downto 0);

    -- Hardware interface
    tr_sdn : out std_logic;             --! Output for enabling transformer
    tr_in  : out std_logic);            --! Output for charging state

end entity batterycharger;
--------------------------------------------------------------------------------
architecture rtl of batterycharger is

  ----------------------------------------
  -- Component declarations

  component batterycharger_logic
    generic (
      SYSTEMCLOCKFREQUENCY : integer;
      COUNTERMAX           : integer;
      FILTERBITS           : integer);
    port (
      clk             : in  std_logic;
      reset_n         : in  std_logic;
      enablecharging  : in  std_logic;
      enable_forced   : in  std_logic;
      enable_automode : in  std_logic;
      halfperiod      : in  integer range 0 to COUNTERMAX;
      trfmcurr        : in  std_logic_vector(11 downto 0);
      sweepreq        : in  std_logic;
      sweepack        : out std_logic;
      maxcurrent      : out std_logic_vector(11 downto 0);
      maxhalfperiod   : out integer range 0 to 2047;
      tr_sdn          : out std_logic;
      tr_in           : out std_logic);
  end component;

  component batterycharger_avalon
    generic (
      SYSTEMCLOCKFREQUENCY : integer;
      DEFAULTFREQUENCY     : integer;
      COUNTERMAX           : integer);
    port (
      clk              : in  std_logic;
      reset_n          : in  std_logic;
      as_chipselect    : in  std_logic;
      as_begintransfer : in  std_logic;
      as_address       : in  std_logic_vector(2 downto 0);
      as_read          : in  std_logic;
      as_readdata      : out std_logic_vector(31 downto 0);
      as_write         : in  std_logic;
      as_writedata     : in  std_logic_vector(31 downto 0);
      maxcurrent       : in  std_logic_vector(11 downto 0);
      maxhalfperiod    : in  integer range 0 to 2047;
      sweepack         : in  std_logic;
      sweepreq         : out std_logic;
      halfperiod       : out integer range 0 to COUNTERMAX;
      enable_forced    : out std_logic;
      enable_automode  : out std_logic);
  end component;

  signal ena_s           : std_logic;
  signal hz_s            : std_logic;
  signal halfperiod_s    : integer range 0 to COUNTERMAX;
  signal automode_s      : std_logic;
  signal sweepreq_s      : std_logic;
  signal sweepack_s      : std_logic;
  signal filtered_s      : std_logic_vector(11 downto 0);
  signal maxcurrent_s    : std_logic_vector(11 downto 0);
  signal maxhalfperiod_s : integer range 0 to 2047;

begin
```

```
  ----------------------------------------
  --! Batterycharger logic

  batterycharger_logic_inst : batterycharger_logic
    generic map(
      SYSTEMCLOCKFREQUENCY => SYSTEMCLOCKFREQUENCY,
      COUNTERMAX           => COUNTERMAX,
      FILTERBITS           => FILTERBITS)
    port map (
      clk             => clk,
      reset_n         => reset_n,
      enablecharging  => enablecharging,
      enable_forced   => ena_s,
      enable_automode => automode_s,
      halfperiod      => halfperiod_s,
      tr_sdn          => tr_sdn,
      tr_in           => tr_in,
      trfmcurr        => trfmcurr,
      sweepreq        => sweepreq_s,
      sweepack        => sweepack_s,
      maxcurrent      => maxcurrent_s,
      maxhalfperiod   => maxhalfperiod_s);

  ----------------------------------------
  --! Avalon interface

  batterycharger_avalon_inst : batterycharger_avalon
    generic map (
      SYSTEMCLOCKFREQUENCY => SYSTEMCLOCKFREQUENCY,
      DEFAULTFREQUENCY     => DEFAULTFREQUENCY,
      COUNTERMAX           => COUNTERMAX
      )
    port map (
      clk              => clk,
      reset_n          => reset_n,
      as_chipselect    => as_chipselect,
      as_begintransfer => as_begintransfer,
      as_address       => as_address,
      as_read          => as_read,
      as_readdata      => as_readdata,
      as_write         => as_write,
      as_writedata     => as_writedata,
      halfperiod       => halfperiod_s,
      enable_forced    => ena_s,
      enable_automode  => automode_s,
      sweepack         => sweepack_s,
      sweepreq         => sweepreq_s,
      maxcurrent       => maxcurrent_s,
      maxhalfperiod    => maxhalfperiod_s);

end rtl;
--------------------------------------------------------------------------------
```

LIITE 2

VHDL-kirjastojen testitiedostot

```
#
# test_batterycharger.py
#
import sys
import vsim
import random
import math
# ================================================================================
def Out(m):
      sys.stdout.write(m)
      sys.stdout.flush()
# ================================================================================
class Base(object):
      """ Base class for initialization and lowest level functionality. """

      def __init__(self,*args,**kwargs):
            self.mod = vsim.Command(*args,**kwargs)
            # ----------------------------------------
            # Register some dynamic functionality
            # Note these are callable and thus written in capital letter!

            self.ResetSequence = vsim.tools.Reset(self.mod,"reset_n","clk")
            self.Tick = vsim.tools.Tick(self.mod,"clk")
            self.Quit = self.mod.Quit
            #----------------------------------------
            #Create avalon interface

            #Component to read/write avalon indexing with "avalon[idx]" style.
            avalonreadwrite = vsim.tools.AvalonWithSignalPrefix(self.mod,



                  delaycallable=self.Tick,



                  signalnameprefix="as_")

            #Add register name to register number mapping
            #Note avalon can also be accessed using register
            #number for example reading: myreg = self.avalon[12]
            #or write example: self.avalon[5] = 33
            self.avalon = vsim.tools.AvalonRegisterMapping(avalonreadwrite)
            self.avalon.MapRead( ReadMode = 0,

                   ReadHalfperiod = 1,

                   ReadSweep = 2,

                   ReadSweepMaxHalfperiod = 3,

                   ReadSweepMaxCurrent = 4)
            self.avalon.MapWrite( WriteMode = 0,

                    WriteHalfperiod = 1,

                    WriteSweep = 2)
            #----------------------------------------

      def SetDefaultInputs(self):
            for port in self.mod.GetInputs():
                  port.SetValue(0)
            self.mod.WriteInputs()

      def Reset(self):
            """ Reset VHDL module and set default values to inputs. """
            self.SetDefaultInputs()
            self.mod.clk = False
            self.ResetSequence()
            self.mod.ReadOutputs()
# ================================================================================
class Tester(Base):
```

```
      def TestHalfperiodSweep(self):
            """ TestHalfperiodSweep """
            assert not self.avalon.ReadSweep
            self.mod.enablecharging = 0
            self.avalon.WriteMode = 2
            self.Tick()
            assert self.avalon.ReadMode == 2
            assert self.mod.tr_sdn == 0
            self.avalon.WriteSweep = 1
            self.Tick()
            assert self.avalon.ReadSweep
            assert self.mod.tr_sdn == 1
            assert self.avalon.ReadMode == 2
            max_x = 1000 # Start at 1000 because current never goes below this
            ripple = 15
            for loops in xrange(1001):
                  f = float(loops) / 999.0 * math.pi
                  x = int(1000 + 1500 * 0.6 * abs(math.sin(f) + math.sin(2 * f - math.-
pi)))
                  max_x = max(x, max_x)
                  for t in xrange(300):
                        y = x + random.randint(-ripple, ripple)
                        self.mod.trfmcurr = y
                        self.Tick()
                  Out('.')
                  assert self.avalon.ReadSweep
                  self.Tick()
                  maxcurrent = self.avalon.ReadSweepMaxCurrent
                  assert abs(maxcurrent - max_x) <= 4, "Found maximum current too far
from average."
            for i in xrange(10):
                  self.Tick()
                  print self.avalon.ReadSweep

      def TestWriteReadMode(self):
            """ Test write and read mode. """
            self.Reset()
            for i in xrange(2000):
                  # Set mode (0 = off, 1 = on, 2 = automatic mode) randomly
                  mode = random.choice([0, 1, 2])
                  self.avalon.WriteMode = mode
                  ena =  random.choice([0, 1])
                  self.mod.enablecharging = ena
                  self.Tick()
                  assert self.avalon.ReadMode == mode
                  # If mode is off, mode read result 0, not forced on, not in automatic
mode
                  if mode == 0:
                        assert self.mod.tr_sdn == 0
                  # If mode is on, mode read result 1, charging forced on, not in auto-
matic mode
                  if mode == 1:
                        assert self.mod.tr_sdn == 1
                  # If mode is automatic, mode read result 2, charging not forced on
                  if mode == 2:
                        assert self.mod.tr_sdn == ena
                  if (i+1) % 100 == 0:
                        Out(".")

      def RunTests(self):
            Testlist = [

      self.TestHalfperiodSweep,

      self.TestWriteReadMode,

      ]
            for testfunc in Testlist:
                  Out( testfunc.__doc__.strip() + "..." )
                  testfunc()
                  Out("OK\n")
 # ============================================================================
def Test(com):
      mod = vsim.BuildModuleFromVHDLFile("./batterycharger_wrapper.vhd")
      t = Tester( com, mod )
      t.Reset()
```

```
      t.RunTests()
      t.Quit()
# ================================================================================
if __name__=="__main__":
      vsim.TestMain(Test)
# ================================================================================

#
# test_batterycharger_avalon.py
#
import sys
import vsim
import random
# ================================================================================
def Out(m):
      sys.stdout.write(m)
      sys.stdout.flush()
# ================================================================================
class Base(object):
      """ Base class for initialization and lowest level functionality. """

      def __init__(self,*args,**kwargs):
            self.mod = vsim.Command(*args,**kwargs)
            # ----------------------------------------
            # Register some dynamic functionality
            # Note these are callable and thus written in capital letter!

            self.ResetSequence = vsim.tools.Reset(self.mod,"reset_n","clk")
            self.Tick = vsim.tools.Tick(self.mod,"clk")
            self.Quit = self.mod.Quit
            #----------------------------------------
            #Create avalon interface

            #Component to read/write avalon indexing with "avalon[idx]" style.
            avalonreadwrite = vsim.tools.AvalonWithSignalPrefix(self.mod,



                  delaycallable=self.Tick,



                  signalnameprefix="as_")

            #Add register name to register number mapping
            #Note avalon can also be accessed using register
            #number for example reading: myreg = self.avalon[12]
            #or write example: self.avalon[5] = 33
            self.avalon = vsim.tools.AvalonRegisterMapping(avalonreadwrite)
            self.avalon.MapRead( ReadMode = 0,

                   ReadHalfperiod = 1,

                   ReadSweep = 2,

                   ReadSweepMaxHalfperiod = 3,

                   ReadSweepMaxCurrent = 4)
            self.avalon.MapWrite( WriteMode = 0,

                    WriteHalfperiod = 1,

                    WriteSweep = 2)
            #----------------------------------------

      def SetDefaultInputs(self):
            for port in self.mod.GetInputs():
                  port.SetValue(0)
            self.mod.WriteInputs()

      def Reset(self):
            """ Reset VHDL module and set default values to inputs. """
            self.SetDefaultInputs()
            self.mod.clk = False
            self.ResetSequence()
            self.mod.ReadOutputs()
# ================================================================================
```

```
class Tester(Base):

      def TestWriteReadHalfperiod(self):
            """ Test write and read halfperiod. """
            # Size of halfperiod size in the beginning.
            defaultcounter = (self.mod.SYSTEMCLOCKFREQUENCY/2/self.mod.DEFAULTFREQUENCY)
- 1
            # Check halfperiod in the beginning.
            assert self.avalon.ReadHalfperiod == defaultcounter
            assert self.mod.halfperiod == defaultcounter
            self.Tick()

            for i in xrange(1000):
                  # Write halfperiod with random value
                  halfperiod = random.randint(0, 50000000)
                  self.avalon.WriteHalfperiod = halfperiod
                  self.Tick()
                  # Read written halfperiod
                  assert self.avalon.ReadHalfperiod == halfperiod
                  assert self.mod.halfperiod == halfperiod
                  if (i+1) % 100 == 0:
                        Out(".")

      def TestWriteReadMode(self):
            """ Test write and read mode. """
            for i in xrange(1000):
                  # Set mode (0 = off, 1 = on, 2 = automatic mode) randomly
                  mode = random.choice([0, 1, 2])
                  self.avalon.WriteMode = mode
                  self.Tick()
                  # If mode is off, mode read result 0, not forced on, not in automatic
mode
                  if mode == 0:
                        assert self.avalon.ReadMode == 0
                        assert self.mod.enable_forced == 0
                        assert self.mod.enable_automode == 0
                  # If mode is on, mode read result 1, charging forced on, not in auto-
matic mode
                  if mode == 1:
                        assert self.avalon.ReadMode == 1
                        assert self.mod.enable_forced == 1
                        assert self.mod.enable_automode == 0
                  # If mode is automatic, mode read result 2, charging not forced on
                  if mode == 2:
                        assert self.avalon.ReadMode == 2
                        assert self.mod.enable_forced == 0
                        assert self.mod.enable_automode == 1
                  if (i+1) % 100 == 0:
                        Out(".")

      def TestWriteReadSweep(self):
            """ Test write and read sweep. """
            assert self.avalon.ReadSweep == False
            for i in xrange(1000):
                  self.mod.sweepack = random.choice([True, False])
                  self.avalon.WriteSweep = 1
                  self.Tick()
                  assert self.mod.sweepreq == (not self.mod.sweepack)
                  if (i+1) % 100 == 0:
                        Out(".")

      def TestReadSweepMaxHalfperiod(self):
            """ Test reading sweep max halfperiod. """
            for i in xrange(1000):
                  # Set max halfperiod randomly
                  maxhalfperiod = random.randint(0, 2047)
                  self.mod.maxhalfperiod = maxhalfperiod
                  self.Tick()
                  # Read max halfperiod (should match written max halfperiod)
                  assert self.avalon.ReadSweepMaxHalfperiod == maxhalfperiod
                  if (i+1) % 100 == 0:
                        Out(".")

      def TestReadMaxCurrent(self):
            """Test reading max current. """
            for i in xrange(1000):
                  # Set max current
```

```
                maxcurrent = random.randint(0, 4095)
                self.mod.maxcurrent = maxcurrent
                self.Tick()
                # Read max current (should match written max current)
                assert self.avalon.ReadSweepMaxCurrent == maxcurrent
                if (i+1) % 100 == 0:
                      Out(".")

      def RunTests(self):
            Testlist = [self.TestWriteReadHalfperiod,

      self.TestWriteReadMode,

      self.TestWriteReadSweep,

      self.TestReadSweepMaxHalfperiod,

      self.TestReadMaxCurrent]
            for testfunc in Testlist:
                  Out( testfunc.__doc__.strip() + "..." )
                  testfunc()
                  Out("OK\n")
 # ================================================================================
def Test(com):
      mod = vsim.BuildModuleFromVHDLFile("./batterycharger_avalon.vhd")
      t = Tester( com, mod )
      t.Reset()
      t.RunTests()
      t.Quit()
# ================================================================================
if __name__=="__main__":
      vsim.TestMain(Test)
# ================================================================================

#
# test_batterycharger_filter.py
#
import sys
import vsim
import random
# ================================================================================
def Out(m):
      sys.stdout.write(m)
      sys.stdout.flush()
# ================================================================================
class Base(object):
      """ Base class for initialization and lowest level functionality. """

      def __init__(self,*args,**kwargs):
            self.mod = vsim.Command(*args,**kwargs)
            # ---------------------------------------
            # Register some dynamic functionality
            # Note these are callable and thus written in capital letter!

            self.ResetSequence = vsim.tools.Reset(self.mod,"reset_n","clk")
            self.Tick = vsim.tools.Tick(self.mod,"clk")
            self.Quit = self.mod.Quit
            #---------------------------------------

      def SetDefaultInputs(self):
            for port in self.mod.GetInputs():
                  port.SetValue(0)
            self.mod.WriteInputs()

      def Reset(self):
            """ Reset VHDL module and set default values to inputs. """
            self.SetDefaultInputs()
            self.mod.clk = False
            self.ResetSequence()
            self.mod.ReadOutputs()
# ================================================================================
class Tester(Base):

      def calc(self, y, x, bits = 5):
            shifty = y >> bits
            y = y + x - shifty
            return (y, (y >> bits))
```

```
      def TestFilter(self):
            """ Test filter. """
            y = 0
            for i in xrange(1000):
                  x = 1000 + random.randint(-50, 50)
                  self.mod.unfiltered = x
                  y, yval = self.calc(y, x)
                  self.Tick()
                  assert self.mod.filtered == yval
                  print yval

      def RunTests(self):
            Testlist = [self.TestFilter]
            for testfunc in Testlist:
                  Out( testfunc.__doc__.strip() + "..." )
                  testfunc()
                  Out("OK\n")
# ============================================================================
def Test(com):
      mod = vsim.BuildModuleFromVHDLFile("./batterycharger_filter.vhd")
      t = Tester( com, mod )
      t.Reset()
      t.RunTests()
      t.Quit()
# ============================================================================
if __name__=="__main__":
      vsim.TestMain(Test)
# ============================================================================

#
# test_batterycharger_logic.py
#
import sys
import vsim
import random
import math
import pylab as p
# ============================================================================
def Out(m):
      sys.stdout.write(m)
      sys.stdout.flush()
# ============================================================================
class Base(object):
      """ Base class for initialization and lowest level functionality. """

      def __init__(self,*args,**kwargs):
            self.mod = vsim.Command(*args,**kwargs)
            # ----------------------------------------
            # Register some dynamic functionality
            # Note these are callable and thus written in capital letter!

            self.ResetSequence = vsim.tools.Reset(self.mod,"reset_n","clk")
            self.Tick = vsim.tools.Tick(self.mod,"clk")
            self.Quit = self.mod.Quit
            #----------------------------------------

      def SetDefaultInputs(self):
            for port in self.mod.GetInputs():
                  port.SetValue(0)
            self.mod.WriteInputs()

      def Reset(self):
            """ Reset VHDL module and set default values to inputs. """
            self.SetDefaultInputs()
            self.mod.clk = False
            self.ResetSequence()
            self.mod.ReadOutputs()
# ============================================================================
class Tester(Base):

      def TestHalfperiodSweep(self):
            """ Test logic module sweep state machine """
            values_in = []
            values_out = []

            self.mod.halfperiod = 750
```

```
            self.mod.sweepreq = not self.mod.sweepack # Generate sweep request
            self.Tick(3) # Run a few clock cycles to get state machine going
            max_x = 1000 # Start at 1000 because current never goes below this
            ripple = 15
            for loops in xrange(1000):
                  # Calculate average current (absolute sum of two sine waves)
                  f = float(loops) / 999.0 * math.pi
                  x = int(1000 + 2000 * 0.6 * abs(math.sin(f) + math.sin(2 * f - math.-
pi)))
                  # Check if current maximum has increased
                  max_x = max(x, max_x)
                  for t in xrange(300):
                        # Apply some ripple
                        y = x + random.randint(-ripple, ripple)
                        self.mod.trfmcurr = y
                        # and clock in new values
                        self.Tick()
                        # Save current state
                        values_in.append(y)
                        values_out.append(self.mod.maxcurrent or 1000)
                  self.Tick(2)
                  assert abs(self.mod.maxcurrent - max_x) <= 4, "Found maximum current
too far from average."
                  self.Tick()
                  Out(".")
            p.plot(p.arange(1250, 250, -1. / 300.), values_in)
            p.plot(p.arange(1250, 250, -1. / 300.), values_out)
            p.gca().set_xlim((1275, 225))
            p.gca().set_ylim((0, 4096))
            p.xlabel("Halfperiod value (clock ticks)")
            p.ylabel("Measured current (AD-converter value)")
            p.savefig("sweep.svg", dpi = 150)
            p.show()

      def TestChargingMode(self):
            """ Test charging mode (automatic, forced or sweep). """
            for i in xrange(1000):
                  # Enable, forced on?
                  enable = random.choice([True, False])
                  forced = random.choice([True, False])
                  # Set automatic mode
                  automode = random.choice([True, False])
                  self.mod.enablecharging = enable
                  self.mod.enable_forced = forced
                  self.mod.enable_automode = automode
                  self.Tick()
                  # If automatic mode is on, charging signal (tr_sdn) should be as
"enable"
                  if automode == True:
                        assert self.mod.tr_sdn == enable
                  # If forced on, charging signal (tr_sdn) should be as "forced"
                  else:
                        assert self.mod.tr_sdn == forced
            for i in xrange(1000):
                  # Enable or force on charging and se mode
                  enable = random.choice([True, False])
                  forced = random.choice([True, False])
                  automode = random.choice([True, False])
                  self.mod.enablecharging = enable
                  self.mod.enable_forced = forced
                  self.mod.enable_automode = automode
                  # Also generate sweep request, and only this will decide...
                  self.mod.sweepreq = not self.mod.sweepack
                  self.Tick()
                  # ...tr_sdn output TRUE!
                  assert self.mod.tr_sdn == True

      def RunTests(self):
            Testlist = [
                  self.TestHalfperiodSweep,
#                 self.TestChargingMode,
                  ]
            for testfunc in Testlist:
                  Out( testfunc.__doc__.strip() + "..." )
                  testfunc()
                  Out("OK\n")
# ============================================================================
```

```
def Test(com):
      mod = vsim.BuildModuleFromVHDLFile("./batterycharger_logic_wrapper.vhd")
      t = Tester( com, mod )
      t.Reset()
      t.RunTests()
      t.Quit()
# ================================================================================
if __name__=="__main__":
      vsim.TestMain(Test)
# ================================================================================

#
# test_batterychargercontrol.py
#
import sys
import vsim
import random
# ================================================================================
def Out(m):
      sys.stdout.write(m)
      sys.stdout.flush()
# ================================================================================
class Base(object):
      """ Base class for initialization and lowest level functionality. """

      def __init__(self,*args,**kwargs):
            self.mod = vsim.Command(*args,**kwargs)
            # ----------------------------------------
            # Register some dynamic functionality
            # Note these are callable and thus written in capital letter!

            self.ResetSequence = vsim.tools.Reset(self.mod,"reset_n","clk")
            self.Tick = vsim.tools.Tick(self.mod,"clk")
            self.Quit = self.mod.Quit
            #----------------------------------------

      def SetDefaultInputs(self):
            for port in self.mod.GetInputs():
                  port.SetValue(0)
            self.mod.WriteInputs()

      def Reset(self):
            """ Reset VHDL module and set default values to inputs. """
            self.SetDefaultInputs()
            self.mod.clk = False
            self.ResetSequence()
            self.mod.ReadOutputs()
# ================================================================================
class Tester(Base):

      def TestSensorsError(self):
            """ Test sensors error, charging shoul not be enabled. """
            for i in xrange(10):
                  for x in xrange(100):
                        errorstate = random.choice([2, 4, 6])
                        self.mod.sensorerror = errorstate
                        self.mod.sensorstate = 0
                        self.Tick()
                        assert self.mod.enablecharging == 0
                        self.mod.sensorstate = 1
                        self.Tick()
                        assert self.mod.enablecharging == 0
                        self.mod.sensorstate = 2
                        self.Tick()
                        assert self.mod.enablecharging == 0
                        self.mod.sensorstate = 6
                        self.Tick()
                        assert self.mod.enablecharging == 0
                  Out(".")

      def TestSensorsNotError(self):
            """ Test sensors not error, charging should be enabled. """
            for i in xrange(10):
                  for x in xrange(100):
                        self.mod.sensorerror = 0
                        sensorstate = random.choice([0, 2, 4, 6])
                        self.mod.sensorstate = sensorstate
```

```
                        self.Tick()
                        if sensorstate == 6:

      assert self.mod.enablecharging == 1
                        else:

      assert self.mod.enablecharging == 0
                  Out(".")

      def RunTests(self):
            Testlist = [self.TestSensorsError,

      self.TestSensorsNotError]
            for testfunc in Testlist:
                  Out( testfunc.__doc__.strip() + "..." )
                  testfunc()
                  Out("OK\n")
 # ================================================================================
def Test(com):
      mod = vsim.BuildModuleFromVHDLFile("./batterychargercontrol.vhd")
      t = Tester( com, mod )
      t.Reset()
      t.RunTests()
      t.Quit()
# ================================================================================
if __name__=="__main__":
      vsim.TestMain(Test)
# ================================================================================

#
# makefile
# Makefile for VHDL-simulation using GHDL
#
# ================================================================================
# Paths
# Note these are optional and not required at all.
# They are used below as shortcutss.
VHDLPATH=../../vhdllib/batterycharger
SIGNALCLOCKPATH=../../vhdllib/signalclkdomain

# Path to vsim base
VSIMPATH=../vsim

# GHDL preprocessor directives
GHDLARGS=-g --workdir=work --warn-unused --warn-binding --warn-body --warn-specs --warn-
error

# VHDLPP preprocessor binary
VHDLPP=../../bin/vhdlpp

# Source file where the simualted VHDL entity is found in.
#ENTITYSRCFILE=./batterycharger.vhd
ENTITYSRCFILE=

# All VHDL source files that need to be compiled in simulation.
# Included the entity source file here also.
#SRCFILES=$(ENTITYSRCFILE)
SRCFILES=

# Pipe file name INPUTPIPE and OUTPUTPIPE.
# Note these files are automatically created with mkfifo
# and removed when cleaning the project path.
PIPENAME=$(basename $(notdir $(ENTITYSRCFILE)))
INPUTPIPE=/tmp/$(PIPENAME)_inputpipe
OUTPUTPIPE=/tmp/$(PIPENAME)_outputpipe

# Default make target
# Note default target must be the first build target in makefile.
default: test

# Define target "clean" to clean everything
# Should call "defaultclean" target to clean everything base project creates.
clean: defaultclean

# Define target "test" to run all tests
# Automatically defined target clean cleans up all temporary and result files.
# Automatically defined target compile compiles everything.
```

```
test1:
      +make clean
      +make compile
      INPUTPIPE=$(INPUTPIPE) OUTPUTPIPE=$(OUTPUTPIPE) PYTHONPATH=$(VSIMPATH)/.. python
-B test_batterycharger_avalon.py

test2:
      +make clean
      +make compile
      INPUTPIPE=$(INPUTPIPE) OUTPUTPIPE=$(OUTPUTPIPE) PYTHONPATH=$(VSIMPATH)/.. python
-B test_batterycharger_filter.py

test3:
      +make clean
      +make compile
      INPUTPIPE=$(INPUTPIPE) OUTPUTPIPE=$(OUTPUTPIPE) PYTHONPATH=$(VSIMPATH)/.. python
-B test_batterycharger_logic.py

test4:
      +make clean
      +make compile
      INPUTPIPE=$(INPUTPIPE) OUTPUTPIPE=$(OUTPUTPIPE) PYTHONPATH=$(VSIMPATH)/.. python
-B test_batterycharger.py

test:
      +make test1 \
            ENTITYSRCFILE=./batterycharger_avalon.vhd \
            SRCFILES=./batterycharger_avalon.vhd
      +make test2 \
            ENTITYSRCFILE=./batterycharger_filter.vhd \
            SRCFILES=./batterycharger_filter.vhd
      +make test3 \
            ENTITYSRCFILE=./batterycharger_logic_wrapper.vhd \
            SRCFILES=./batterycharger_logic_wrapper.vhd \
            SRCFILES+=batterycharger_logic.vhd \
            SRCFILES+=batterycharger_filter.vhd
      +make test4 \
            ENTITYSRCFILE=./batterycharger_wrapper.vhd \
            SRCFILES=./batterycharger_wrapper.vhd \
            SRCFILES+=batterycharger.vhd \
            SRCFILES+=batterycharger_logic.vhd \
            SRCFILES+=batterycharger_filter.vhd \
            SRCFILES+=batterycharger_avalon.vhd

# Include base makefile
include ../vsim/vsim.mk
# ================================================================================
```

LIITE 3

BATTERYCHARGER NIOS-kirjasto

```
//
// batterycharger.c
// Battery charger module for using gripper board's light curtain battery charger
//
#include <string.h>
#include "batterycharger/batterycharger.h"
//------------------------------------------------------------------------------
// Initialize interface.
void BATTERYCHARGERInit(BATTERYCHARGER *batterycharger,

            uint32_t baseaddr) { // Base address of VHDL-module
  memset( batterycharger, 0, sizeof(*batterycharger) );
  batterycharger->baseaddr = baseaddr;
}
//------------------------------------------------------------------------------
//
// batterycharger.h
// Battery charger module for using gripper board's light curtain battery charger
//
#include <stdint.h>
//------------------------------------------------------------------------------
#ifndef BATTERYCHARGER_H
#define BATTERYCHARGER_H
//------------------------------------------------------------------------------
struct BATTERYCHARGER;
typedef struct BATTERYCHARGER BATTERYCHARGER;

// Initialize interface.
void BATTERYCHARGERInit(BATTERYCHARGER *batterycharger,

            uint32_t baseaddr); // Base address of VHDL-module

//------------------------------------------------------------------------------
// Enable batterycharger
static inline void BATTERYCHARGERSetModeForceOn(BATTERYCHARGER *batterycharger);

//------------------------------------------------------------------------------
// Disable batterycharger
static inline void BATTERYCHARGERSetModeForceOff(BATTERYCHARGER *batterycharger);

//------------------------------------------------------------------------------
// Set batterycharger to automatic mode
static inline void BATTERYCHARGERSetModeAuto(BATTERYCHARGER *batterycharger);

//------------------------------------------------------------------------------
// Start automatic calibration
static inline void BATTERYCHARGERCalibStart(BATTERYCHARGER *batterycharger);

//------------------------------------------------------------------------------
// Is batterycharger enabled
static inline int BATTERYCHARGERIsModeForceOn(uint32_t mode);

//------------------------------------------------------------------------------
// Is batterycharger disabled
static inline int BATTERYCHARGERIsModeForceOff(uint32_t mode);

//------------------------------------------------------------------------------
// Is batterycharger in automatic mode
static inline int BATTERYCHARGERIsModeAuto(uint32_t mode);

//------------------------------------------------------------------------------
// Is calibration in progress?
static inline int BATTERYCHARGERCalibIsBusy(BATTERYCHARGER *batterycharger);

//------------------------------------------------------------------------------
// Get frequency of highest possible calibrated current
static inline uint32_t BATTERYCHARGERCalibResultGetHalfperiod(BATTERYCHARGER *batteryc-
harger);

//------------------------------------------------------------------------------
// Get highest possible calibrated current
```

```
static inline uint32_t BATTERYCHARGERCalibResultGetCurrent(BATTERYCHARGER *batterychar-
ger);

//----------------------------------------------------------------------------
// Get mode
// Mode is one of BATTERYCHARGER_REG_MODE_xxx
static inline uint32_t BATTERYCHARGERGetMode(BATTERYCHARGER *batterycharger);

//----------------------------------------------------------------------------
// Set halfperiod
// Actual frequency is calculated from halfperiod:
// frequency = SYSTEMCLOCKFREQUENCY / 2 / halfperiod
static inline void BATTERYCHARGERSetHalfPeriod(BATTERYCHARGER *batterycharger, uint32_t
halfperiod);

//----------------------------------------------------------------------------
// Get halfperiod
// Halfperiod is the number of system clock ticks per half an output clock cycle.
// Halfperiod is calculated from system clock frequency:
// halfperiod = SYSTEMCLOCKFREQUENCY / 2 / frequency
static inline uint32_t BATTERYCHARGERGetHalfPeriod(BATTERYCHARGER *batterycharger);

//----------------------------------------------------------------------------
// Leave this include last!
#include "batterycharger/batterycharger_inline.h"
//----------------------------------------------------------------------------
#endif //BATTERYCHARGER_H
//----------------------------------------------------------------------------
//
// batterycharger_inline.h
// Battery charger module for using gripper board's light curtain battery charger
//
#include <io.h>
#include "batterycharger/batterycharger.h"
//----------------------------------------------------------------------------
#ifndef BATTERYCHARGER_INLINE_H
#define BATTERYCHARGER_INLINE_H
//----------------------------------------------------------------------------
// Internals
//----------------------------------------------------------------------------
struct BATTERYCHARGER {
  uint32_t baseaddr;
};
//----------------------------------------------------------------------------
// Batterycharger register-addresses
#define BATTERYCHARGER_REG_RW_MODE             0 // Mode register
#define BATTERYCHARGER_REG_RW_HALFPERIOD       1 // Set/get halfperiod
#define BATTERYCHARGER_REG_RW_CALIB            2 // Start/query calibration status
#define BATTERYCHARGER_REG_R_CALIB_HALFPERIOD  3 // Get halfperiod after calibration
#define BATTERYCHARGER_REG_R_CALIB_CURRENT     4 // Get current after calibration

// Batterycharger mode register values
#define BATTERYCHARGER_REG_MODE_FORCEOFF  0 // Disable charger
#define BATTERYCHARGER_REG_MODE_FORCEON   1 // Enable charger
#define BATTERYCHARGER_REG_MODE_AUTO      2 // Set charger to automatic mode

//----------------------------------------------------------------------------
static inline uint32_t BATTERYCHARGER_ReadReg(BATTERYCHARGER *batterycharger,


                   uint32_t reg) {
  return IORD(batterycharger->baseaddr,reg);
}
//----------------------------------------------------------------------------
static inline void BATTERYCHARGER_WriteReg(BATTERYCHARGER *batterycharger,


               uint32_t reg,


               uint32_t value) {
  IOWR(batterycharger->baseaddr,reg,value);
}
//----------------------------------------------------------------------------
// Public API
//----------------------------------------------------------------------------
// Force batterycharger on
```

```
static inline void BATTERYCHARGERSetModeForceOn(BATTERYCHARGER *batterycharger) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_MODE,
BATTERYCHARGER_REG_MODE_FORCEON);
}
//------------------------------------------------------------------------------
// Force batterycharger off
static inline void BATTERYCHARGERSetModeForceOff(BATTERYCHARGER *batterycharger) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_MODE,
BATTERYCHARGER_REG_MODE_FORCEOFF);
}
//------------------------------------------------------------------------------
// Set batterycharger to automatic mode
static inline void BATTERYCHARGERSetModeAuto(BATTERYCHARGER *batterycharger) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_MODE,
BATTERYCHARGER_REG_MODE_AUTO);
}
//------------------------------------------------------------------------------
// Get halfperiod that is set
static inline uint32_t BATTERYCHARGERGetHalfPeriod(BATTERYCHARGER *batterycharger) {
  return BATTERYCHARGER_ReadReg(batterycharger, BATTERYCHARGER_REG_RW_HALFPERIOD);
}
//------------------------------------------------------------------------------
// Set halfperiod
static inline void BATTERYCHARGERSetHalfPeriod(BATTERYCHARGER *batterycharger, uint32_t
halfperiod) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_HALFPERIOD, halfperiod);
}
//------------------------------------------------------------------------------
// Start automatic calibration to find the highest possible transformer current
// and its corresponding frequency
static inline void BATTERYCHARGERCalibStart(BATTERYCHARGER *batterycharger) {
  BATTERYCHARGER_WriteReg(batterycharger, BATTERYCHARGER_REG_RW_CALIB, 1);
}
//------------------------------------------------------------------------------
// Get batterycharger mode
static inline uint32_t BATTERYCHARGERGetMode(BATTERYCHARGER *batterycharger) {
  return BATTERYCHARGER_ReadReg(batterycharger, BATTERYCHARGER_REG_RW_MODE);
}
//------------------------------------------------------------------------------
// Is calibration in progress?
static inline int BATTERYCHARGERCalibIsBusy(BATTERYCHARGER *batterycharger) {
  return (BATTERYCHARGER_ReadReg(batterycharger, BATTERYCHARGER_REG_RW_CALIB) & 1) == 1;
}
//------------------------------------------------------------------------------
// Get frequency of highest possible calibrated current
static inline uint32_t BATTERYCHARGERCalibResultGetHalfperiod(BATTERYCHARGER *batteryc-
harger) {
  return BATTERYCHARGER_ReadReg(batterycharger, BATTERYCHARGER_REG_R_CALIB_HALFPERIOD);
}
//------------------------------------------------------------------------------
// Get highest possible calibrated current
static inline uint32_t BATTERYCHARGERCalibResultGetCurrent(BATTERYCHARGER *batterychar-
ger) {
  return BATTERYCHARGER_ReadReg(batterycharger, BATTERYCHARGER_REG_R_CALIB_CURRENT);
}
//------------------------------------------------------------------------------
// Check if batterycharger is forced on
static inline int BATTERYCHARGERIsModeForceOn(uint32_t mode) {
  return mode == BATTERYCHARGER_REG_MODE_FORCEON;
}
//------------------------------------------------------------------------------
// Check if batterycharger is forced off
static inline int BATTERYCHARGERIsModeForceOff(uint32_t mode) {
  return mode == BATTERYCHARGER_REG_MODE_FORCEOFF;
}
//------------------------------------------------------------------------------
// Check if batterycharger is on automatic mode
static inline int BATTERYCHARGERIsModeAuto(uint32_t mode) {
  return mode == BATTERYCHARGER_REG_MODE_AUTO;
}
//------------------------------------------------------------------------------
#endif //BATTERYCHARGER_INLINE_H
//------------------------------------------------------------------------------
```

LIITE 4

TESTER NIOS-kirjasto

```
//
// tester_app.c
// Tester application
//
#include "tester/tester.h"
#include "tester/tester_data.h"
#include "testmanager/testmanager.h"
#include "testinstance/testinstance.h"
#include "testioapi/testioapi.h"
#include "systimer/systimer.h"
#include "msp430/msp430.h"
//----------------------------------------------------------------------------
static void PrintMainMenu( TESTMANAGER *tm ) {
  TESTIOAPIPrintStr( TESTMANAGERGetIOApi(tm), "\nMain menu:\n" );
  TESTMANAGERPrintTests( tm, TESTER_CPU_VERSION ); // This must be defined in 16tes-
terconfig.mk
  TESTIOAPIPrintStr( TESTMANAGERGetIOApi(tm), "0. Quit\n" );
}
//----------------------------------------------------------------------------
void TESTERRun(TESTER *t) {
  TESTMANAGER *tm = &t->testmanager;

  PrintMainMenu(tm);

  int gettest;
  do {
      gettest = TESTIOAPIQueryRangeInteger( TESTMANAGERGetIOApi(tm), "Please select menu
item", 0, tm->tests, &(t->selectedtest) );
  } while( gettest == TESTIOAPI_QUERY_ERROR );

  if( gettest == TESTIOAPI_QUERY_CANCEL )
      return;

  if( t->selectedtest == 0 )
      return;

  TESTMANAGERSetCurrent( tm, t->selectedtest - 1 );
  TESTINSTANCE *ti = TESTMANAGERGetCurrent(tm);
  uint32_t masterversion = TESTINSTANCEGetMasterVersion(ti);
  if( masterversion != TESTER_CPU_VERSION ) {
      TESTIOAPIPrintStr( TESTMANAGERGetIOApi(tm), "Tester in different version, reboo-
ting in 500 ms.\n" );
      SYSTIMERWaitMS(500);
      MSP430RebootToVersion( &(t->msp430), masterversion );
  }

  TESTMANAGERStart(tm);

  // Do this until return flag is either done or error
  while( TESTMANAGERUpdate(tm) == TESTINSTANCE_UPDATE_CONTINUE ) {
      // Do here whatever main processing you might need
  }

  TESTMANAGERStop(tm);
}
//----------------------------------------------------------------------------
//
// tester_data.h
// Data structure for tester
//
//----------------------------------------------------------------------------
#ifndef TESTER_DATA_H
#define TESTER_DATA_H
//----------------------------------------------------------------------------
#include "uartirq/uartirq.h"
#include "testioapi/testioapi.h"
#include "testmanager/testmanager.h"
#include "testinstance/testinstance.h"
#include "msp430/msp430.h"
// DDR includes
#ifdef TESTER_DDR_ENABLED
#include "testmodddr/testmodddr.h"
```

```
#endif
// SD-Card includes
#ifdef TESTER_SDCARD_ENABLED
#include "sdcard/sdcard.h"
#include "testmodsdcard/testmodsdcard.h"
#endif
// Accelerometer includes (elevator and capsule)
#if defined(TESTER_ACCELEROMETER_ENABLED) && (defined(CPUTYPE_TESTERELEVATOR) || defi-
ned(CPUTYPE_TESTERCAPSULE))
#include "accelerometer/accelerometer.h"
#include "testmodaccelerometer/testmodaccelerometer.h"
#endif
// Sensor includes
#ifdef TESTER_SENSORS_ENABLED
#include "sensors/sensors.h"
#include "testmodsensors/testmodsensors.h"
#endif
// Radio includes (gripper, capsule, loading station)
#if defined(TESTER_RADIO_ENABLED) && (defined(CPUTYPE_TESTERGRIPPER) ||
defined(CPUTYPE_TESTERCAPSULE) || defined(CPUTYPE_TESTERLOADINGSTATION))
#include "cpusharedmemory/cpusharedmemory.h"
#include "radioctrl/radioctrlmaster.h"
#include "radiopackettransfer/radiopackettransfersharedmemory.h"
#include "testmodradio/testmodradio.h"
#endif
// Motor includes
#ifdef TESTER_MOTOR_ENABLED
#include "muxencmaster/muxencmaster.h"
#include "mc2/mc2.h"
#include "testmodmotor/testmodmotor.h"
#endif
// Tape encoder (only in elevator)
#if defined(TESTER_TAPEENCODER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
#include "tapeencoder3/tapeencoder3.h"
#include "testmodtapeencoder/testmodtapeencoder.h"
#endif
// MSP430
#ifdef TESTER_MSP430_ENABLED
#include "testmodmsp430/testmodmsp430.h"
#endif
// Temp and multi-ADC
#ifdef TESTER_TEMPANDMULTIADC_ENABLED
#include "tempandmultiadc/tempandmultiadc.h"
#include "testmodtempandmultiadc/testmodtempandmultiadc.h"
#endif
// Battery charger (only in gripper)
#if defined(TESTER_BATTERYCHARGER_ENABLED) && defined(CPUTYPE_TESTERGRIPPER)
#include "batterycharger/batterycharger.h"
#include "testmodbatterycharger/testmodbatterycharger.h"
#endif
// Power manager (only in elevator)
#if defined(TESTER_POWERMANAGER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
// #include "powermanagerelevator/powermanagerelevator.h"
#include "testmodpowermanager/testmodpowermanager.h"
#endif
// Distance sensors (only in capsule)
#if defined(TESTER_DISTANCESENSORS_ENABLED) && defined(CPUTYPE_TESTERCAPSULE)
#include "testmoddistancesensors/testmoddistancesensors.h"
#endif
// Safety edges (only in loading station)
#if defined(TESTER_SAFETYEDGES_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
#include "safetyedges/safetyedges.h"
#include "safetyedgesadc/safetyedgesadc.h"
#include "testmodsafetyedges/testmodsafetyedges.h"
#endif
// Ethernet/homeplug
#ifdef TESTER_ETHERNET_ENABLED
#include "net/all.h"
#include "niceth/niceth.h"
#include "testmodethernet/testmodethernet.h"
#endif
// Light curtains (only in loading station)
#if defined(TESTER_LIGHTCURTAINS_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
#include "testmodlightcurtains/testmodlightcurtains.h"
#endif
//----------------------------------------------------------------------------
struct TESTER {
```

```
  // Runtime state
  int32_t selectedtest; // 0=Nothing selected

  // Testing process main objects
  UARTIRQ *uartirq;
  TESTIOAPI testioapi;
  TESTMANAGER testmanager;

  // This is found in every board, and it is necessary for reboot
  MSP430 msp430;

  // ----------------------------------------
  // Test modules and their test instances
  // Note that context structures must only be reserved if the test exists in this ver-
sion
  // #if TESTER_CPU_VERSION == TESTER_xxx_MASTER
  // MYCONTEXT mycontext;
  // #endif
  // This is done to save quite a lot of memory

  // DDR
#ifdef TESTER_DDR_ENABLED
  TESTMODDDR testmodddr;
  TESTINSTANCE testinstanceddr;
#endif

  // SDCard
#ifdef TESTER_SDCARD_ENABLED
  TESTMODSDCARD testmodsdcard;
  TESTINSTANCE testinstancesdcard;
#endif

  // SCA3000 (Accelerometer)
#if defined(TESTER_ACCELEROMETER_ENABLED) && (defined(CPUTYPE_TESTERELEVATOR) || defi-
ned(CPUTYPE_TESTERCAPSULE))
#if TESTER_CPU_VERSION == TESTER_ACCELEROMETER_MASTER
  ACCELEROMETER accelerometer;
#endif
  TESTMODACCELEROMETER testmodaccelerometer;
  TESTINSTANCE testinstanceaccelerometer;
#endif

  // Sensors
#ifdef TESTER_SENSORS_ENABLED
#if TESTER_CPU_VERSION == TESTER_SENSORS_MASTER
  SENSORS sensors;
#endif
  TESTMODSENSORS testmodsensors;
  TESTINSTANCE testinstancesensors;
#endif

  // Radio
#ifdef TESTER_RADIO_ENABLED
#ifndef CPUTYPE_TESTERELEVATOR
  // Shared memory between processors
  CPUSHAREDMEMORY *cpusm;

#if TESTER_CPU_VERSION == TESTER_RADIO_MASTER
  RADIOCTRLMASTER radioctrlmaster;
#endif
  TESTMODRADIO testmodradio;
  TESTINSTANCE testinstanceradio;
#endif
#endif

  // Motors
#ifdef TESTER_MOTOR_ENABLED
#if TESTER_CPU_VERSION == TESTER_MOTOR_MASTER
  MUXENCMASTERCONTEXT *muxencmaster[MOTORCOUNT]; // Pointers only
  MC2 mc2[MOTORCOUNT];
#endif
  TESTMODMOTOR testmodmotor;
  TESTINSTANCE testinstancemotor;
#endif

  // Tape encoders, two separate test entities
#if defined(TESTER_TAPEENCODER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
```

```
#if TESTER_CPU_VERSION == TESTER_TAPEENCODER_MASTER
  TAPEENCODER3 tapeencfront;
#endif
  TESTMODTAPEENCODER testmodtapeencfront;
  TESTINSTANCE testinstancetapeencfront;

#if TESTER_CPU_VERSION == TESTER_TAPEENCODER_MASTER
  TAPEENCODER3 tapeencrear;
#endif
  TESTMODTAPEENCODER testmodtapeencrear;
  TESTINSTANCE testinstancetapeencrear;
#endif

  // MSP430
#ifdef TESTER_MSP430_ENABLED
  TESTMODMSP430 testmodmsp430;
  TESTINSTANCE testinstancemsp430;
#endif

  // Temp and multi-ADC
#ifdef TESTER_TEMPANDMULTIADC_ENABLED
#if TESTER_CPU_VERSION == TESTER_TEMPANDMULTIADC_MASTER
  TEMPANDMULTIADC tempandmultiadc;
#endif
  TESTMODTEMPANDMULTIADC testmodtempandmultiadc;
  TESTINSTANCE testinstancetempandmultiadc;
#endif

  // Battery charger
#if defined(TESTER_BATTERYCHARGER_ENABLED) && defined(CPUTYPE_TESTERGRIPPER)
#if TESTER_CPU_VERSION == TESTER_BATTERYCHARGER_MASTER
  BATTERYCHARGER batterycharger;
#endif
  TESTMODBATTERYCHARGER testmodbatterycharger;
  TESTINSTANCE testinstancebatterycharger;
#endif

  // Power manager (elevator)
#if defined(TESTER_POWERMANAGER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
  TESTMODPOWERMANAGER testmodpowermanager;
  TESTINSTANCE testinstancepowermanager;
#endif

  // Distance sensors
#if defined(TESTER_DISTANCESENSORS_ENABLED) && defined(CPUTYPE_TESTERCAPSULE)
  TESTMODDISTANCESENSORS testmoddistancesensors;
  TESTINSTANCE testinstancedistancesensors;
#endif

  // Safety edges
#if defined(TESTER_SAFETYEDGES_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
#if TESTER_CPU_VERSION == TESTER_SAFETYEDGES_MASTER
  SAFETYEDGES safetyedges;
  SAFETYEDGESADC safetyedgesadc;
#endif
  TESTMODSAFETYEDGES testmodsafetyedges;
  TESTINSTANCE testinstancesafetyedges;
#endif

  // Ethernet/homeplug
#ifdef TESTER_ETHERNET_ENABLED
#if TESTER_CPU_VERSION == TESTER_ETHERNET_MASTER
  NET net;
  NICETH niceth;
#endif
  TESTMODETHERNET testmodethernet;
  TESTINSTANCE testinstanceethernet;
#endif

  // Light curtains
#if defined(TESTER_LIGHTCURTAINS_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
  TESTMODLIGHTCURTAINS testmodlightcurtains;
  TESTINSTANCE testinstancelightcurtains;
#endif
};
//---------------------------------------------------------------------------
#endif //TESTER_DATA_H
```

```
//------------------------------------------------------------------------------
//
// tester.h
// Tester application main interface
//
//------------------------------------------------------------------------------
#ifndef TESTER_H
#define TESTER_H
//------------------------------------------------------------------------------
#include "uartirq/uartirq.h"
//------------------------------------------------------------------------------
struct TESTER;
typedef struct TESTER TESTER;
//------------------------------------------------------------------------------
// Initialize library
void TESTERInit(TESTER *tester, UARTIRQ *uartirq);

// Run application
void TESTERRun(TESTER *tester);
//------------------------------------------------------------------------------
// Note include this last
#include "tester/tester_data.h"
//------------------------------------------------------------------------------
#endif //TESTER_H
//------------------------------------------------------------------------------
//
// tester_init.c
// Initialize tester application
//
#include <assert.h>
#include <string.h> // memset()
#include "cpuctrl/cpuctrl.h"
#include "systimer/systimer.h"
#include "tester/tester.h"
#include "tester/tester_data.h"
#include "randomize/randomize.h"
#include "sys/alt_cache.h"
#include "uartirq/uartirq.h"
#include "uart.h" // For debugging only
#include "mutexconfig/mutexconfig.h"
#include "mutex/mutex.h"
#include "msp430/msp430.h"
#include "syserror/syserror.h"
#include "sysio/sysio.h"
// DDR includes
#ifdef TESTER_DDR_ENABLED
#include "testmodddr/testmodddr.h"
#endif
// SD-Card includes
#ifdef TESTER_SDCARD_ENABLED
#include "sdcard/sdcard.h"
#include "testmodsdcard/testmodsdcard.h"
#endif
// Accelerometer includes (elevator and capsule)
#if defined(TESTER_ACCELEROMETER_ENABLED) && (defined(CPUTYPE_TESTERELEVATOR) || defi-
ned(CPUTYPE_TESTERCAPSULE))
#include "accelerometer/accelerometer.h"
#include "testmodaccelerometer/testmodaccelerometer.h"
#endif
// Sensor includes
#ifdef TESTER_SENSORS_ENABLED
#include "sensors/sensors.h"
#include "testmodsensors/testmodsensors.h"
#endif
// Radio includes (gripper, capsule, loading station)
#if defined(TESTER_RADIO_ENABLED) && (defined(CPUTYPE_TESTERGRIPPER) ||
defined(CPUTYPE_TESTERCAPSULE) || defined(CPUTYPE_TESTERLOADINGSTATION))
#include "cpusharedmemory/cpusharedmemory.h"
#include "radioctrl/radioctrlmaster.h"
#include "radiopackettransfer/radiopackettransfersharedmemory.h"
#include "testmodradio/testmodradio.h"
#endif
// Motor includes
#ifdef TESTER_MOTOR_ENABLED
#include "muxencmaster/muxencmaster.h"
#include "motorenables/motorenables.h"
#include "mc2/mc2.h"
```

```
#include "testmodmotor/testmodmotor.h"
#ifdef CPUTYPE_TESTERELEVATOR
#include "brake/brake.h"
#endif
#endif
// Tape encoder (only in elevator)
#if defined(TESTER_TAPEENCODER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
#include "tapeencoder3/tapeencoder3.h"
#include "testmodtapeencoder/testmodtapeencoder.h"
#endif
// MSP430
#ifdef TESTER_MSP430_ENABLED
#include "testmodmsp430/testmodmsp430.h"
#endif
// Temp and multi-ADC
#ifdef TESTER_TEMPANDMULTIADC_ENABLED
#include "tempandmultiadc/tempandmultiadc.h"
#include "testmodtempandmultiadc/testmodtempandmultiadc.h"
#endif
// Battery charger (only in gripper)
#if defined(TESTER_BATTERYCHARGER_ENABLED) && defined(CPUTYPE_TESTERGRIPPER)
#include "batterycharger/batterycharger.h"
#include "testmodbatterycharger/testmodbatterycharger.h"
#endif
// Power manager (only in elevator)
#if defined(TESTER_POWERMANAGER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
#include "powermanagerelevator/powermanagerelevator.h"
#include "testmodpowermanager/testmodpowermanager.h"
#endif
// Distance sensors (only in capsule)
#if defined(TESTER_DISTANCESENSORS_ENABLED) && defined(CPUTYPE_TESTERCAPSULE)
#include "testmoddistancesensors/testmoddistancesensors.h"
#endif
// Safety edges (only in loading station)
#if defined(TESTER_SAFETYEDGES_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
#include "safetyedgesadc/safetyedgesadc.h"
#include "safetyedges/safetyedges.h"
#include "testmodsafetyedges/testmodsafetyedges.h"
#endif
// Light curtains (only in loading station)
#if defined(TESTER_LIGHTCURTAINS_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
#include "lightcurtains/lightcurtains.h"
#include "testmodlightcurtains/testmodlightcurtains.h"
#endif
// UPS (only in loading station)
#ifdef CPUTYPE_TESTERLOADINGSTATION
#include "ups/ups.h"
#endif
// Ethernet/homeplug
#ifdef TESTER_ETHERNET_ENABLED
#include "net/all.h"
#include "niceth/niceth.h"
#include "nic/nic.h"
#include "testmodethernet/testmodethernet.h"
#endif
//-----------------------------------------------------------------------------
//static void InitUART(TESTER *tester) {
  //UARTIRQInit( &tester->uartirq, UART_TESTER_BASE, UART_TESTER_IRQ );
//}
//-----------------------------------------------------------------------------
static void InitSystemTimer(TESTER *tester) {
  SYSTIMERInit();
}
//-----------------------------------------------------------------------------
// Ugly hack!
// This way we provide access to specific UART.

static UARTIRQ *testeruart = NULL;

// Returns non-zero if new data is readable.
//static int UARTCharReady() {
//  if( testeruart )
//    return
//}

static inline int UARTIsReadable() {
  return UARTIRQIsCharReadable(testeruart);
```

```
}

// Returns next letter from receive buffer (0 if new data was not available)
static inline char UARTGetChar() {
  return UARTIRQGetChar(testeruart);
}

// Returns zero if buffer is full, non-zero if data was sent.
static inline int UARTPutChar(char c) {
  return UARTIRQSendBuffer( testeruart, &c, 1 );
}

static void InitTestIOAPI(TESTER *tester) {
  testeruart = tester->uartirq;
  TESTIOAPIInit( &tester->testioapi,
                        UARTIsReadable,
                        UARTGetChar,
                        UARTPutChar );
}
//-----------------------------------------------------------------------------
static void InitTestManager(TESTER *tester) {
  TESTMANAGERInit( &tester->testmanager, &tester->testioapi );
}
//-----------------------------------------------------------------------------
static void InitMSP430Base(TESTER *tester) {
  MSP430Init( &tester->msp430, MSP_BOOT_INST_BASE ); // MSP430BASEADDRESS
}
//-----------------------------------------------------------------------------
static void InitSysError(TESTER *tester) {
  SYSERRORInit( SYSERROR_INST_BASE );
}
//-----------------------------------------------------------------------------
static void InitSysIO(TESTER *tester) {
  SYSIOInit( SYSIO_INST_BASE );
}
//-----------------------------------------------------------------------------
#ifdef CPUTYPE_TESTERLOADINGSTATION
static void InitUPS(TESTER *tester) {
  // Loading station needs to always enable UPS for testing purposes
  // Note that the context is not needed anywhere else, only started once
  UPS ups;
  UPSInit( &ups, UPS_INST_BASE );
  UPSOn(&ups);
  // uint32_t halfperiod = UPSGetHalfperiod(&ups);
  // TESTIOAPIPrintf( &(tester.testioapi), "UPS Enabled at halfperiod: %d\n", halfperiod
);
}
#endif
//-----------------------------------------------------------------------------
#ifdef TESTER_DDR_ENABLED
static void InitDDR(TESTER *tester) {
  TESTINSTANCEInit( &tester->testinstanceddr, TESTER_DDR_MASTER );

  TESTMODDDRInit( &tester->testmodddr,
                         &tester->testinstanceddr,
                         &tester->testioapi,
                         ALTMEMDDR_BASE, ALTMEMDDR_SPAN );

#if TESTER_CPU_VERSION == TESTER_DDR_MASTER
  TESTMODDDRSetupMenu( &tester->testmodddr );

  TESTMODDDRSetupOnline( &tester->testinstanceddr );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstanceddr );
}
#endif
//-----------------------------------------------------------------------------
#ifdef TESTER_SDCARD_ENABLED
static void InitSDCard(TESTER *tester) {
  TESTINSTANCEInit( &tester->testinstancesdcard, TESTER_SDCARD_MASTER );

  TESTMODSDCARDInit( &tester->testmodsdcard,

        &tester->testinstancesdcard,

        &tester->testioapi );
```

```
#if TESTER_CPU_VERSION == TESTER_SDCARD_MASTER
  TESTMODSDCARDSetupMenu( &tester->testmodsdcard );

  TESTMODSDCARDSetupOnline( &tester->testinstancesdcard );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancesdcard );
}
#endif
//-------------------------------------------------------------------------------
#if defined(TESTER_ACCELEROMETER_ENABLED) && (defined(CPUTYPE_TESTERELEVATOR) || defi-
ned(CPUTYPE_TESTERCAPSULE))
static void InitAccelerometer(TESTER *tester) {
#if TESTER_CPU_VERSION == TESTER_ACCELEROMETER_MASTER
  ACCELEROMETERInit( &tester->accelerometer,

       SCA3000V2_INST_BASE );
#endif

  TESTINSTANCEInit( &tester->testinstanceaccelerometer, TESTER_ACCELEROMETER_MASTER );

  TESTMODACCELEROMETERInit( &tester->testmodaccelerometer,

                 &tester->testinstanceaccelerometer,

                 &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_ACCELEROMETER_MASTER
  TESTMODACCELEROMETERSetupContext( &tester->testmodaccelerometer, &tester->accelerome-
ter );

  TESTMODACCELEROMETERSetupMenu( &tester->testmodaccelerometer );

  TESTMODACCELEROMETERSetupOnline( &tester->testinstanceaccelerometer );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstanceaccelerometer );
}
#endif
//-------------------------------------------------------------------------------
#ifdef TESTER_SENSORS_ENABLED
static void InitSensors(TESTER *tester) {
#if TESTER_CPU_VERSION == TESTER_SENSORS_MASTER
  SENSORSInit( &tester->sensors, SENSORS_INST_BASE );
#endif

  TESTINSTANCEInit( &tester->testinstancesensors, TESTER_SENSORS_MASTER );

  TESTMODSENSORSInit( &tester->testmodsensors,

        &tester->testinstancesensors,

        &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_SENSORS_MASTER
  TESTMODSENSORSSetupContext( &tester->testmodsensors, &tester->sensors );

  TESTMODSENSORSSetupMenu( &tester->testmodsensors );

  TESTMODSENSORSSetupOnline( &tester->testinstancesensors );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancesensors );
}
#endif
//-------------------------------------------------------------------------------
#ifdef TESTER_RADIO_ENABLED
#ifndef CPUTYPE_TESTERELEVATOR
static void InitSharedMemory(TESTER *tester) {
  // Flush cache
  alt_dcache_flush_all();

  // Init shared memory
  tester->cpusm = (CPUSHAREDMEMORY *)alt_remap_uncached( (void *)MEM_SHARED_BASE,
MEM_SHARED_SPAN );
  memset( tester->cpusm, 0, MEM_SHARED_SPAN );
```

```
  alt_dcache_flush_all();
  TESTIOAPIPrintf( &(tester->testioapi),
                          "Shared memory initialized, allocated: %d, used: %d.\n",
                          MEM_SHARED_SPAN, sizeof(CPUSHAREDMEMORY) );
}
//-----------------------------------------------------------------------------
static void InitRadioConfig(TESTER *tester) {
  volatile CPUSHAREDMEMORY *cpusm = tester->cpusm;
  // APPCONFIG *appconfig = COMCPUDATAGet(appconfig);

  RADIOCTRLSHAREDMEMORY *radioctrlsm = (RADIOCTRLSHAREDMEMORY *)&cpusm->radiodata.ra-
dioctrl;
  RADIOCTRLSHAREDMEMORYInit( radioctrlsm ); // Zeroes out radio config memory

#if TESTER_CPU_VERSION == TESTER_RADIO_MASTER
  RADIOCTRLMASTERInit( &(tester->radioctrlmaster), radioctrlsm, MUTEXCONFIG_MASK_RADIO-
MEM );

  RADIOCTRLMASTERConfigSetGroupID( &(tester->radioctrlmaster), 0x123456 );
  /*RADIOCTRLMASTERConfigSetFreq2( &(tester->radioctrlmaster), 0x59 );
  RADIOCTRLMASTERConfigSetFreq1( &(tester->radioctrlmaster), 0x81 );
  RADIOCTRLMASTERConfigSetFreq0( &(tester->radioctrlmaster), 0x2f );*/
  RADIOCTRLMASTERConfigSetFreq( &(tester->radioctrlmaster), 0x59812f );

  // Update configuration version
  RADIOCTRLMASTERRequestConfigure( &(tester->radioctrlmaster) );

  // Start radio
  RADIOCTRLMASTERRequestStart( &(tester->radioctrlmaster) );
  TESTIOAPIPrintf( &(tester->testioapi), "Initialized radio configuration.\n" );
#endif
}
//-----------------------------------------------------------------------------
static void InitPacketTransferMemory(TESTER *tester) {
  volatile CPUSHAREDMEMORY *cpusm = tester->cpusm;

  RADIOPACKETTRANSFERSHAREDMEMORY *packettransferapptoradio = (RADIOPACKETTRANSFERSHA-
REDMEMORY *)&(cpusm->radiodata.packettransferapptoradio);
  RADIOPACKETTRANSFERSHAREDMEMORY *packettransferradiotoapp = (RADIOPACKETTRANSFERSHA-
REDMEMORY *)&(cpusm->radiodata.packettransferradiotoapp);

  void *packettransferapptoradiobuffer = (void *)&(cpusm->radiodata.packettransferappto-
radiobuffer);
  void *packettransferradiotoappbuffer = (void *)&(cpusm->radiodata.packettransferradio-
toappbuffer);

  RADIOPACKETTRANSFERSHAREDMEMORYInit( packettransferapptoradio,


         packettransferapptoradiobuffer,


         CPUSM_RADIODATA_PACKETTRANSFER_PACKETBUFFERSIZE );
  RADIOPACKETTRANSFERSHAREDMEMORYInit( packettransferradiotoapp,


         packettransferradiotoappbuffer,


         CPUSM_RADIODATA_PACKETTRANSFER_PACKETBUFFERSIZE );
  TESTIOAPIPrintf( &(tester->testioapi), "Initialized packet transfer shared
memories.\n" );
}
//-----------------------------------------------------------------------------
static void InitLightCurtainMemory(TESTER *tester) {
  volatile CPUSHAREDMEMORY *cpusm = tester->cpusm;

  LCSSHAREDMEMORY *lcssm = (LCSSHAREDMEMORY *)&cpusm->radiodata.lightcurtainscanner;
  LCDSHAREDMEMORY *lcdsm = (LCDSHAREDMEMORY *)&cpusm->radiodata.lightcurtaindetector;

  void *lcsbuffer = (void *)&cpusm->radiodata.lightcurtainscannerbuffer;
  void *shared = (void *)alt_remap_uncached( lcsbuffer, CPUSM_RADIODATA_LIGHTCUR-
TAINSCANNER_BUFFERSIZE );

  LCSSHAREDMEMORYInit( lcssm, shared, CPUSM_RADIODATA_LIGHTCURTAINSCANNER_BUFFERSIZE );
  LCDSHAREDMEMORYInit(lcdsm);
```

```
}
//---------------------------------------------------------------------------
static void InitCollisionDetectorMemory(TESTER *tester) {
  volatile CPUSHAREDMEMORY *cpusm = tester->cpusm;
  // APPCONFIG *appconfig = COMCPUDATAGet(appconfig);

  COLLISIONDETECTORSHAREDMEMORY *cdsm = (COLLISIONDETECTORSHAREDMEMORY *)&cpusm->radio-
data.collisiondetector;

  COLLISIONDETECTORSHAREDMEMORYInit(cdsm);
}
//---------------------------------------------------------------------------
static void InitWeighingScaleDetectorMemory(TESTER *tester) {
  volatile CPUSHAREDMEMORY *cpusm = tester->cpusm;

  WEIGHINGSCALESHAREDMEMORY *wssm = (WEIGHINGSCALESHAREDMEMORY *)&cpusm->radiodata.weig-
hingscale;

  WEIGHINGSCALESHAREDMEMORYInit(wssm);
}
//---------------------------------------------------------------------------
static void InitRadioTestMod(TESTER *tester) {
  volatile CPUSHAREDMEMORY *cpusm = tester->cpusm;

  RADIOPACKETTRANSFERSHAREDMEMORY *packettransferapptoradio = (RADIOPACKETTRANSFERSHA-
REDMEMORY *)&(cpusm->radiodata.packettransferapptoradio);
  RADIOPACKETTRANSFERSHAREDMEMORY *packettransferradiotoapp = (RADIOPACKETTRANSFERSHA-
REDMEMORY *)&(cpusm->radiodata.packettransferradiotoapp);
  LCSSHAREDMEMORY *lcssm = (LCSSHAREDMEMORY *)&cpusm->radiodata.lightcurtainscanner;
  LCDSHAREDMEMORY *lcdsm = (LCDSHAREDMEMORY *)&cpusm->radiodata.lightcurtaindetector;
  COLLISIONDETECTORSHAREDMEMORY *cdsm = (COLLISIONDETECTORSHAREDMEMORY *)&cpusm->radio-
data.collisiondetector;
  WEIGHINGSCALESHAREDMEMORY *wssm = (WEIGHINGSCALESHAREDMEMORY *)&cpusm->radiodata.weig-
hingscale;

  TESTINSTANCEInit( &tester->testinstanceradio, TESTER_RADIO_MASTER );

  TESTMODRADIOInit( &tester->testmodradio,

      &tester->testinstanceradio,

      &tester->testioapi,

      packettransferapptoradio,

      packettransferradiotoapp,

      lcssm,

      lcdsm,

      cdsm,

      wssm,

      MUTEXCONFIG_MASK_RADIOMEM );

#if TESTER_CPU_VERSION == TESTER_RADIO_MASTER
  TESTMODRADIOSetupContext( &tester->testmodradio, &tester->radioctrlmaster );

  TESTMODRADIOSetupMenu( &tester->testmodradio );

  TESTMODRADIOSetupOnline( &tester->testinstanceradio );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstanceradio );
}
//---------------------------------------------------------------------------
void InitRadio(TESTER* tester) {
  CPUCTRL cpuctrlradio;
  CPUCTRLInit(&cpuctrlradio, CPUCTRL_RADIO_INST_BASE);
  CPUCTRLStart(&cpuctrlradio);
}
#endif // No radio or anything radio-related in elevator
#endif
//---------------------------------------------------------------------------
static void InitMutex(TESTER *tester) {
```

```
  MUTEXInit( MUTEX_INST_BASE, MUTEXCONFIG_CHANNEL_COMCPU );
}
//-----------------------------------------------------------------------------
#ifdef TESTER_MOTOR_ENABLED
static void InitMotor(TESTER *tester) {
#if TESTER_CPU_VERSION == TESTER_MOTOR_MASTER
  uint32_t i;
  // By default offset to roughly middle, since motors are unidentified
  for( i = 0; i < MOTORCOUNT; i++ )
      tester->muxencmaster[i] = MUXENCMASTERInit( i, ALTMEMDDR_BASE, ALTMEMDDR_BASE,
2048, 2048 ); // No tables of any kind
  // Initialize as many MC2 as there are motors
  MC2Init( &tester->mc2[0], MC2_0_BASE );
  MC2Init( &tester->mc2[1], MC2_1_BASE );
#if MOTORCOUNT > 2
  MC2Init( &tester->mc2[2], MC2_2_BASE );
  MC2Init( &tester->mc2[3], MC2_3_BASE );
#if MOTORCOUNT > 4
  MC2Init( &tester->mc2[4], MC2_4_BASE );
  MC2Init( &tester->mc2[5], MC2_5_BASE );
  MC2Init( &tester->mc2[6], MC2_6_BASE );
  MC2Init( &tester->mc2[7], MC2_7_BASE );
#endif
#endif

  // Configure motor controllers
  for( i = 0; i < MOTORCOUNT; i++ )
      MC2Configure( &tester->mc2[i], 500, -50, 50, 25 );

  // Initialize motorenables controller
  MOTORENABLESInit( MOTORENABLES_INST_BASE );

#ifdef CPUTYPE_TESTERELEVATOR
  // Initialize brake module, if we are elevator
  BRAKEInit();
#endif
#endif

  TESTINSTANCEInit( &tester->testinstancemotor, TESTER_MOTOR_MASTER );

  TESTMODMOTORInit( &tester->testmodmotor,

      &tester->testinstancemotor,

      &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_MOTOR_MASTER
  TESTMODMOTORSetupContext( &tester->testmodmotor,

                  tester->muxencmaster,

                  tester->mc2,

                  &tester->msp430 );

  TESTMODMOTORSetupMenu( &tester->testmodmotor );

  TESTMODMOTORSetupOnline( &tester->testinstancemotor );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancemotor );
}
#endif
//-----------------------------------------------------------------------------
// Only elevator has tape encoders
#if defined(TESTER_TAPEENCODER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
static const char* strfront = "Front";
static const char* strrear = "Rear";
static void InitTapeEncoders(TESTER *tester) {
#if TESTER_CPU_VERSION == TESTER_TAPEENCODER_MASTER
  TAPEENC3Init( &tester->tapeencfront, TAPEENCODER3_FRONT_BASE );
  TAPEENC3Init( &tester->tapeencrear, TAPEENCODER3_REAR_BASE );
#endif

  TESTINSTANCEInit( &tester->testinstancetapeencfront, TESTER_TAPEENCODER_MASTER );
  TESTINSTANCEInit( &tester->testinstancetapeencrear, TESTER_TAPEENCODER_MASTER );
```

```
  TESTINSTANCESetExtraDescription( &tester->testinstancetapeencfront, strfront );
  TESTINSTANCESetExtraDescription( &tester->testinstancetapeencrear, strrear );

  TESTMODTAPEENCODERInit( &tester->testmodtapeencfront,

              &tester->testinstancetapeencfront,

              &tester->testioapi );
  TESTMODTAPEENCODERInit( &tester->testmodtapeencrear,

              &tester->testinstancetapeencrear,

              &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_TAPEENCODER_MASTER
  TESTMODTAPEENCODERSetupContext( &tester->testmodtapeencfront, &tester->tapeencfront );
  TESTMODTAPEENCODERSetupContext( &tester->testmodtapeencrear, &tester->tapeencrear );

  TESTMODTAPEENCODERSetupMenu( &tester->testmodtapeencfront );
  TESTMODTAPEENCODERSetupMenu( &tester->testmodtapeencrear );

  TESTMODTAPEENCODERSetupOnline( &tester->testinstancetapeencfront );
  TESTMODTAPEENCODERSetupOnline( &tester->testinstancetapeencrear );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancetapeencfront );
  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancetapeencrear );
}
#endif
//---------------------------------------------------------------------------
#ifdef TESTER_MSP430_ENABLED
static void InitMSP430(TESTER *tester) {
  TESTINSTANCEInit( &tester->testinstancemsp430, TESTER_MSP430_MASTER );

  TESTMODMSP430Init( &tester->testmodmsp430,

       &tester->testinstancemsp430,

       &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_MSP430_MASTER
  TESTMODMSP430SetupContext( &tester->testmodmsp430, &tester->msp430 );

  TESTMODMSP430SetupMenu( &tester->testmodmsp430 );

  TESTMODMSP430SetupOnline( &tester->testinstancemsp430 );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancemsp430 );
}
#endif
//---------------------------------------------------------------------------
#ifdef TESTER_TEMPANDMULTIADC_ENABLED
static void InitTempAndMultiADC(TESTER *tester) {
#if TESTER_CPU_VERSION == TESTER_TEMPANDMULTIADC_MASTER
  TEMPANDMULTIADCInit( &tester->tempandmultiadc, TEMPANDMULTIADC_INST_BASE );
#endif

  TESTINSTANCEInit( &tester->testinstancetempandmultiadc,
TESTER_TEMPANDMULTIADC_MASTER );

  TESTMODTEMPANDMULTIADCInit( &tester->testmodtempandmultiadc,

                   &tester->testinstancetempandmultiadc,

                   &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_TEMPANDMULTIADC_MASTER
  TESTMODTEMPANDMULTIADCSetupContext( &tester->testmodtempandmultiadc, &tester->tempand-
multiadc );

  TESTMODTEMPANDMULTIADCSetupMenu( &tester->testmodtempandmultiadc );

  TESTMODTEMPANDMULTIADCSetupOnline( &tester->testinstancetempandmultiadc );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancetempandmultiadc );
```

```
}
#endif
//-------------------------------------------------------------------------------
#if defined(TESTER_BATTERYCHARGER_ENABLED) && defined(CPUTYPE_TESTERGRIPPER)
static void InitBatteryCharger(TESTER *tester) {
#if TESTER_CPU_VERSION == TESTER_BATTERYCHARGER_MASTER
  BATTERYCHARGERInit( &tester->batterycharger, BATTERYCHARGER_INST_BASE );
#endif

  TESTINSTANCEInit( &tester->testinstancebatterycharger, TESTER_BATTERYCHARGER_MASTER );

  TESTMODBATTERYCHARGERInit( &tester->testmodbatterycharger,

                  &tester->testinstancebatterycharger,

                  &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_BATTERYCHARGER_MASTER
  TESTMODBATTERYCHARGERSetupContext( &tester->testmodbatterycharger, &tester->batteryc-
harger );

  TESTMODBATTERYCHARGERSetupMenu( &tester->testmodbatterycharger );

  TESTMODBATTERYCHARGERSetupOnline( &tester->testinstancebatterycharger );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancebatterycharger );
}
#endif
//-------------------------------------------------------------------------------
#if defined(TESTER_POWERMANAGER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
static void InitPowerManager(TESTER *tester) {
  // This does not need a context, so we just initialize it here
  POWERMANAGERELEVATORInit(POWERMANAGERELEVATOR_INST_BASE);

  TESTINSTANCEInit( &tester->testinstancepowermanager, TESTER_POWERMANAGER_MASTER );

  TESTMODPOWERMANAGERInit( &tester->testmodpowermanager,

               &tester->testinstancepowermanager,

               &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_POWERMANAGER_MASTER
  TESTMODPOWERMANAGERSetupMenu( &tester->testmodpowermanager );

  TESTMODPOWERMANAGERSetupOnline( &tester->testinstancepowermanager );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancepowermanager );
}
#endif
//-------------------------------------------------------------------------------
#if defined(TESTER_DISTANCESENSORS_ENABLED) && defined(CPUTYPE_TESTERCAPSULE)
static void InitDistanceSensors(TESTER *tester) {
  // Distance sensors does not have a nioslib, instead it uses SysIO directly
  TESTINSTANCEInit( &tester->testinstancedistancesensors,
TESTER_DISTANCESENSORS_MASTER );

  TESTMODDISTANCESENSORSInit( &tester->testmoddistancesensors,

                   &tester->testinstancedistancesensors,

                   &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_DISTANCESENSORS_MASTER
  TESTMODDISTANCESENSORSSetupMenu( &tester->testmoddistancesensors );

  TESTMODDISTANCESENSORSSetupOnline( &tester->testinstancedistancesensors );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancedistancesensors );
}
#endif
//-------------------------------------------------------------------------------
#if defined(TESTER_SAFETYEDGES_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
static void InitSafetyEdges(TESTER *tester) {
```

```
#if TESTER_CPU_VERSION == TESTER_SAFETYEDGES_MASTER
  SAFETYEDGESADCInit( &tester->safetyedgesadc, SAFETYEDGESADC_INST_BASE );
  SAFETYEDGESInit( &tester->safetyedges, SAFETYEDGES_INST_BASE );
#endif

  TESTINSTANCEInit( &tester->testinstancesafetyedges, TESTER_SAFETYEDGES_MASTER );

  TESTMODSAFETYEDGESInit( &tester->testmodsafetyedges,

              &tester->testinstancesafetyedges,

              &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_SAFETYEDGES_MASTER
  TESTMODSAFETYEDGESSetupContext( &tester->testmodsafetyedges, &tester->safetyedgesadc,
&tester->safetyedges );

  TESTMODSAFETYEDGESSetupMenu( &tester->testmodsafetyedges );

  TESTMODSAFETYEDGESSetupOnline( &tester->testinstancesafetyedges );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancesafetyedges );
}
#endif
//------------------------------------------------------------------------------
#ifdef TESTER_ETHERNET_ENABLED
static void InitEthernet(TESTER *tester) {
#if TESTER_CPU_VERSION == TESTER_ETHERNET_MASTER
  // Set some MAC-address initially
  uint8_t mymac[6];
  mymac[0] = 0x76;
  mymac[1] = 0x19;
  mymac[2] = 0x07;
  mymac[3] = 0x0F;
  mymac[4] = 0x00;
  mymac[5] = 0x02;

  uint32_t i;
  // Reset ethernet chip
  for( i = 0; i < 10000; i++ )
      SYSIOSetEthernetReset();

  for( i = 0; i < 10000; i++ )
      SYSIOUnsetEthernetReset();

  for( i = 0; i < 10000; i++ )
      SYSIOSetEthernetReset();

  SYSIOUnsetEthernetReset();

  // Initialize NIC Ethernet controller
  NICETHInit( &tester->niceth,
                    TSE_MAC_BASE,
#ifdef NET_NIC_TRIPLESPEEDETHERNET_HOMEPLUG
                    HOMEPLUGAFE_0_BASE,
#endif
                    SGDMA_RX_BASE, SGDMA_RX_NAME,
                    SGDMA_TX_BASE, SGDMA_TX_NAME,
                    mymac );
  // NICETHSetTxRxContext( &niceth, (void*)&servicenow );
  // NICETHSetRxCallback( &niceth, ReceivedPacket );

  // Initialize net with some initial configuration
  // IP:      192.168.0.5
  // Netmask: 255.255.255.0
  // Gateway: 192.168.0.230
  // NIC callback context is internal to all NICxxx devices
  NETInit( &tester->net, NICETHGetNIC(&tester->niceth), 0xC0A80005, 0xFFFFFF00,
0xC0A800E6 );

  // Set callback
  //NETSetUserContext( &net, (void*)&msp );
  //NETSetUDPIPCallback( &net, UDPIPProcess );
#endif

  TESTINSTANCEInit( &tester->testinstanceethernet, TESTER_ETHERNET_MASTER );
```

```
  TESTMODETHERNETInit( &tester->testmodethernet,

         &tester->testinstanceethernet,

         &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_ETHERNET_MASTER
  TESTMODETHERNETSetupContext( &tester->testmodethernet, &tester->net );

  TESTMODETHERNETSetupMenu( &tester->testmodethernet );

  TESTMODETHERNETSetupOnline( &tester->testinstanceethernet );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstanceethernet );
}
#endif
//---------------------------------------------------------------------------
#if defined(TESTER_LIGHTCURTAINS_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
static void InitLightCurtains(TESTER *tester) {
  // This does not need a context, so we just initialize it here
  LIGHTCURTAINSInit(AVALON_LIGHTCURTAINS_INST_BASE);

  TESTINSTANCEInit( &tester->testinstancelightcurtains, TESTER_LIGHTCURTAINS_MASTER );

  TESTMODLIGHTCURTAINSInit( &tester->testmodlightcurtains,

                 &tester->testinstancelightcurtains,

                 &tester->testioapi );

#if TESTER_CPU_VERSION == TESTER_LIGHTCURTAINS_MASTER
  TESTMODLIGHTCURTAINSSetupOnline( &tester->testinstancelightcurtains );
#endif

  TESTMANAGERAddInstance( &tester->testmanager, &tester->testinstancelightcurtains );
}
#endif
//---------------------------------------------------------------------------
void TESTERInit(TESTER *tester, UARTIRQ *uartirq) {
  // Clear memory
  memset( tester, 0, sizeof(*tester) );

  // Setup UART IRQ
  tester->uartirq = uartirq;

  // ----------------------------------------
  // Initialize all modules
  typedef void INITFUNCTION(TESTER *);
  typedef INITFUNCTION *INITFUNCTIONPTR;

  INITFUNCTIONPTR initlist[] = {
      // System devices
      //InitUART,
                  // UART
      InitSystemTimer,
            // System timer
      InitMutex,
      // Test control components
      InitTestIOAPI,
      InitTestManager,
#ifdef CPUTYPE_TESTERLOADINGSTATION
      // UPS, needed by loading station testing (because we cannot use 230V)
      InitUPS,
#endif
      // MSP430, needed for rebooting
      InitMSP430Base,
      // SysError for possible error states
      InitSysError,
      // SysIO for system IO-controller (needed by distance sensors)
      InitSysIO
      // Test cases and related components
#ifdef TESTER_DDR_ENABLED
      , InitDDR
                        // DDR
#endif
```

```
#ifdef TESTER_SDCARD_ENABLED
      , InitSDCard
                        // SDCard
#endif
#ifdef TESTER_SENSORS_ENABLED
      , InitSensors
                        // AMR sensors
#endif
#ifdef TESTER_MOTOR_ENABLED
      , InitMotor
                        // Motor
#endif
#if defined(TESTER_ACCELEROMETER_ENABLED) && (defined(CPUTYPE_TESTERELEVATOR) || defi-
ned(CPUTYPE_TESTERCAPSULE))
      , InitAccelerometer
                        // SCA3000 Accelerometer
#endif
#if defined(TESTER_TAPEENCODER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
      , InitTapeEncoders
                        // Tape encoders (front and rear)
#endif
#ifdef TESTER_RADIO_ENABLED
#ifndef CPUTYPE_TESTERELEVATOR
      , InitSharedMemory
                        // Must-have
      , InitPacketTransferMemory
           // Initialize packet transfer memories
      , InitLightCurtainMemory
           // Initialize light curtain shared memory
      , InitCollisionDetectorMemory
      // Initialize collision detector shared memory
      , InitWeighingScaleDetectorMemory
      // Initialize weighing scale shared memory
      , InitRadioConfig
                  // Configure radio
      , InitRadioTestMod
                        // Radio test mod
      , InitRadio
                        // Start RadioCPU
#endif // No radio in elevator
#endif
#ifdef TESTER_MSP430_ENABLED
      , InitMSP430
                        // MSP430
#endif
#ifdef TESTER_TEMPANDMULTIADC_ENABLED
      , InitTempAndMultiADC
                  // Temperature and multi-ADC
#endif
#if defined(TESTER_BATTERYCHARGER_ENABLED) && defined(CPUTYPE_TESTERGRIPPER)
      , InitBatteryCharger
                  // Battery charger
#endif
#if defined(TESTER_POWERMANAGER_ENABLED) && defined(CPUTYPE_TESTERELEVATOR)
      , InitPowerManager
                        // Power manager (elevator)
#endif
#if defined(TESTER_DISTANCESENSORS_ENABLED) && defined(CPUTYPE_TESTERCAPSULE)
      , InitDistanceSensors
                  // Distance sensors
#endif
#if defined(TESTER_SAFETYEDGES_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
      , InitSafetyEdges
                  // Safety edges
#endif
#ifdef TESTER_ETHERNET_ENABLED
      , InitEthernet
                        // Ethernet
#endif
#if defined(TESTER_LIGHTCURTAINS_ENABLED) && defined(CPUTYPE_TESTERLOADINGSTATION)
      , InitLightCurtains
                        // Light curtains
#endif
  };

  // ----------------------------------------
  // Init process
```

```
  uint32_t initcount = sizeof(initlist) / sizeof(initlist[0]);
  uint32_t i;
  for( i=0; i<initcount; i++ ) {
      // Update random number generator from time
      RANDUpdateFromTimer();

      // Process next init
      INITFUNCTION *f = initlist[i];
      f(tester);

      // Update random number generator from time
      RANDUpdateFromTimer();
  }
  // ----------------------------------------
}
//-------------------------------------------------------------------------------
```

LIITE 5

TESTMANAGER NIOS-kirjasto

```
//
// testmanager.c
//
//---------------------------------------------------------------------------
#include <stdint.h>
#include <string.h> // memset()

#include "testmanager/testmanager.h"
#include "testinstance/testinstance.h"
//---------------------------------------------------------------------------
void TESTMANAGERInit( TESTMANAGER* tmcon, TESTIOAPI* ioapi ) {
  memset( tmcon, 0, sizeof(*tmcon) );

  // ----------------------------------------
  // Initialize variables
  tmcon->ioapi   = ioapi;
  tmcon->status  = 0;
  tmcon->current = 0;
  tmcon->head    = 0;
}
//---------------------------------------------------------------------------
void TESTMANAGERPrintTests( TESTMANAGER* tmcon, uint32_t myversion )
{
  TESTIOAPI *io = tmcon->ioapi;
  uint32_t i;
  TESTINSTANCE *ti = tmcon->head;
  for( i = 0; i < tmcon->tests; i++ )
  {
      TESTIOAPIPrintf( io, "%d. %s ", i + 1, TESTINSTANCEGetName(ti) );
      if( TESTINSTANCEHasExtraDescription(ti) )
        TESTIOAPIPrintf( io, "(%s) ", TESTINSTANCEGetExtraDescription(ti) );
      TESTIOAPIPrintStr( io, "Test" );

      if( TESTINSTANCEGetMasterVersion(ti) != myversion )
        TESTIOAPIPrintStr( io, " (*)\n" );
      else
        TESTIOAPIPrintStr( io, "\n" );
      ti = TESTINSTANCEGetNext(ti);
  }
}
//---------------------------------------------------------------------------
void TESTMANAGERAddInstance( TESTMANAGER* tmcon, TESTINSTANCE* ti )
{
  ti->next = 0;
  if( tmcon->head == 0 ) {
      tmcon->head = ti;
  } else {
      TESTINSTANCE* tmp = tmcon->head;
      while( TESTINSTANCEGetNext(tmp) != 0 )
        tmp = TESTINSTANCEGetNext(tmp);
      TESTINSTANCESetNext( tmp, ti );
  }
  tmcon->tests++;
}
//---------------------------------------------------------------------------
//
// testmanager.h
// TestManager
//
//---------------------------------------------------------------------------
#ifndef TESTMANAGER_H
#define TESTMANAGER_H
//---------------------------------------------------------------------------
#include "testioapi/testioapi.h"
#include "testinstance/testinstance.h"
//---------------------------------------------------------------------------
// Declare internal types
struct TESTMANAGER;
typedef struct TESTMANAGER TESTMANAGER;
typedef TESTMANAGER* TESTMANAGERPTR;

//---------------------------------------------------------------------------
```

```
// Initialize TestManager context
void TESTMANAGERInit( TESTMANAGER* tmcon, TESTIOAPI* ioapi );

// Print test instance names
void TESTMANAGERPrintTests( TESTMANAGER* tmcon, uint32_t myversion );

//-----------------------------------------------------------------------------
// Add a new test instance to test manager
void TESTMANAGERAddInstance( TESTMANAGER* tmcon, TESTINSTANCE* ti );

//-----------------------------------------------------------------------------
// Basic getter and setter functions
static inline TESTIOAPI* TESTMANAGERGetIOApi( TESTMANAGER* tmcon );
static inline int TESTMANAGERGetTestCount( TESTMANAGER* tmcon );
static inline TESTINSTANCE* TESTMANAGERGetCurrent( TESTMANAGER* tmcon );

//-----------------------------------------------------------------------------
// Change current test, only if one isn't running now
// or has some internal status set to zero, indicating it can be changed
static inline int TESTMANAGERSetCurrent( TESTMANAGER* tmcon, uint32_t current );

//-----------------------------------------------------------------------------
// Start, Stop and Update functions to be called from main
static inline void TESTMANAGERStart( TESTMANAGER* tmcon );
static inline void TESTMANAGERStop( TESTMANAGER* tmcon );
static inline int TESTMANAGERUpdate( TESTMANAGER* tmcon );

//-----------------------------------------------------------------------------
#include "testmanager/testmanager_inline.h"
//-----------------------------------------------------------------------------
#endif // TESTMANAGER_H
//-----------------------------------------------------------------------------
//
// testmanager_inline.h
//
//-----------------------------------------------------------------------------
#ifndef TESTMANAGER_INLINE_H
#define TESTMANAGER_INLINE_H
//-----------------------------------------------------------------------------
#include "testmanager/testmanager.h"
#include "testinstance/testinstance.h"
//-----------------------------------------------------------------------------
struct TESTMANAGER
{
  TESTIOAPI *ioapi;
  TESTINSTANCE *head;
  TESTINSTANCE *current;

  int status;
  int tests; // Number of tests
};
//-----------------------------------------------------------------------------
// Return io-api
static inline TESTIOAPI* TESTMANAGERGetIOApi( TESTMANAGER* tmcon )
{
  return tmcon->ioapi;
}
//-----------------------------------------------------------------------------
// Get current test count
static inline int TESTMANAGERGetTestCount( TESTMANAGER* tmcon )
{
  return tmcon->tests;
}
//-----------------------------------------------------------------------------
// Get current test instance
static inline TESTINSTANCE* TESTMANAGERGetCurrent( TESTMANAGER* tmcon )
{
  return tmcon->current;
}
//-----------------------------------------------------------------------------
// Change current test, only if one isn't running now
// or has some internal status set to zero, indicating it can be changed
static inline int TESTMANAGERSetCurrent( TESTMANAGER* tmcon, uint32_t current )
{
  uint32_t i;
  if( tmcon->status == 0 )
  {
```

```
      if( current >= tmcon->tests )
        return 0;
      tmcon->current = tmcon->head;
      for( i = 0; i < current; i++ )
        tmcon->current = tmcon->current->next;
      return 1;
  }
  return 0;
}
//---------------------------------------------------------------------------
static void TESTMANAGER_INTERNAL_PrintExtendedName( TESTIOAPI* io, TESTINSTANCE* ti,
const char* action )
{
  TESTIOAPIPrintf( io, "Test <%s", TESTINSTANCEGetName(ti) );
  if( TESTINSTANCEHasExtraDescription(ti) ) {
      TESTIOAPIPrintf( io, " (%s)>: %s\n", TESTINSTANCEGetExtraDescription(ti),
action );
  } else {
      TESTIOAPIPrintf( io, ">: %s\n", action );
  }
}
//---------------------------------------------------------------------------
static inline void TESTMANAGERStart( TESTMANAGER* tmcon )
{
  TESTINSTANCE *ti = tmcon->current;
  if( ti != 0 ) {
      TESTMANAGER_INTERNAL_PrintExtendedName( tmcon->ioapi, ti, "Start" );
      TESTINSTANCEStart(ti);
  }
}
//---------------------------------------------------------------------------
static inline void TESTMANAGERStop( TESTMANAGER* tmcon )
{
  TESTINSTANCE *ti = tmcon->current;
  if( ti != 0 ) {
      TESTMANAGER_INTERNAL_PrintExtendedName( tmcon->ioapi, ti, "Stop" );
      TESTINSTANCEStop(ti);
  }
}
//---------------------------------------------------------------------------
static inline int TESTMANAGERUpdate( TESTMANAGER* tmcon )
{
  TESTINSTANCE *ti = tmcon->current;
  if( ti != 0 ) {
      // By default let's not print the Update text because it:
      //   a) Prints out many many times (depending on the test)
      //   b) Clogs up the UART
      // TESTMANAGER_INTERNAL_PrintExtendedName( tmcon->ioapi, ti, "Update" );
      return TESTINSTANCEUpdate(ti);
  }
}
//---------------------------------------------------------------------------
#endif // TESTMANAGER_INLINE_H
//---------------------------------------------------------------------------
```

LIITE 6

TESTINSTANCE NIOS-kirjasto

```
//
// testinstance.c
//
//---------------------------------------------------------------------------
#include <stdint.h>
#include <string.h> // memset()

#include "testinstance/testinstance.h"
//---------------------------------------------------------------------------
void TESTINSTANCEInit( TESTINSTANCE* ti, uint32_t masterversion )
{
  memset( ti, 0, sizeof(*ti) );
  ti->masterversion = masterversion;
}
//---------------------------------------------------------------------------
//
// testinstance.h
// Test Instance
//
//---------------------------------------------------------------------------
#ifndef TESTINSTANCE_H
#define TESTINSTANCE_H
//---------------------------------------------------------------------------
#include <stdint.h>
//---------------------------------------------------------------------------
// Define update function return values
#define TESTINSTANCE_UPDATE_DONE     1
#define TESTINSTANCE_UPDATE_CONTINUE 2
#define TESTINSTANCE_UPDATE_ERROR    3
#define TESTINSTANCE_UPDATE_CANCEL   4
//---------------------------------------------------------------------------
// Declare internal types
typedef void        STARTFUNCTION(void*);
typedef void        STOPFUNCTION(void*);
typedef uint32_t    UPDATEFUNCTION(void*); // Return one of the values defined above
typedef const char* NAMEFUNCTION(void*);

struct TESTINSTANCE;
typedef struct TESTINSTANCE TESTINSTANCE;
typedef TESTINSTANCE* TESTINSTANCEPTR;

// Initialize test instance
void TESTINSTANCEInit( TESTINSTANCE* ti, uint32_t masterversion );

// Set functions for context and function pointers
static inline void TESTINSTANCESetContext( TESTINSTANCE* ti, void* con );
static inline void TESTINSTANCESetStart( TESTINSTANCE* ti, STARTFUNCTION *startFunc );
static inline void TESTINSTANCESetStop( TESTINSTANCE* ti, STOPFUNCTION *stopFunc );
static inline void TESTINSTANCESetUpdate( TESTINSTANCE* ti, UPDATEFUNCTION
*updateFunc );
static inline void TESTINSTANCESetName( TESTINSTANCE* ti, NAMEFUNCTION *nameFunc );

// Get functions for start, stop, update and name
// Calls the attached function pointer with context and returns result (if applicable)
static inline void TESTINSTANCEStart(TESTINSTANCE* ti);
static inline void TESTINSTANCEStop(TESTINSTANCE* ti);
static inline uint32_t TESTINSTANCEUpdate(TESTINSTANCE* ti);
static inline const char* TESTINSTANCEGetName(TESTINSTANCE* ti);

// Get/set pointer to next test instance
static inline void TESTINSTANCESetNext( TESTINSTANCE* ti, TESTINSTANCE* next );
static inline TESTINSTANCE* TESTINSTANCEGetNext(TESTINSTANCE* ti);

// Get master version number
static inline uint32_t TESTINSTANCEGetMasterVersion(TESTINSTANCE* ti);

// Get/set/has functions for extra description
static inline void TESTINSTANCESetExtraDescription( TESTINSTANCE* ti, char* extradesc-
ription );
static inline int TESTINSTANCEHasExtraDescription(TESTINSTANCE* ti);
static inline const char* TESTINSTANCEGetExtraDescription(TESTINSTANCE* ti);
//---------------------------------------------------------------------------
```

```
#include "testinstance/testinstance_inline.h"
//-----------------------------------------------------------------------------
#endif // TESTINSTANCE_H
//-----------------------------------------------------------------------------
//
// testinstance_inline.h
//
//-----------------------------------------------------------------------------
#ifndef TESTINSTANCE_INLINE_H
#define TESTINSTANCE_INLINE_H
//-----------------------------------------------------------------------------
//#include <io.h>
#include "testinstance/testinstance.h"
//-----------------------------------------------------------------------------
// Linked list
struct TESTINSTANCE
{
  void* context;

  // Callback functions
  //INITFUNCTION   *init;
  STARTFUNCTION  *start;
  STOPFUNCTION   *stop;
  UPDATEFUNCTION *update;
  NAMEFUNCTION   *name;

  // Test master version number, from #defines
  uint32_t masterversion;

  // Pointer to extra description after test name (to differentiate between multiple
instances)
  // By default this is set to zero and will not show up.
  char* extradescription;

  // Linked list pointer
  TESTINSTANCE *next;
};
//-----------------------------------------------------------------------------
static inline void TESTINSTANCESetContext( TESTINSTANCE* ti, void* con )
{
  ti->context = con;
}
//-----------------------------------------------------------------------------
static inline void TESTINSTANCESetStart( TESTINSTANCE* ti, STARTFUNCTION *startFunc )
{
  ti->start = startFunc;
}
//-----------------------------------------------------------------------------
static inline void TESTINSTANCESetStop( TESTINSTANCE* ti, STOPFUNCTION *stopFunc )
{
  ti->stop = stopFunc;
}
//-----------------------------------------------------------------------------
static inline void TESTINSTANCESetUpdate( TESTINSTANCE* ti, UPDATEFUNCTION *updateFunc )
{
  ti->update = updateFunc;
}
//-----------------------------------------------------------------------------
static inline void TESTINSTANCESetName( TESTINSTANCE* ti, NAMEFUNCTION *nameFunc )
{
  ti->name = nameFunc;
}
//-----------------------------------------------------------------------------
static inline void TESTINSTANCEStart(TESTINSTANCE* ti)
{
  if( ti->start != 0 )
      ti->start(ti->context);
}
//-----------------------------------------------------------------------------
static inline void TESTINSTANCEStop(TESTINSTANCE* ti)
{
  if( ti->stop != 0 )
      ti->stop(ti->context);
}
//-----------------------------------------------------------------------------
// If no update function is attached, return TESTINSTANCE_UPDATE_ERROR
static inline uint32_t TESTINSTANCEUpdate(TESTINSTANCE* ti)
```

```
{
  if( ti->update != 0 )
      return ti->update(ti->context);
  return TESTINSTANCE_UPDATE_ERROR;
}
//---------------------------------------------------------------------------
// If no name function is attached, return "No name"
static const char* nonamestr = "No name";

static inline const char* TESTINSTANCEGetName(TESTINSTANCE* ti)
{
  if( ti->name != 0 )
      return ti->name(ti->context);
  return nonamestr;
}
//---------------------------------------------------------------------------
static inline void TESTINSTANCESetNext( TESTINSTANCE* ti, TESTINSTANCE* next )
{
  ti->next = next;
}
//---------------------------------------------------------------------------
static inline TESTINSTANCE* TESTINSTANCEGetNext(TESTINSTANCE* ti)
{
  return ti->next;
}
//---------------------------------------------------------------------------
static inline uint32_t TESTINSTANCEGetMasterVersion(TESTINSTANCE* ti)
{
  return ti->masterversion;
}
//---------------------------------------------------------------------------
static inline void TESTINSTANCESetExtraDescription( TESTINSTANCE* ti, char* extradesc-
ription )
{
  ti->extradescription = extradescription;
}
//---------------------------------------------------------------------------
static inline int TESTINSTANCEHasExtraDescription(TESTINSTANCE* ti)
{
  return (ti->extradescription != 0);
}
//---------------------------------------------------------------------------
static inline const char* TESTINSTANCEGetExtraDescription(TESTINSTANCE* ti)
{
  return ti->extradescription;
}
//---------------------------------------------------------------------------
#endif // TESTINSTANCE_INLINE_H
//---------------------------------------------------------------------------
```

LIITE 7

TESTIOAPI NIOS-kirjasto

```
//
// testioapi.c
//
//-----------------------------------------------------------------------------
#include <stdint.h>
#include <string.h> // memset()
#include <stdarg.h> // va_list

#include "testioapi/testioapi.h"
#include "testioapi/testioapi_internal.h" // Internal functions
//-----------------------------------------------------------------------------
void TESTIOAPIInit( TESTIOAPI *ioapi,

      TESTIOAPIISCHARREADYCALLBACK chrready,

      TESTIOAPIGETCHARCALLBACK getchr,

      TESTIOAPIPUTCHARCALLBACK putchr )
{
  memset( ioapi, 0, sizeof(*ioapi) );
  ioapi->chrready = chrready;
  ioapi->getchr   = getchr;
  ioapi->putchr   = putchr;
}
//-----------------------------------------------------------------------------
// Replace this with actual avalon hardware calls later
void TESTIOAPIPutChar( TESTIOAPI *ioapi, char ch )
{
  // alt_putchar(ch);
  while( ioapi->putchr(ch) == 0 ); // Keep trying until there is enough room in the buf-
fer..
}
//-----------------------------------------------------------------------------
// Replace this with actual avalon hardware calls later
char TESTIOAPIGetChar( TESTIOAPI *ioapi )
{
  //char ch = alt_getchar();
  if( ioapi->chrready != NULL )   {
      if( ioapi->chrready() )
        return ioapi->getchr();
      else
        return 0;
  }
  return ioapi->getchr();
}
//-----------------------------------------------------------------------------
void TESTIOAPIPrintInt( TESTIOAPI *ioapi, int32_t num )
{
  // Is zero?
  if( num == 0 ) {
      TESTIOAPIPutChar( ioapi, '0' );
      return;
  }

  // Use PrintIntFixed with no sign, space as filler and zero fill size
  TESTIOAPI_INTERNAL_PrintIntFixed( ioapi, num, 0, 0, 0 );
}
//-----------------------------------------------------------------------------
void TESTIOAPIPrintBin( TESTIOAPI *ioapi, int32_t num )
{
  // Special case if value is just zero
  if( num == 0 ) {
      TESTIOAPIPutChar( ioapi, '0' );
      return;
  }

  // Use PrintBinFixed with space as filler and zero fill size
  TESTIOAPI_INTERNAL_PrintBinFixed( ioapi, num, 0, 0 );
}
//-----------------------------------------------------------------------------
void TESTIOAPIPrintHex( TESTIOAPI *ioapi, int32_t num )
{
```

```
  TESTIOAPI_INTERNAL_PrintHex( ioapi, 0, num );
}
//-----------------------------------------------------------------------------
void TESTIOAPIPrintHexSmall( TESTIOAPI *ioapi, int32_t num )
{
  TESTIOAPI_INTERNAL_PrintHex( ioapi, 1, num );
}
//-----------------------------------------------------------------------------
void TESTIOAPIPrintStr( TESTIOAPI *ioapi, const char *str )
{
  uint32_t i;
  for( i = 0; i < strlen(str); i++ )
      TESTIOAPIPutChar( ioapi, (int)str[i] );
}
//-----------------------------------------------------------------------------
static inline void TESTIOAPI_INTERNAL_ZeroStatus(void* status)
{
  // const int clrsize = 5 * sizeof(uint8_t);
  memset( status, 0, 6 );
}
//-----------------------------------------------------------------------------
void TESTIOAPIPrintf( TESTIOAPI *ioapi, const char *fmt, ... )
{
  uint32_t sLen = strlen(fmt);
  uint32_t i = 0;
  union {
      struct {
        uint8_t fillZero; //= 0;
        uint8_t fillsize; //= 0;
        uint8_t hasZero; //= 0;
        uint8_t percent; //= 0;
        uint8_t hexSmall; //= 0;
        uint8_t insertSign; //= 0;
      } stat;
      uint8_t status[6];
  } pfstat;
  TESTIOAPI_INTERNAL_ZeroStatus( (void*)pfstat.status );
  va_list vl;
  va_start( vl, fmt );
  while( i < sLen )
  {
      if( (fmt[i] != '%') && (pfstat.stat.percent == 0) )
      {
        TESTIOAPIPutChar( ioapi, (int)fmt[i] );
        i++;
      }
      else if( (fmt[i] == '%') && (pfstat.stat.percent == 0) )
      {
        pfstat.stat.percent = 1;
        i++;
      }
      else // Is either %-sign or we are processing it
      {
        // i++;
        // percent = 1;
        switch( fmt[i] )
        {
            case '%':
              TESTIOAPIPutChar( ioapi, '%' );
              pfstat.stat.percent = 0;
              i++;
              break;
            case 'c':
            {
              char ch = va_arg( vl, int );
              TESTIOAPIPutChar( ioapi, ch );
            }
              pfstat.stat.percent = 0;
              i++;
              break;
            case 's':
            {
              char* str = va_arg( vl, char* );
              TESTIOAPIPrintStr( ioapi, str );
            }
              pfstat.stat.percent = 0;
              i++;
```

```
              break;
            case 'x': // Small hex
              pfstat.stat.hexSmall = 1;
            case 'X': // Big hex
            case 'd': // Integer
            case 'i': // Integer
            case 'b': // Binary
            {
              int32_t val = va_arg( vl, int32_t );
              if( val == 0 ) { // Zero is 'special case', check it here so there's no
need later
                  TESTIOAPI_INTERNAL_PrintZero( ioapi, pfstat.stat.hasZero,
(int32_t)pfstat.stat.fillsize );
              } else {
                  if( fmt[i] == 'd' || fmt[i] == 'i' ) // Integer
                    TESTIOAPI_INTERNAL_PrintIntFixed( ioapi, val, pfstat.stat.insert-
Sign, pfstat.stat.fillZero, (int32_t)pfstat.stat.fillsize );
                  if( fmt[i] == 'x' || fmt[i] == 'X' ) // Hex
                    TESTIOAPI_INTERNAL_PrintHexFixed( ioapi, val, pfstat.stat.fillZero,
(int32_t)pfstat.stat.fillsize, pfstat.stat.hexSmall );
                  if( fmt[i] == 'b' ) // Binary
                    TESTIOAPI_INTERNAL_PrintBinFixed( ioapi, val, pfstat.stat.fillZero,
(int32_t)pfstat.stat.fillsize );
              }
              TESTIOAPI_INTERNAL_ZeroStatus( (void*)pfstat.status );
            }
              i++;
              break;
            case '+':
              pfstat.stat.insertSign = 1;
              i++;
              break;
            default:
              if( (fmt[i] == '0') && (pfstat.stat.hasZero == 0) )
              {
                  pfstat.stat.fillZero = 1;
                  pfstat.stat.hasZero = 1;
              }
              else if( (fmt[i] == '0') && (pfstat.stat.hasZero == 1) )
              {
                  pfstat.stat.fillsize *= 10;
              }
              else if( (fmt[i] >= '1') && (fmt[i] <= '9') )
              {
                  pfstat.stat.fillsize = (10 * pfstat.stat.fillsize) + (int)(fmt[i] -
'0');
                  pfstat.stat.hasZero = 1;
              }
              i++;
              break;
        }
      }
  }
  va_end(vl);
}
//--------------------------------------------------------------------------
static int TESTIOAPI_INTERNAL_GetInteger( TESTIOAPI *ioapi, int32_t def, int16_t
process_def, int32_t *number )
{
  char buffer[64];
  char ch = 0;
  uint32_t i = 0;
  uint32_t bLen = 0;
  int16_t non_ws = 0; // false
  int16_t first_char = 1; // true
  int16_t sign_found = 0;
  do {
      // Returns -1 if no new character is received
      ch = TESTIOAPIGetChar(ioapi);
      if( ch != 0 ) {
        if( ch == 'x' )
            return TESTIOAPI_QUERY_CANCEL;
        TESTIOAPIPutChar( ioapi, ch ); // Echo back
        if( process_def ) // Do we have a default value in case of enter press?
        {
            if( first_char && ch == 0x0A )
            {
```

```
              *number = def;
              return TESTIOAPI_QUERY_OK;
            }
            first_char = 0; // first_char
        }
        if( !non_ws && !(ch == ' ' || ch == '\t') )
        {
            non_ws = 1; // true;
        }
        if( non_ws ) // Non-whitespace found, start collecting numbers
        {
            if( ch == 0x08 ) // Backspace
            {
              if( i > 0 )
                  i--;
              if( i == 0 )
                  sign_found = 0;
            }
            else if( !sign_found && (ch == '-' || ch == '+') )
            {
              buffer[i] = ch;
              i++;
              sign_found = 1;
            }
            else if( (ch >= '0' && ch <= '9') )
            {
              buffer[i] = ch;
              i++;
              sign_found = 1;
            }
            else if( ch == 0x0A )
            {
              // Do nothing
            }
            else
            {
              // Could just ignore all other characters, or report error
              // TESTIOAPIPrintStr( ioapi, "Error! Not a number.\n" );
              // return TESTIOAPIQUERYERROR; // Error
            }
        }
      }
  } while( ch != 0x0A ); // While not enter
  buffer[i] = '\0';
  bLen = i;
  int32_t value = 0;
  int32_t sign = 1;
  i = 0;

  // Check sign
  if( buffer[0] == '-' ) {
      sign = -1;
      i = 1;
  } else {
      if( buffer[0] == '+' )
        i = 1;
  }

  // Check we actually have numbers
  if( buffer[i] == 0 )
      return TESTIOAPI_QUERY_ERROR;

  // Convert to integer
  while( i < bLen ) {
      int32_t d = (int32_t)(buffer[i] - '0');
      if( d < 0 || d > 9 ) // This shouldn't happen because collector already ignores
everything else
      {
        //alt_printf("\nError! Malformed number, please enter again\n");
        return TESTIOAPI_QUERY_ERROR;
      }
      value = value * 10 + d;
      i++;
  }

  *number = sign * value;
  return TESTIOAPI_QUERY_OK; // Success
```

```
}
//----------------------------------------------------------------------------
static const char* enterInt   = "Please enter an integer";
static const char* errorRange = "Error! Number out of range.\n";
int TESTIOAPIQueryInteger( TESTIOAPI *ioapi, const char *query, int32_t *number )
{
/*  if( query == 0 )
      TESTIOAPIPrintStr( ioapi, "Please enter an integer: " );
  else
      TESTIOAPIPrintf( ioapi, "%s: ", query );*/
  TESTIOAPIPrintf( ioapi, "%s: ", (query == 0) ? enterInt : query );
  return TESTIOAPI_INTERNAL_GetInteger( ioapi, 0, 0, number );
}
//----------------------------------------------------------------------------
int TESTIOAPIQueryRangeInteger( TESTIOAPI *ioapi, const char *query, int32_t min_val,
int32_t max_val, int32_t *number )
{
  *number = min_val;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      /*if( query == 0 )
        TESTIOAPIPrintf( ioapi, "Please enter an integer (%d .. %d): ", min_val, max_val
);
      else
        TESTIOAPIPrintf( ioapi, "%s (%d .. %d): ", query, min_val, max_val );*/
      TESTIOAPIPrintf( ioapi, "%s (%d .. %d): ", (query == 0) ? enterInt : query,
min_val, max_val );
      ret_val = TESTIOAPI_INTERNAL_GetInteger( ioapi, 0, 0, number );
      if( (*number >= min_val) && (*number <= max_val) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, errorRange );
  }
  return ret_val;
}
//----------------------------------------------------------------------------
int TESTIOAPIQueryIntegerDefault( TESTIOAPI *ioapi, const char *query, int32_t def,
int32_t *number )
{
  /*if( query == 0 )
      TESTIOAPIPrintf( ioapi, "Please enter an integer [%d]: ", def );
  else
      TESTIOAPIPrintf( ioapi, "%s [%d]: ", query, def );*/
  TESTIOAPIPrintf( ioapi, "%s [%d]: ", (query == 0) ? enterInt : query, def );
  return TESTIOAPI_INTERNAL_GetInteger( ioapi, def, 1, number );
}
//----------------------------------------------------------------------------
int TESTIOAPIQueryRangeIntegerDefault( TESTIOAPI *ioapi, const char *query, int32_t
min_val, int32_t max_val, int32_t def, int32_t *number )
{
  *number = min_val;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      /*if( query == 0 )
        TESTIOAPIPrintf( ioapi, "Please enter an integer (%d .. %d) [%d]: ", min_val,
max_val, def );
      else
        TESTIOAPIPrintf( ioapi, "%s (%d .. %d) [%d]: ", query, min_val, max_val,
def );*/
      TESTIOAPIPrintf( ioapi, "%s (%d .. %d) [%d]: ", (query == 0) ? enterInt : query,
min_val, max_val, def );
      ret_val = TESTIOAPI_INTERNAL_GetInteger( ioapi, def, 1, number );
      if( (*number >= min_val) && (*number <= max_val) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, errorRange );
  }
  return ret_val;
}
//----------------------------------------------------------------------------
#if 0
// Short query functions
int TESTIOAPIQueryShort( TESTIOAPI *ioapi, const char *query, int16_t *number )
```

```
{
  int32_t temp = 0;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
        TESTIOAPIPrintStr( ioapi, "Please enter a short: " );
      else
        TESTIOAPIPrintf( ioapi, "%s: ", query );
      ret_val =  TESTIOAPI_INTERNAL_GetInteger( ioapi, 0, 0, &temp );
      if( (temp >= -32768) && (temp <= 32767) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int16_t)temp;
  return ret_val;
}
//---------------------------------------------------------------------------
int TESTIOAPIQueryRangeShort( TESTIOAPI *ioapi, const char *query, int16_t min_val,
int16_t max_val, int16_t *number )
{
  int32_t temp = (int32_t)min_val;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
        TESTIOAPIPrintf( ioapi, "Please enter a short (%d .. %d): ", (int32_t)min_val,
(int32_t)max_val );
      else
        TESTIOAPIPrintf( ioapi, "%s (%d .. %d): ", query, (int32_t)min_val,
(int32_t)max_val );
      ret_val =  TESTIOAPI_INTERNAL_GetInteger( ioapi, 0, 0, &temp );
      if( (temp >= (int32_t)min_val) && (temp <= (int32_t)max_val) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int16_t)temp;
  return ret_val;
}
//---------------------------------------------------------------------------
int TESTIOAPIQueryShortDefault( TESTIOAPI *ioapi, const char *query, int16_t def,
int16_t *number )
{
  int32_t def32 = (int32_t)def;
  int32_t temp = 0;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
        TESTIOAPIPrintf( ioapi, "Please enter a short [%d]: ", def32 );
      else
        TESTIOAPIPrintf( ioapi, "%s [%d]: ", query, def32 );
      ret_val = TESTIOAPI_INTERNAL_GetInteger( ioapi, def32, 1, &temp );
      if( (temp >= -32768) && (temp <= 32767) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int16_t)temp;
  return ret_val;
}
//---------------------------------------------------------------------------
int TESTIOAPIQueryRangeShortDefault( TESTIOAPI *ioapi, const char *query, int16_t
min_val, int16_t max_val, int16_t def, int16_t *number )
{
  int32_t def32 = (int32_t)def;
  int32_t temp = (int32_t)min_val;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
```

```
        TESTIOAPIPrintf( ioapi, "Please enter a short (%d .. %d) [%d]: ",
(int32_t)min_val, (int32_t)max_val, def32 );
      else
        TESTIOAPIPrintf( ioapi, "%s (%d .. %d) [%d]: ", query, (int32_t)min_val,
(int32_t)max_val, def32 );
      ret_val = TESTIOAPI_INTERNAL_GetInteger( ioapi, def32, 1, &temp );
      if( (temp >= (int32_t)min_val) && (temp <= (int32_t)max_val) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int16_t)temp;
  return ret_val;
}
//---------------------------------------------------------------------------
// Byte query functions
int TESTIOAPIQueryByte( TESTIOAPI *ioapi, const char *query, int8_t *number )
{
  int32_t temp = 0;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
        TESTIOAPIPrintStr( ioapi, "Please enter a byte: " );
      else
        TESTIOAPIPrintf( ioapi, "%s: ", query );
      ret_val =  TESTIOAPI_INTERNAL_GetInteger( ioapi, 0, 0, &temp );
      if( (temp >= -128) && (temp <= 127) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int8_t)temp;
  return ret_val;
}
//---------------------------------------------------------------------------
int TESTIOAPIQueryRangeByte( TESTIOAPI *ioapi, const char *query, int8_t min_val, int8_t
max_val, int8_t *number )
{
  int32_t temp = (int32_t)min_val;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
        TESTIOAPIPrintf( ioapi, "Please enter a byte (%d .. %d): ", (int32_t)min_val,
(int32_t)max_val );
      else
        TESTIOAPIPrintf( ioapi, "%s (%d .. %d): ", query, (int32_t)min_val,
(int32_t)max_val );
      ret_val =  TESTIOAPI_INTERNAL_GetInteger( ioapi, 0, 0, &temp );
      if( (temp >= (int32_t)min_val) && (temp <= (int32_t)max_val) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int8_t)temp;
  return ret_val;
}
//---------------------------------------------------------------------------
int TESTIOAPIQueryByteDefault( TESTIOAPI *ioapi, const char *query, int8_t def, int8_t
*number )
{
  int32_t def32 = (int32_t)def;
  int32_t temp = 0;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
        TESTIOAPIPrintf( ioapi, "Please enter a byte [%d]: ", (int32_t)def32 );
      else
        TESTIOAPIPrintf( ioapi, "%s [%d]: ", query, (int32_t)def32 );
      ret_val = TESTIOAPI_INTERNAL_GetInteger( ioapi, def32, 1, &temp );
      if( (temp >= -128) && (temp <= 127) )
        isnumberok = 1;
```

```
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int8_t)temp;
  return ret_val;
}
//-----------------------------------------------------------------------------
int TESTIOAPIQueryRangeByteDefault( TESTIOAPI *ioapi, const char *query, int8_t min_val,
int8_t max_val, int8_t def, int8_t *number )
{
  int32_t def32 = (int32_t)def;
  int32_t temp = (int32_t)min_val;
  int16_t isnumberok = 0;
  int ret_val;
  while( !isnumberok )
  {
      if( query == 0 )
        TESTIOAPIPrintf( ioapi, "Please enter a byte (%d .. %d) [%d]: ",
(int32_t)min_val, (int32_t)max_val, def32 );
      else
        TESTIOAPIPrintf( ioapi, "%s (%d .. %d) [%d]: ", query, (int32_t)min_val,
(int32_t)max_val, def32 );
      ret_val = TESTIOAPI_INTERNAL_GetInteger( ioapi, def32, 1, &temp );
      if( (temp >= (int32_t)min_val) && (temp <= (int32_t)max_val) )
        isnumberok = 1;
      else
        TESTIOAPIPrintStr( ioapi, "Error! Number out of range.\n" );
  }
  *number = (int8_t)temp;
  return ret_val;
}
#endif
//-----------------------------------------------------------------------------
static const char* yesno       = " (y)es or (n)o";
static const char* yesnocancel = " (y)es, (n)o or (c)ancel";
static const char* errorChoice = "\nError! Choices must be either yes/no or
yes/no/cancel.\n";
int TESTIOAPIQueryConfirmation( TESTIOAPI *ioapi, const char *query, int choices, int
*select )
{
  TESTIOAPIPrintStr( ioapi, query );
  switch( choices )
  {
      case TESTIOAPI_CONFIRMCHOICES_YESNO:
        TESTIOAPIPrintf( ioapi, "%s: ", yesno );
        break;
      case TESTIOAPI_CONFIRMCHOICES_YESNOCANCEL:
        TESTIOAPIPrintf( ioapi, "%s: ", yesnocancel );
        break;
      default:
        TESTIOAPIPrintStr( ioapi, errorChoice );
        return TESTIOAPI_QUERY_ERROR;
  }
  char ch = 0;
  int ret_val;
  while( 1 )
  {
      // Returns 0 if no new character is received
      ch = TESTIOAPIGetChar(ioapi);
      if( ch != 0 )
      {
        if( ch == 'y' || ch == 'Y' )
        {
            TESTIOAPIPutChar( ioapi, ch );
            ret_val = TESTIOAPI_CONFIRM_YES;
            break;
        }
        else if( ch == 'n' || ch == 'N' )
        {
            TESTIOAPIPutChar( ioapi, ch );
            ret_val = TESTIOAPI_CONFIRM_NO;
            break;
        }
        else if( choices == TESTIOAPI_CONFIRMCHOICES_YESNOCANCEL && (ch == 'c' || ch ==
'C') )
        {
            TESTIOAPIPutChar( ioapi, ch );
```

```
            ret_val = TESTIOAPI_CONFIRM_CANCEL;
            break;
        }
      }
  }
  *select = ret_val;
  TESTIOAPIPutChar( ioapi, '\n' );
  return TESTIOAPI_QUERY_OK;
}
//-----------------------------------------------------------------------------
int TESTIOAPIQueryConfirmationDefault( TESTIOAPI *ioapi, const char *query, int choices,
int def, int *select )
{
  TESTIOAPIPrintStr( ioapi, query );
  switch( choices )
  {
      case TESTIOAPI_CONFIRMCHOICES_YESNO:
        TESTIOAPIPrintf( ioapi, "%s ", yesno );
        break;
      case TESTIOAPI_CONFIRMCHOICES_YESNOCANCEL:
        TESTIOAPIPrintf( ioapi, "%s ", yesnocancel );
        break;
      default:
        TESTIOAPIPrintStr( ioapi, errorChoice );
        return TESTIOAPI_QUERY_ERROR;
  }
  char defch = 0;
  switch( def )
  {
      case TESTIOAPI_CONFIRM_YES:
        defch = 'y';
        break;
      case TESTIOAPI_CONFIRM_NO:
        defch = 'n';
        break;
      case TESTIOAPI_CONFIRM_CANCEL:
        defch = 'c';
        break;
  }
  TESTIOAPIPrintf( ioapi, " [%c]: ", defch );
  char ch = 0;
  int ret_val;
  while( 1 )
  {
      // Returns 0 if no new character is received
      ch = TESTIOAPIGetChar(ioapi);
      if( ch != 0 )
      {
        if( ch == 'y' || ch == 'Y' )
        {
            TESTIOAPIPutChar( ioapi, ch );
            ret_val = TESTIOAPI_CONFIRM_YES;
            break;
        }
        else if( ch == 'n' || ch == 'N' )
        {
            TESTIOAPIPutChar( ioapi, ch );
            ret_val = TESTIOAPI_CONFIRM_NO;
            break;
        }
        else if( choices == TESTIOAPI_CONFIRMCHOICES_YESNOCANCEL && (ch == 'c' || ch ==
'C') )
        {
            TESTIOAPIPutChar( ioapi, ch );
            ret_val = TESTIOAPI_CONFIRM_CANCEL;
            break;
        }
        else if( ch == 0x0A )
        {
            TESTIOAPIPutChar( ioapi, defch );
            ret_val = def;
            break;
        }
      }
  }
  *select = ret_val;
  TESTIOAPIPutChar( ioapi, '\n' );
```

```
  return TESTIOAPI_QUERY_OK;
}
//-----------------------------------------------------------------------------
static const char* enterStr = "Please enter a string";
int TESTIOAPIQueryString( TESTIOAPI *ioapi, const char *query, char *buffer, uint32_t
*buffersize, uint32_t maxbuffersize )
{
  char ch = 0;
  uint32_t i = 0;

  TESTIOAPIPrintf( ioapi, "%s: ", (query == 0) ? enterStr : query );

  do {
      ch = TESTIOAPIGetChar(ioapi);
      if( ch != 0 ) {
        if( ch == 27 ) // ESC Character
            return TESTIOAPI_QUERY_CANCEL;
        TESTIOAPIPutChar( ioapi, ch ); // Echo back

        // Append character if not enter, also append null character automatically
        if( ch != 0x0A ) {
            buffer[i] = ch;
            i++;
            buffer[i] = '\0';
            if( (i + 1) == maxbuffersize ) // If we run out of buffer space, return er-
ror
              return TESTIOAPI_QUERY_ERROR;
        }
      }
  } while( ch != 0x0A ); // While not enter

  buffer[i] = '\0'; // Just in case only enter was pressed
  *buffersize = i;
  return TESTIOAPI_QUERY_OK; // Success
}
//-----------------------------------------------------------------------------
#if TESTIOAPI_HAS_TESTPRINTF
void TESTIOAPITestPrintf( TESTIOAPI *ioapi ) {
  TESTIOAPIPrintf( ioapi, "Int: %%d    '%d' == '100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Int: %%+d   '%+d' == '+100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Int: %%05d  '%05d' == '00100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Int: %%5d   '%5d' == '  100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Int: %%+5d  '%+5d' == ' +100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Int: %%+05d '%+05d' == '+0100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Int: %%d    '%d' == '-100'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Int: %%+d   '%+d' == '-100'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Int: %%05d  '%05d' == '-0100'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Int: %%5d   '%5d' == ' -100'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Int: %%+5d  '%+5d' == ' -100'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Int: %%+05d '%+05d' == '-0100'\n", -100 );
  TESTIOAPIPutChar( ioapi, '\n' );

  TESTIOAPIPrintf( ioapi, "Hex: %%x    '%x' == '64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%+x   '%+x' == '64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%05x  '%05x' == '00064'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%5x   '%5x' == '   64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%+5x  '%+5x' == '   64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%+05x '%+05x' == '00064'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%x    '%x' == 'ffffff9c'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%+x   '%+x' == 'ffffff9c'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%05x  '%05x' == 'ffffff9c'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%5x   '%5x' == 'ffffff9c'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%+5x  '%+5x' == 'ffffff9c'\n", -100 );
  TESTIOAPIPrintf( ioapi, "Hex: %%+05x '%+05x' == 'ffffff9c'\n", -100 );
  TESTIOAPIPutChar( ioapi, '\n' );

  TESTIOAPIPrintf( ioapi, "HeX: %%X    '%X' == '64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%+X   '%+X' == '64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%05X  '%05X' == '00064'\n", 100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%5X   '%5X' == '   64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%+5X  '%+5X' == '   64'\n", 100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%+05X '%+05X' == '00064'\n", 100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%X    '%X' == 'FFFFFF9C'\n", -100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%+X   '%+X' == 'FFFFFF9C'\n", -100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%05X  '%05X' == 'FFFFFF9C'\n", -100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%5X   '%5X' == 'FFFFFF9C'\n", -100 );
  TESTIOAPIPrintf( ioapi, "HeX: %%+5X  '%+5X' == 'FFFFFF9C'\n", -100 );
```

```
  TESTIOAPIPrintf( ioapi, "HeX: %%+05X '%+05X' == 'FFFFFF9C'\n", -100 );
  TESTIOAPIPutChar( ioapi, '\n' );

  TESTIOAPIPrintf( ioapi, "Bin: %%b    '%b' == '1100100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%+b   '%+b' == '1100100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%09b  '%09b' == '001100100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%9b   '%9b' == '  1100100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%+9b  '%+9b' == '  1100100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%+09b '%+09b' == '001100100'\n", 100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%b    '%b' == '11111111111111111111111110011100'\n",
-100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%+b   '%+b' == '11111111111111111111111110011100'\n",
-100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%09b  '%09b' == '11111111111111111111111110011100'\n",
-100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%9b   '%9b' == '11111111111111111111111110011100'\n",
-100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%+9b  '%+9b' == '11111111111111111111111110011100'\n",
-100 );
  TESTIOAPIPrintf( ioapi, "Bin: %%+09b '%+09b' == '11111111111111111111111110011100'\n",
-100 );

  TESTIOAPIPutChar( ioapi, '\n' );
  TESTIOAPIPrintf( ioapi, "Char: %%c '%c' == 'a'\n", 'a' );
  TESTIOAPIPrintf( ioapi, "String: %%s '%s' == 'Teststring'\n", "Teststring" );
}
#endif // TESTIOAPI_HAS_TESTPRINTF
//-----------------------------------------------------------------------------
//
// testioapi.h
// Test IO API and command parser
//
//-----------------------------------------------------------------------------
#ifndef TESTIOAPI_H
#define TESTIOAPI_H
//-----------------------------------------------------------------------------
#include <stdint.h>
#include "sys/alt_stdio.h"
//-----------------------------------------------------------------------------
// Query function return values
#define TESTIOAPI_QUERY_ERROR  0x00
#define TESTIOAPI_QUERY_OK     0x01
#define TESTIOAPI_QUERY_CANCEL 0x02
// Possible choices for confirmation query
#define TESTIOAPI_CONFIRMCHOICES_YESNO       0x10
#define TESTIOAPI_CONFIRMCHOICES_YESNOCANCEL 0x11
// Possible return values from confirmation query
#define TESTIOAPI_CONFIRM_YES    0x20
#define TESTIOAPI_CONFIRM_NO     0x21
#define TESTIOAPI_CONFIRM_CANCEL 0x22
//-----------------------------------------------------------------------------
// Declare struct
struct TESTIOAPI;
typedef struct TESTIOAPI TESTIOAPI;
typedef TESTIOAPI* TESTIOAPIPTR;
//-----------------------------------------------------------------------------
// Function pointers used by PutChar and GetChar
typedef int  (*TESTIOAPIISCHARREADYCALLBACK)(void); // Returns non-zero if new data is
readable.
typedef char (*TESTIOAPIGETCHARCALLBACK)(void);     // Returns next letter from receive
buffer (0 if new data was not available)
typedef int  (*TESTIOAPIPUTCHARCALLBACK)(char ch);  // Returns zero if buffer is full,
non-zero if data was sent.
//-----------------------------------------------------------------------------
void TESTIOAPIInit( TESTIOAPI *ioapi,

      TESTIOAPIISCHARREADYCALLBACK chrready,

      TESTIOAPIGETCHARCALLBACK getchr,

      TESTIOAPIPUTCHARCALLBACK putchr );
//-----------------------------------------------------------------------------
// Print functions
// Return zero if buffer is full, non-zero if data was sent?
void TESTIOAPIPrintStr( TESTIOAPI *ioapi, const char *str );
void TESTIOAPIPrintInt( TESTIOAPI *ioapi, int32_t num );
void TESTIOAPIPrintHex( TESTIOAPI *ioapi, int32_t num );
```

```
void TESTIOAPIPrintHexSmall( TESTIOAPI *ioapi, int32_t num );
void TESTIOAPIPrintBin( TESTIOAPI *ioapi, int32_t num );
void TESTIOAPIPrintf( TESTIOAPI *ioapi, const char *fmt, ... );
//---------------------------------------------------------------------------
// Read or write a single character
void TESTIOAPIPutChar( TESTIOAPI *ioapi, char ch );
char TESTIOAPIGetChar( TESTIOAPI *ioapi ); // Non-blocking in altera
//---------------------------------------------------------------------------
// Query functions return two values
//   Function return value is one of TESTIOAPI_QUERY_xxx
//   And actual value is returned in last parameter (integer pointer)
//   query parameter can be null, in which case default is used.
// Integer
int TESTIOAPIQueryInteger( TESTIOAPI *ioapi, const char *query, int32_t *number );
int TESTIOAPIQueryRangeInteger( TESTIOAPI *ioapi, const char *query, int32_t min_val,
int32_t max_val, int32_t *number );
int TESTIOAPIQueryIntegerDefault( TESTIOAPI *ioapi, const char *query, int32_t def,
int32_t *number );
int TESTIOAPIQueryRangeIntegerDefault( TESTIOAPI *ioapi, const char *query, int32_t
min_val, int32_t max_val, int32_t def, int32_t *number );
//---------------------------------------------------------------------------
// ifdeffed away in case needed later
#if 0
// Short
int TESTIOAPIQueryShort( TESTIOAPI *ioapi, const char *query, int16_t *number );
int TESTIOAPIQueryRangeShort( TESTIOAPI *ioapi, const char *query, int16_t min_val,
int16_t max_val, int16_t *number );
int TESTIOAPIQueryShortDefault( TESTIOAPI *ioapi, const char *query, int16_t def,
int16_t *number );
int TESTIOAPIQueryRangeShortDefault( TESTIOAPI *ioapi, const char *query, int16_t
min_val, int16_t max_val, int16_t def, int16_t *number );
//---------------------------------------------------------------------------
// Byte
int TESTIOAPIQueryByte( TESTIOAPI *ioapi, const char *query, int8_t *number );
int TESTIOAPIQueryRangeByte( TESTIOAPI *ioapi, const char *query, int8_t min_val, int8_t
max_val, int8_t *number );
int TESTIOAPIQueryByteDefault( TESTIOAPI *ioapi, const char *query, int8_t def, int8_t
*number );
int TESTIOAPIQueryRangeByteDefault( TESTIOAPI *ioapi, const char *query, int8_t min_val,
int8_t max_val, int8_t def, int8_t *number );
#endif
//---------------------------------------------------------------------------
// Confirmation query
// Third parameter must be one of TESTIOAPI_CONFIRMCHOICES_xxx
// Return value is one of TESTIOAPI_QUERY_xxx
// Actual return value is in last parameter, it is one of TESTIOAPI_CONFIRM_xxx
int TESTIOAPIQueryConfirmation( TESTIOAPI *ioapi, const char *query, int choices, int
*select );
int TESTIOAPIQueryConfirmationDefault( TESTIOAPI *ioapi, const char *query, int choices,
int def, int *select );
//---------------------------------------------------------------------------
// String query
// Maximum buffer size (pointed to by buffer) is in the fourth parameter
// Third parameter returns the number of characters read
// Return value is one of TESTIOAPI_QUERY_xxx
int TESTIOAPIQueryString( TESTIOAPI *ioapi, const char *query, char *buffer, uint32_t
*buffersize, uint32_t maxbuffersize );
//---------------------------------------------------------------------------
#if TESTIOAPI_HAS_TESTPRINTF
// Test function for printing out all possible combinations of %-flags
void TESTIOAPITestPrintf( TESTIOAPI *ioapi );
#endif // TESTIOAPI_HAS_TESTPRINTF
//---------------------------------------------------------------------------
#include "testioapi/testioapi_inline.h"
//---------------------------------------------------------------------------
#endif // TESTIOAPI_H
//---------------------------------------------------------------------------
//
// testioapi_inline.h
//
//---------------------------------------------------------------------------
#ifndef TESTIOAPI_INLINE_H
#define TESTIOAPI_INLINE_H
//---------------------------------------------------------------------------
#include "testioapi/testioapi.h"
//---------------------------------------------------------------------------
// Declare internal types
```

```
struct TESTIOAPI {
  TESTIOAPIISCHARREADYCALLBACK chrready;
  TESTIOAPIGETCHARCALLBACK getchr;
  TESTIOAPIPUTCHARCALLBACK putchr;
};
//----------------------------------------------------------------------------
#endif // TESTIOAPI_INLINE_H
//----------------------------------------------------------------------------
//
// testioapi_internal.h
//
//----------------------------------------------------------------------------
#ifndef TESTIOAPI_INTERNAL_H
#define TESTIOAPI_INTERNAL_H
//----------------------------------------------------------------------------
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

#include "testioapi/testioapi.h"
#include "sys/alt_stdio.h"
//----------------------------------------------------------------------------
// Used in all hex printing
static char tohexbig[16]   = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A',
'B', 'C', 'D', 'E', 'F' };
static char tohexsmall[16] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a',
'b', 'c', 'd', 'e', 'f' };
//----------------------------------------------------------------------------
static inline void TESTIOAPI_INTERNAL_PrintZero( TESTIOAPI *ioapi, uint8_t fillZero,
int32_t fillsize )
{
  int32_t i;
  char ch = (fillZero == 1) ? '0' : ' ';

  for( i = 1; i < fillsize; i++ )
      TESTIOAPIPutChar( ioapi, ch );
  TESTIOAPIPutChar( ioapi, '0' );
}
//----------------------------------------------------------------------------
static void TESTIOAPI_INTERNAL_PrintIntFixed( TESTIOAPI *ioapi, int32_t num, uint8_t in-
sertsign, uint8_t fillZero, int32_t fillsize )
{
  uint32_t unum = (uint32_t)num;
  int32_t digit_count = 0;
  int32_t i;
  char buffer[10]; // 4294967295

  // Negative number?
  if( num < 0 ) {
      unum = -num;
      if( fillsize > 0 )
        fillsize--;
  } else if( insertsign ) {
      if( fillsize > 0 )
        fillsize--;
  }

  // Fill buffer while there's something to fill it with
  while( unum > 0 ) {
      buffer[digit_count++] = '0' + (unum % 10);
      unum /= 10;
  }

  // Filler characters behave differently, spaces are printed before possible sign
  if( !fillZero ) { // Filler == ' '
      for( i = 0; i < (fillsize - digit_count); i++ )
        TESTIOAPIPutChar( ioapi, ' ' );
  }

  // Print sign character, if necessary
  if( num < 0 )
      TESTIOAPIPutChar( ioapi, '-' );
  else if( insertsign )
      TESTIOAPIPutChar( ioapi, '+' );

  // and zeroes are printed after possible sign
  if( fillZero ) { // Filler == '0'
```

```
      for( i = 0; i < (fillsize - digit_count); i++ )
        TESTIOAPIPutChar( ioapi, '0' );
  }

  // Print buffer
  while( digit_count > 0 )
      TESTIOAPIPutChar( ioapi, buffer[--digit_count] );
}
//---------------------------------------------------------------------------
static void TESTIOAPI_INTERNAL_PrintHexFixed( TESTIOAPI *ioapi, int32_t num, uint8_t
fillZero, int32_t fillsize, uint8_t hexSmall )
{
  uint32_t unum = (uint32_t)num;
  int32_t digit_count = 0;
  int32_t i;
  char buffer[8]; // FFFFFFFF
  char ch = (fillZero == 1) ? '0' : ' ';

  // Fill buffer, no negative hex numbers, so just regular shift
  while( unum > 0 ) {
      buffer[digit_count++] = (hexSmall == 1) ? tohexsmall[unum & 0xF] : tohexbig[unum &
0xF];
      unum >>= 4;
  }

  // Print filler character(s)?
  for( i = 0; i < (fillsize - digit_count); i++ )
      TESTIOAPIPutChar( ioapi, ch );

  // Print buffer
  while( digit_count > 0 )
      TESTIOAPIPutChar( ioapi, buffer[--digit_count] );
}
//---------------------------------------------------------------------------
static void TESTIOAPI_INTERNAL_PrintBinFixed( TESTIOAPI *ioapi, int32_t num, uint8_t
fillZero, uint8_t fillsize )
{
  uint32_t value = (uint32_t)num;
  int32_t digit_count = 32;
  uint32_t bitmask = 0x80000000;
  int32_t i;
  char ch = (fillZero == 1) ? '0' : ' ';

  // Skip all leading zero bits
  while( (value & bitmask) == 0 ) {
      bitmask >>= 1;
      digit_count--;
  }

  // Print prefix fill bytes
  for( i = 0; i < (fillsize - digit_count); i++ )
      TESTIOAPIPutChar( ioapi, ch );

  // Print each data bit
  while( bitmask > 0 ) {
      char c = (value & bitmask) ? '1' : '0';
      TESTIOAPIPutChar( ioapi, c );
      bitmask >>= 1;
  }
}
//---------------------------------------------------------------------------
static void TESTIOAPI_INTERNAL_PrintHex( TESTIOAPI *ioapi, uint8_t hexSmall, int32_t num
)
{
  // Is zero?
  if( num == 0 ) {
      TESTIOAPIPutChar( ioapi, '0' );
      return;
  }

  // Use PrintHexFixed with space as filler and zero fill size
  TESTIOAPI_INTERNAL_PrintHexFixed( ioapi, num, 0, 0, hexSmall );
}
//---------------------------------------------------------------------------
#endif // TESTIOAPI_INTERNAL_H
//---------------------------------------------------------------------------
```