TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonika instituut

MHK70LT

Kardo Aia

KÕRGEPINGETESTRI OSALINE AUTOMATISEERIMINE

Autor taotleb
tehnikateaduse magistri
akadeemilist kraadi

Tallinn
2015

AUTORIDEKLARATSIOON

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus.
Esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.
Töös kasutatud kõik teiste autorite materjalid on varustatud vastavate viidetega.

Töö valmis Maido Hiiemaa juhendamisel

“.....”.....2015 a.

Töö autor

..... allkiri

Töö vastab magistritööle esitatavatele nõuetele.

“.....”.....2015 a.

Juhendaja

..... allkiri

Lubatud kaitsmisele.

..... eriala/õppekava kaitsmiskomisjoni esimees

“.....”.....2015 a.

..... allkiri

TUT Department of Mechatronics
Chair of Mechatronics

MASTER'S THESIS SHEET OF TASK'S

Year2015semester spring

Student: Kardo Aia, 121913 (name, code)
Curricula: MAHM02/09 - Mechatronics
Speciality: Mechatronics
Supervisor: Researcher, Maido Hiiemaa (position, name)
Advisers: …….. (name, position, phone)

MASTER'S THESIS TOPIC:
(in Estonian) Kõrgpingetestri osaline automatiseerimine
(in English) Partial Automation of High Voltage Tester

Thesis tasks to be completed and the timetable:

Nr	Description of tasks	Timetable
1.	**Familiarization with the available tester, creating a partial automatiation plan for it.**	**10.02.2015**
2.	**Creating a connection beween the tester and controlunit (PC). Testing the functions of the tester and their effect in diferent situations.**	**17.02.2015**
3.	**Creating the software to control the tester.**	**14.04.2015**
4.	**Creating, constructing and testing a testing rig for one product family.**	**02.05.2015**
5.	**Creating a user manual.**	**11.05.2015**

Solved engineering and economic problems:

Software must be easy to use and allow to upload testing parameters to the tester without additional command from the testingpersonel (TP). It must allow fast continious testing of products and save the results. Testing rig must allow to manually quickly change the tested product and be safe (connector connected automatically and test can start only if coverings are closed). Reconnections must be made automatically for the rig (these were done manually before, by reconnecting wires). The rig must withstand long time and continuous use.

Additional comments and requirements: work completed for Note OÜ

Language: Estonian (eesti keel)

Application is filed not later than 12.05.2015

Deadline for submitting the theses 22.05.2015

Student Kardo Aia /signature/ ……..... date……

Supervisor Maido Hiiemaa /signature/ ……..... date……

Confidentiality requirements and other conditions of the company are formulated as a company official signed letter

SISUKORD

TÄHISED

EMF	Elektromotoorjõud (electromotive force)
CTR	Voolu ülekande faktor (current transfer ratio)
nCTR	Normaliseeritud voolu ülekande faktor (normalized current transfer ratio)
IO	Sisend/väljund (input/output)
LED	Valgusdiood (light emitting diode)
VDC	Alalisvoolu volti (võlts direct current)
VAC	Vahelduvvoolu volti (võlts active current)

EESSÕNA

Lõputöö teema tuli ettevõttelt Note OÜ Pärnu. Klient, andis osa oma tooteid all töövõtjana Notele tootmiseks ja kuna toodete puhul on tähtis isolatsioon, siis saatis ka oma kõrgepinge testri (Sefelec Premier 2804). Nii Note kui ka nende kliendi poolt oli huvi testimise kiirendamiseks, lisaks soovis klient logi testide tulemuste kohta. Seetõttu sai kokku lepitud, et automatiseerin osaliselt kõrgepingetestri töö.

Kokku sai lepitud, et testrile tuleks ehitada rakis, mis laseks osasid tooteid automaatselt testida, aga ei segaks teiste manuaalselt testimist. Toode, millele rakis ehitada sai valitud vastavalt testimistingimustele, mille tõttu paljud ei sobinud: vajavad testi peale korpusesse monteerimist või sarnaste toodete ehituses on selliseid erinevusi, mis nõuaks rakise osade vahetamist.

Vastavalt kliendi nõudele on aruandess lisatud fotodest kustutatud identifitseerivad märgid, kuid üldiselt ei olnud vastuväiteid toodete ja rakise piltide lisamisele avalikult kaitstavasse aruandesse. Vabandan piltide kehva kvaliteedi eest, kuid üritasin tähtsamad kohad mitme lähemalt salvestada.

1. SISSEJUHATUS

Töö eesmärgiks oli muuta toodete testimine kiiremaks ja salvestada tulemused logisse. Testimiseks kasutatakse Sefelec Premier 2804 testrit [1]. Esialgu toimus kogu testimine käsitsi ja kui tootel oli mitu testi, siis tuli testide vahepeal muuta sooritatavat testi. Lisaks tuli testri otsad asetada vastu toodet või rakist (kui on rakis, mis ühendatud toote pesadega), kusjuures erinevate testide puhul on kasutusel erinevad testipunktid. Kasutatavate testipunktide asukohad on kirjas testimise juhendis kuid siin tekib probleem kui testija ei tee teste õiges järjekorras vaid mugavuse järgi (näiteks teeb enne testi, mis on testri mälupeasas eespool). Sel juhul võib ta valida esimesena muu programmi kuid kogemata juhendist siiski vaadata esimese testi juhendit (eriti testpunktide asukohta). Programmi puhul saaks selle ära hoida kui enne testi anda testijale teada kuhu tuleks testri otsikud asetada. Sel juhul kaoks vajadus testimisjuhendist järge pidada ja vigade tõenäosus väheneks, sest programm valiks ise, millise testi enne teha ja kuvaks ka kohe selle jaoks vajaliku info testijale.

Eesmärgiks oli muuta testimine kiiremaks ja lihtsamaks ning salvestada tulemused logisse. See tähendab, et operaator peaks tegema vähem tegevusi. Selle saavutamiseks oli plaanis luua programm, mis juhib testrit jadaühenduse kaudu. Programm peaks seadistama testri vastavalt testitavale tootele, käivitama testri testid ja salvestama testide tulemused. Nende nõudmiste põhjal lõin programmi. Programmeerimiskeeleks valisin C#, sest see pakkus kasutajaliidese tegemiseks head tuge.
Testri programmile/rakisele seatud nõueteks olid:

- Rakise abil saab testida võimalikule suurt hulka sarnaseid tooteid.
- Programm lubab ilma rakiseta testida kõiki tooteid.
- Testija ei pea testimiseks tundma testri tööpõhimõtet. Juhendi abil saab igaüks minimaalse juhendamisega testida tooteid kasutades programmi liidest.
- Tagatud on testija ohutus:
    - Programm paneb pearõhu ohutusele (selleks tuleb arvestada natuke väiksema töökiirusega). See tähendab viiviseid, et käsud kindlalt kohale jõuaksid ja lisakontrolle, kas tester on ikka testimise lõpetanud enne muu tegevuse tegemist.
    - Indikaatorid testijale, kas testimine käib või mitte. Siia kuuluvad indikaatorid nii rakisel (eri värvi tuled) kui ka programmis (selge ja arusaadav tekst ja/või olekut näitavad pildid).
- Rakis mahub olemasoleva testimiseks mõeldud laua (mõõdud 1m x 0,5m) paele ja jääb ruumi ka muude toodete testimiseks.
- Salvestatakse testi tulemused vastavalt seerianumbrile (arvestada tuleb, et mõnel tootel puudub seerianumber, mida sisse skaneerida ning seerianumbrite formaat võib muutuda).

Kuna erinevaid tooteid on palju ja tootmine toimub partiide kaupa (mitte liinitootmisena), siis pole võimalik täielikult testimist automatiseerida. Täielikku automatiseerimist takistab ka toote iseloom (tootel on käsitsi lisatav pistik, mille takistaks liinil liikumist). Esialgu sai kokku lepitud, et teen rakise kahe toote põhjal: toode 1 (Sele 7.3 ja Sele 7.4) ja toode 2 (Sele 7.5 ja Sele 7.6). Nendel toodetel on suhteliselt sarnased testide parameetrid (testi pinge) kuid ühele tehakse vahelduvvooluga kõrgepinge test, aga teisele isolatsiooni test (alalisvooluga), sest see sisaldab elektrolüüt kondensaatoreid. Sellegipoolest ei erine testimine eriti, ka esimesele tootele võib teha isolatsiooni test ja teisele võib teha alalisvooluga kõrgepinge testi. Tooted erinevad selle poolest kuidas nende ühenduspesa pinnid on ühendatud elektriskeemiga. Seetõttu ei oleks mõistlik kasutada mõlema toote jaoks ühte pistikut. Seda küll saaks kasutada, aga see nõuaks ümbervahetamiseks lisaks neli (sel juhul sobiks rakis toodetele, mille pesa pinnide ühendused on identsed valitud kahe tootega, hilisem uute toodete lisamine oleks võimatu) kõrgepingereleed. Selleks, et ühe pistiku abil toetada ka uusi tooteid oleks vaja 30 (iga pistiku viieteistkümnest pinnist tuleb ühendada läbi relee nii maanduse kui kõrgepinge otsikuga) kõrgepingereleed (neid pole võimalik ka kuhugi paigutada, kõrgepingereleede mõõtmed on suhteliselt suured) ja teistsugust kujundust. Jõudsime kliendiga kokkuleppele, et see ei oleks

majanduslikult otstarbekas ja neile sobib kui erineva toote tüübi testimisel peab vahetama pistiku kui selle vahetus on tehtud lihtsaks.
Valitud toodete (toode 1 ja toode 2) kasutamisel tuli muuta testri otsikute asukohta rakisel. See on suureks probleemiks, sest sellisel juhul on tõsine võimalus, et testija ajab testpunktid segi ja seetõttu rakendub pinge valede punktide vahele. See võib põhjustada vigastusi tootele, sest ei ole garanteeritud, et tester suudab piisavalt kiiresti reageerida ebanormaalsele olukorrale ja testi lõpetada. Valitud toodete rakised on kehvasti disainitud, nende puhul on suur oht, et testija asetab testri otsiku valesse kohta (toode 1: *Sele 7.1* ja Sele 7.2).

Rakise suuruse otsustasin piirata 40x40 cm peale, kusjuures selle ruumi sisse peavad mahtuma ka laua peale jäävad kaablid ja õhuvoolikud. Suuruse valikul sai arvestatud, et rakise ette oleks võimalik vajadusel ka toode asetada ja kõrvale mahuks peale suurima testitava toote ka veel selle juurde käiv abiseade. Lisaks peaks testri otsad olema rakisega jäigale seotud ja nende tööpinnad (voolu kandvad osad) testija eest varjatud. See suurendab ohutust, sest isegi väga ettevaatlikul tegutsemisel võib õnnetusi juhtuda (ja vaatamata testri enda ohutus-protokollidele), aga kui otsad on kaetud nii, et neid puutuda ei saa, siis on selliste õnnetuste juhtumise tõenäosus märgatavalt väiksem.

2. TESTER

2.1. Testri ülevaade

Kasutatavaks testriks on ohutustester Sefelec Premier 2804 [1] (Sele 7.7). See võimaldab järgmisi teste:

1) Kõrgepinge test vahelduvvoolule (kuni 5 kV).
2) Kõrgepinge test alalisvoolule (kuni 6 kV).
3) Isolatsiooni test (kuni 1000 V, alalisvool).
4) Maanduse test (kuni 30 A, vahelduvvool).

Testril on mitmeid kasulikke funktsioone:

- Pinge aeglase tõstmise võimalus („ramp up time“).
- Ohutu maha laadimine („safety discharge“).
- Ületemperatuuri, -pinge ja –voolukaitse.
- Läbilöögi avastamine („flashover detection“).
- Limiidid maksimaalsele voolule (Sele 2.1).
- Testi automaatne katkestamine kui jälgitav parameeter (kõrgepinge testi puhul vool, isolatsiooni ja maanduse testil takistus) väljub seatud piiridest.
- Kiire voolu katkestamine (<150 µs) kui

Table 1: Output Limitation in Withstanding Voltage Testing

	Upper Current	Pause	Output Time
AC	30mA≤I≤40mA	At least as long as the output time	Maximum 240 seconds
	0.001mA≤I<30 mA	Not necessary	Continuous output possible
DC	0.001mA≤I≤10 mA	Not necessary	Continuous output possible
GB	15A<I≤32A	At least as long as the output time	999.9
	3A≤I≤15A	Not necessary	999.9

NOTE: Output Time = Ramp Time + Test Time.

Sele 2.1. Testri piirangu voolule. [2]

2.1.1. Testide põhimõte

Testri testide eesmärgiks on ära tunda vigast isolatsiooni või maandust. Peaaegu kõik elektrilised tooted peavad läbima kõrgepinge testi. Tootja võib ise või vastavalt toote iseloomule testida ka maandust. Vigane isolatsioon või maandus võib põhjustada kasutaja surma. Seetõttu testitakse tavakasutusest kümneid kordi kõrgemate parameetritega, et vältida hilisemaid probleeme.

Elektrilised ohutuse testid on [3] [4] [5] [6] [7] [8] [9] [10] [11]:

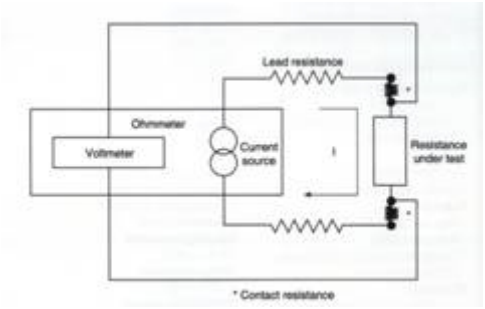
- Kõrgepinge test [12] [13] – Tuntud kui „Hipot test“ või „Dielectric Voltage Withstand test“. Mõõdab lekkevoolu (tavaliselt on faasi- ja neutraaljuhe ühendatud) kui kahe punkti vahel (tavaliselt on üks neist maandus) rakendatakse potentsiaalide erinevust (pinget). Lekkevoolu mõõdetakse nende punktide vahel, mis ei ole ühendatud. Nende punktide vahel saavad laengud liikuda vaid läbi isolatsiooni, mis tavatingimustel ei tohi voolu juhtida. Seetõttu testitakse kasutustingimustest palju kõrgemal pingel ja vaadatakse, et lekkevool ei oleks suur (tavaliselt alla 10 mA) ega tekiks ülelööki. Kasutatakse alalis- või vahelduvvoolu, vastavalt nõuetele või toote opereerimise voolu tüübile. Vahelduvvoolu puhul ei tohi koheselt rakendada testipinget, sest mahtuvuslikud elemendid põhjustavad vooluhüpet.
    - On olemas ka purustav kõrgepinge test („Dielectric Voltage Breakdown Test“), mille puhul ei mõõdeta lekkevoolu vaid rakendatakse tõusvat pinget kuni isolatsioon hävineb (voolutugevus tõuseb järsult). Kuna see on purustav test, siis kasutatakse seda vaid toote arendamisel.

  Lisaks testitava seadme kasutatavale pingele on ka muid põhjuseid, miks valida alalis- või vahelduvvool (Tabel 2.1).

- Isolatsiooni test [14] – Tuntud kui „Insulation Resistance Test“ ehk „IR Test“. Mõõdab isolatsiooni takistuslikku komponenti, seetõttu kasutatakse testimiseks alalisvoolu (vahelduvvoolu kasutamisel mõõdetaks lisaks ka mahtuvuslikku komponenti). Mõõtmiseks rakendatakse kahe mitteühendatud punkti vahele pinge ja mõõdetakse voolu. Nende järgi saab arvutada isolatsiooni takistuse, mis peab olema kõrgem nõutust.
- Maanduse test [15] – Tuntud kui „Ground Bond Test“ ehk „GB Test“ või „PE resistance test, ground continuity test“. Testib, kas testitava objekti maanduspunktid on ühendatud omavahel ja toite maandusega. Mõõdetakse takistust, mis peab olema lubatust madalam, nende punktide vahel. Väikese takistuse täpseks mõõtmiseks antakse väga stabiilset voolu kahe juhtme kaudu ja kombineeritakse pinge mõõtmine kahe juhtme kaudu (Sele 2.2).
- Lekkevoolu test [16] – Tuntud kui „Line Leakage Test“ või „Leakage Current Test“. Mõeldud inimkeha olemasolu simuleerimiseks testimisel. Mõõtmiseks kasutatakse milliampermeetrit 1750 Ω näivtakistusega ja 225 μs ajakonstandiga, mis simuleerib inimkeha lisamist. Ampermeeter ühendatakse neutraalse juhi ja korpuse vahele ning mõõdetav vool peab jääma alla inimesele ohtliku piiri.

Tabel 2.1. Alalis ja vahelduvvoolu kasutamise eelised ja puudused.

Vahelduvvool	
Eelised	**Puudused**
• Ei vaja pinge aeglast tõstmist (mahtuvuslike komponentide tõttu). • Ei vaja testitava objekti tühjaks laadimist. • Kiirendab vigase materjali hävimist. • Osadel juhtudel võib kasutada liini lekke testi asemel. • Osadel juhtudel aktsepteeritakse vaid vahelduvvoolu testi.	• Mahtuvusliku komponendiga objektide testimine nõuab rohkem voolu.
Alalisvool	
Eelised	**Puudused**
• Ainus viis objekti osade parameetrite testimiseks (näiteks kondensaatorite pingereiting või dioodide tagasipinge reiting).	• Kõrge mahtuvuslikkusega objektide testimine on aeglane (pinget tuleb tõsta väga aeglaselt).



Sele 2.2. Maanduse testi põhimõte.

2.2. Testri juhtimine

Testrile käskude saatmiseks on 3 võimalust:

1) RS232 ühendus.
2) USB (A tüüp) ühendus, testris on usb-RS232 sild.
3) GPIB ühendus (vajab lisakaarti testrile).

Parim variant on USB liides. See võimaldab kõiki RS232 võimalusi (kuna see on tänu usb-RS232 sillale põhimõtteliselt RS232 ühendu USB kaabli kaudu). Ühendumine testriga on samamoodi nagu RS232. Kahjuks on testril väike puudus kui kasutada USB liidest. Arvutis on ühendus nähtav RS232 pordina ja enne ühendamist tuleb valida andmeside kiirus („baud rate“) kuid testris saab ühenduse kiirust valida vaid RS232 ühenduse režiimis. Sel juhul tuleb tester muuta RS232 režiimi, muuta ühenduse kiirus ja seejärel muuta USB režiimi.

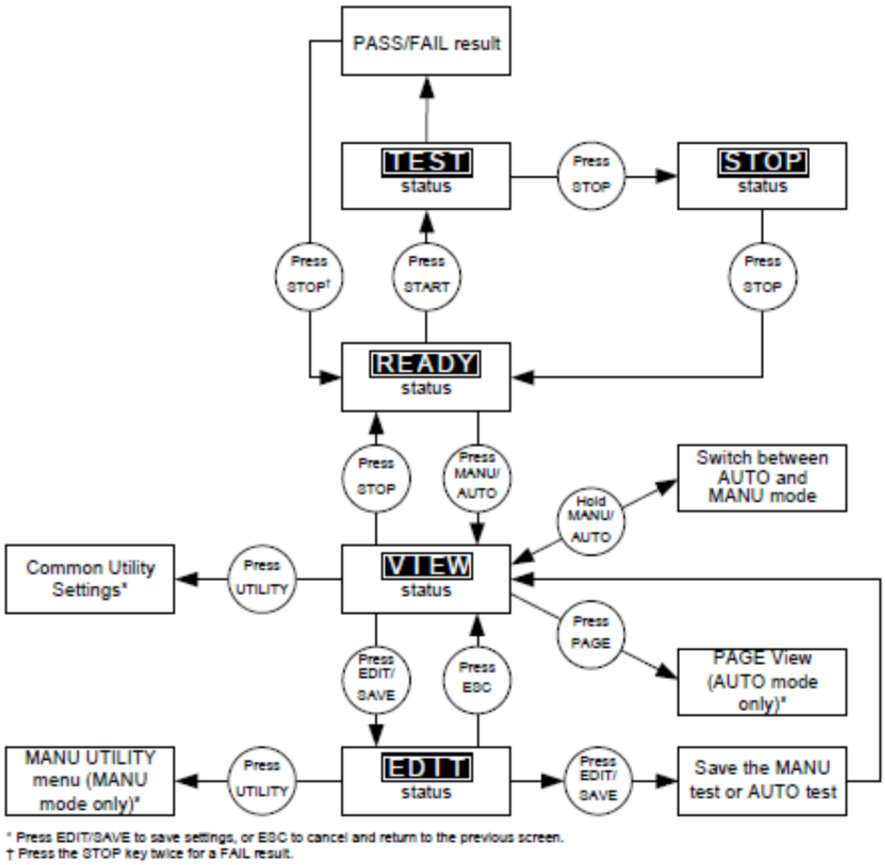
2.2.1. Testri menüü ja olekud

Testril on kaks testimise režiimi [17]:

1. Manu – testimine kasutades vaid ühte manuaalset testi. Selles režiimis saab ka muuta teste.
2. Auto – testimine kasutades automaatset testi, mis kasutab valitud manuaalseid teste.

Viis olekut (Sele 2.3):

1. View – saab vaadata valitud testi parameetreid. Saab muuta testri režiimi (manu või auto).
2. Edit – saab muuta valitud testi parameetreid.
3. Ready – tester on valmis testimiseks. START nupu vajutamine alustab testi.
4. Test – testeri test on aktiivne. STOP nupu vajutamine peatab testi. Oluline on, et testija ei puutuks sel ajal rakist ja testri osasid ohtlikest kohtadest. Seetõttu peab programm selgelt näitama kui test on aktiivne ja kontrollima kas test on ikka lõppenud.
5. Stop – olek peale testi peatamist.



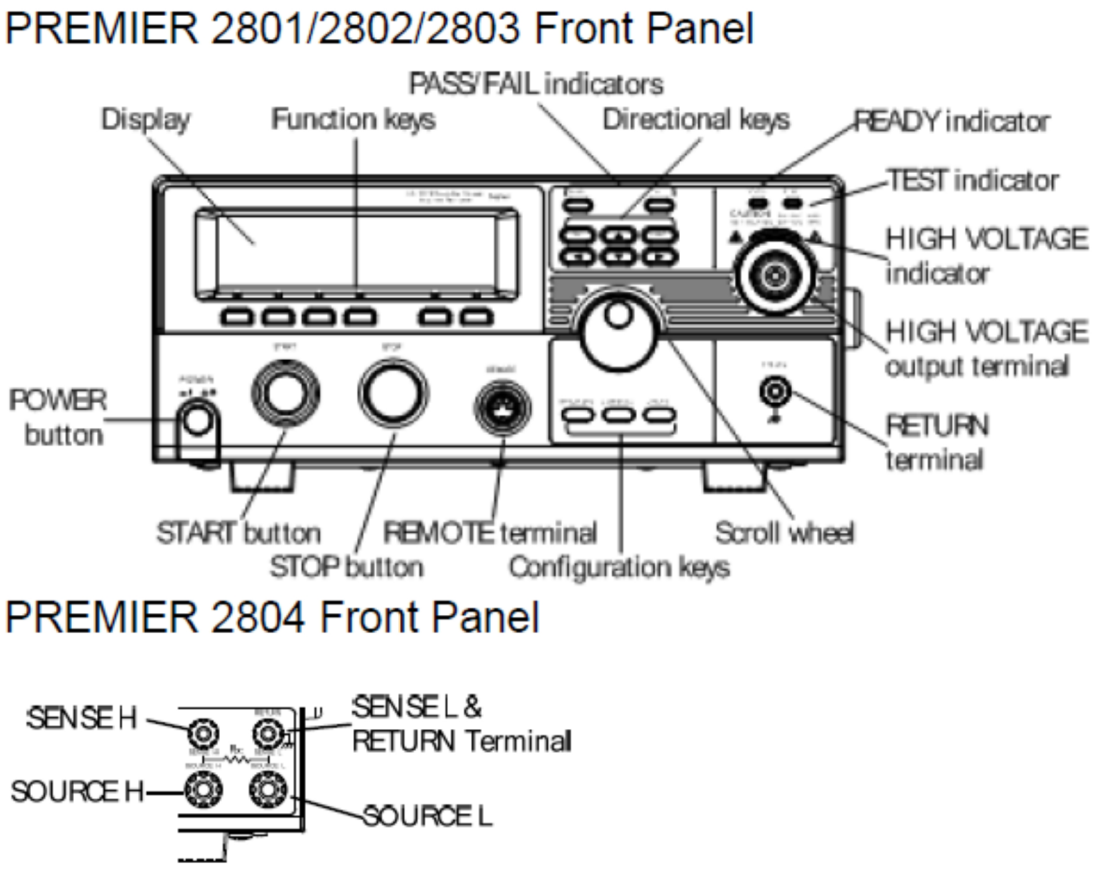
Sele 2.3. Testri olekud. [17] *lk 24*

Tester võimaldab ka testri otsade nullimist („Zerocheck“). See on võimalik vaid GB („Ground bond“) testidele. Zerocheck on lisatud testri juhtimise programmile kuid mitte teisele programmile, mis mõeldud rakise juhtimiseks (sest rakis on mõeldud kõrgepinge ja isolatsiooni testide jaoks).

2.2.2. Testri sisendid ja väljundid

Testri esiküljel (Sele 2.4) on:

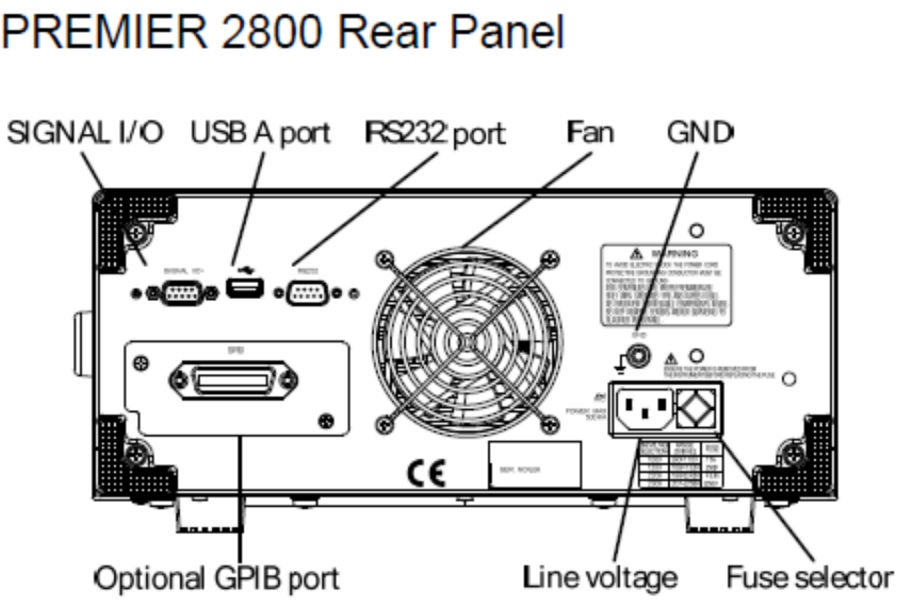
- sisse lülitamise nupp (’POWER button’).
- start nupp (’START button’).
- stop nupp (’STOP button’).
- Tuled testi tulemuse näitamiseks (‘PASS/FAIL indicators’).
- Testimiseks valimis olekut näitav tuli (‘READY indicator’). Näitab, et start nupu vajutamine alustab testi.
- Testi käimist näitav tuli (‘TEST indicator’).
- Tuli, mis näitab, et testri otsikute vahele on rakendatud kõrgepinge (‘HIGH VOLTAGE indicator’). Vilgub testi ajal.
- Keeratav nupp (‘Scroll wheel’) andmete sisestamiseks.
- Noolenupud (‘Directional keys’) menüüdes liikumiseks. ’ESC’ nupp liigub menüüst välja või katkestab parameetri seadmise. ’PAGE’ nupp on automaattesti informatsiooni ja testitulemuste vaatamiseks.
- Konfiguratsiooni nupud (‘Configuration keys’) testri muude funktsioonide kasutamiseks.
- Funktsiooni nupud (’Function keys’). Töötavad vastavalt ekraanil näidatud sildile.
- Pesa (‘Remote terminal’) kaugjuhtimise puldi jaoks.
- Testri maanduse ja mõõtmise terminalid (‘RETURN terminal’). Sele 2.4 alumises pooles on toodud tegelik asetus lähemalt. ’RETURN’ terminal on voolu tagasijooksuks kõikide testide jaoks, ’SOURCE’ ja ’SENSE’ terminalid on ’GB’ testide jaoks.
- Kõrgepinge väljundterminal (‘HIGH VOLTAGE output terminal’).



Sele 2.4. Testri esikülg. [17] *lk12*

Testri tagaküljel (Sele 2.5) on:

- Port signaalide saatmiseks (’SIGNAL I/O port’). Testri olekute (’pass’, ’fail’, ’test’) jälgimiseks ja testrile käskude (’start’, ’stop’) saatmiseks.
- RS232 port (’RS232 interface port’) testri juhtimiseks ja püsivara uuendamiseks.
- USB A port testri juhtimiseks.
- Ventilaator (’Fan’) jahutamiseks.
- Maanduse ühendus (’GND’) maanduse ühendamiseks maaga.
- Liinipinge sisend (’Line voltage input’), millele sobivad 100/120/220/230 VAC (±10%).
- Liinipinge kaitse (’Line voltage fuse’).
- GPIB port (valikuline) (’Optional GPIB port’) testri juhtimiseks.



Sele 2.5. Testri tagakülg. [17] *lk15*

2.2.3. Testri juhtimise käsud

Kõik käsud on loetletud testri manuaalis [17] lk 90 kuni 118.

3. PROGRAMM TESTRI JUHTIMISEKS

3.1. Üldine tööalgoritm

3.1.1. Nõuded programmile

Programmile on seatud nõuded:

- Tagab ohutuse:
    - Käskude saatmise vahele lisatud viivised tagavad, et käsud ka täidetakse.
    - Tähtsamate käskude puhul järgneb käsule testri oleku kontrolli käsk.
    - Otseselt mitte vajalikud testri oleku kontrollimised, mille eesmärk on tuvastada, kas mingil põhjusel test ikka käib.
- Kogub testide tulemused logisse, koostöös NOTE'ga mõtlesin juurde lisanõuded, mis teevad testija töö lihtsamaks:
    - Kui kasutatakse manuaalset seerianumbri sisestamist (juhtudel kui tootel ei ole skaneeritavat seerianumbrit), siis peab programm numbrit automaatselt suurendama. Tuleb arvestada, et seerianumber võib sisaldada ka tähti ja sel juhul tuleks suurendada viimast numbrit.
    - Seerianumbrit saab peale sisestamist ka käsitsi muuta.
- Indikaatorid testi olekust, programm näitab, millised ülesanded on tehtud, millised testid tuleb teha, millised neist on juba tehtud ja mis oli tulemus. Lisaks näitab, kas testri test on aktiivne:
    - Kiri ja visuaalne indikaator, mis näitab testri olekut.
    - Visuaalne indikaator, mis näitab, kas test on aktiivne.
- Võimalike veaolukordade puhul näitab testijale olukorra lahendamiseks juhist.
- Testi saab peatada lisaks testri nupule ka programmi abil.
- Muud lisafunktsioonid mugavuse jaoks:
    - Lisamoodul, et kustutada ajutised failid (programmi parameetrite korrigeeritud failid, mida programmeerimisel kasutatakse ja mis luuakse igal programmeerimisel uuesti) ja 'debug' failid.

- Võimalus salvestada programmi oleks, et saaks testrile programmeerida uus programm ja seejärel kiirelt jätkata vana programmi kasutamist. Sel juhul tuleks uus programm kirjutada testri uutesse mälukohtadesse, aga jätta meelde eelmise programmi asukohad mälus, et sellega jätkamisel ei peaks testrit uuesti programmeerima. See funktsioon on vajalik olukorras kui mingi toote suure hulga testimise ajal tekib vajadus kiirelt testida väike hulk muud toodet.

3.1.2. Programmi tööalgoritm

Algoritmid programmidele, mis on mõeldud töötamiseks rakisega (Sele 7.26) või ilma (Sele 7.25) on natuke erinevad. Peamine erinevus on tegutsemine testimise ajal. Ilma rakiseta testimiseks mõeldud programm vajan testija tagasisidet kui juhtmed on järgmise testi jaoks ümber seatud. Rakisega testimiseks mõeldud programm lülitab ise vajalikud releed ümber ja liigutab ka silindrit.

3.2. Programmeerimiskeel

Programmeerimiskeelena kasutasin C#. Valikuteks olid C, C++, Python (eelnevad kolm on heal tasemel selged), C#, Java. C# eelised teiste keelte ees olid:

- Paremad graafilise väljundi lisamise funktsioonid kui C++ (ja C).
- Suurem valik teeke kui C++ (ja C).
- Sisse ehitatud mälu haldamine, mida tuleks C ja C++ puhul ise teha.
- Lihtsam õppida kui Java (minu eelnevate teadmiste baasilt).
- Rohkem võimalusi kui Java ja Python.
- C# puhul on olemas head teegid jadapordi kaudu suhtlemiseks.

Kuid sellel on ka puuduseid:

- Kood on vähemefektiivne kui C++ ja C (kuid minu programm ei vaja nii suurt andmete töötlemist, et see mõjutaks).

Põhilised faktorid C# kasuks olid lihtne õppimine C++ baasil, head teegid jadapordi jaoks ja lihtne kasutajaliidese loomine.

3.3. Programmi tähtsamad lõigud

3.3.1. Kirjutamine väljundisse

Programmi koodi lihtsama kirjutamise ja hiljem muutmise huvides oli vaja lihtsat meetodit, mille abil oleks võimalik valitud tekst kirjutada valitud väljundisse. Väljundeid on kokku kolm: ’Console Output’ (peamised teated programmi olekust), ’Error Output’ (vigade teated) ja ’Debug’ (muud teated). Funktsiooni eesmärk oli valitud tekst kirjutada ’debug’ väljundisse ja vastavalt vajadusele ka ülejäänutest mõlemasse või ainult ühte neist. Funktsiooni kutsumiseks on klass WriteConsole (L7.1 Programmi fail 15: WriteConsole.cs, lk 166). Klassi meetodid on staatilised, et neid saaks kutsuda ilma klassi loomata. Käsk Wri.ConWrite(string text, int select = 0) käivitan ConWrite meetodi, mis kutsub meetodi, milles käivitatakse soov, mida jälgitakse klassis RequestListener (L7.1 Programmi fail 12: RequestListener.cs, lk 158). Samuti kirjutatakse sama tekst ’debug’ väljundisse. Käivitatakse meetod ProcessRequest (Programmi kood 3.1), milles käivitatakse meetod MainWin klassist (MainWin klass saadeti WriteConsole klassi Wri_m meetodis, mis käivitati MainWin klassis). Käivitatud meetod (’LogToOutput’ ja/või ’LogToError’) valitakse vastavalt teisele parameetrile ja meetodile saadetakse sisendiks esimene parameeter (kood: L7.1 Programmi fail 2: MainWin.cs, lk 91). Meetodid erinevad selle poolest, millisesse nimekirja nad soovitud teksti lisavad.

Programmi kood 3.1. Meetod ProcessRequest (L7.1 Programmi fail 12: RequestListener.cs):

```
public void ProcessRequest(object o, RequestArgs e)
        {
            switch (e.Select)//redirect to selected output
            {
                case 1:
                    _main.LogToOutput(e.Text);//execute Mainwin class method
                    break;
                case 2:
                    _main.LogToError(e.Text);//execute Mainwin class method
                    break;
                case 3:
                    _main.LogToOutput(e.Text);
                    _main.LogToError(e.Text);
                    break;
            }
        }
```

3.3.2. Kirjutamine ’debug’ väljundisse

’Debug’ väljundisse kirjutamine on saavutatud konsooli väljundi ümbersuunamisega. See on teostatud MainWin klassis (L7.1 Programmi fail 2: MainWin.cs, lk 91), kus väljund suunatakse ListBoxWriter klassi (L7.1 Programmi fail 5: ListBoxWriter.cs, lk 147). Ümbersuunamine saavutatakse nagu näitab Programmi kood 3.2 (käsud ei ole programmi koodis järjestikku).

Programmi kood 3.2. Konsooli ümber suunamine (L7.1 Programmi fail 2: MainWin.cs, lk 91):

```
private TextWriter _writer; //deklareerida klassi muutuja
_writer = new ListBoxWriter(DebugWin); //luua klassi ’instance’ andes edasi
listbox kuhu teks ümber suunatakse. ListBoxWriter on vasavalt üles seatud, et ta
parameetri vastu võtaks.
Console.SetOut(_writer); //suunab konsooli väljundi ümber ’_writer’ peale. Kõik
’Write’ ja ’WriteLine’ käsud suunatakse ’_writer’ peale.
```

ListBoxWriter klass on välja toodud lisades (L7.1 Programmi fail 5: ListBoxWriter.cs, 147). Selline ümbersuunamine võimaldab lihtsa käsuga (’Console.WriteLine(string);’) saata teksti (’string’) valitud väljundisse.

3.3.3. Programmi salvestamine

Lisasin ka võimaluse programm salvestada. See võimaldab lukustada testri mälupesad, mida kasutati valitud programmi jaoks ja salvestada järgmine programm teistesse mälupesadesse, et antud programmi saaks hiljem edasi kasutada. See töötab põhimõttel, et kui salvestada testri programm, siis järgmine programm salvestatakse teistesse mälupesadesse (programmi sulgemisel vana testri programm kaob). Meetod on teostatud peamiselt klassis LoadProgram (L7.1 Programmi fail 6: LoadProgram.cs, lk 149) ja MainWin klassi (L7.1 Programmi fail 2: MainWin.cs, 91) meetodis ’button_saveset_Click’ (Programmi kood 3.3) (kood teostatakse nupu ’Save Settings’ vajutamisel).

Programmi kood 3.3. Meetod button_saveset_Click (L7.1 Programmi fail 2: MainWin.cs, 91). Osa koodi eemaldatud:

```
private void button_saveset_Click(object sender, EventArgs e)
        {//user interaction removed
                return;}//exit
            _currentTest.StartedTest = _startedtest.ToArray();//save test
parameters
            _autoTestNoMin--;//Set program indexes to free places for next program
            _manualTestNoMin = _manualTestNo--;//Set program indexes to free
places for next program
            _listOfSavedTests.Add(_currentTest);//add to tests list //user
interaction removed
            string parameters = null;
            foreach (TestList param in _currentTest.TestParams)
            {//show all parameters
                parameters += "\n";
                parameters += param.TestParameters;
                Wri.ConWrite("\t" + Resources.Name + ": " + param.TestName + " ("
+ param.TestNo + "), " + Resources._test + ":" + param.TestParameters, 1);
            }//user interaction removed
            mb.ShowDialog();
        }
```

Esiteks salvestatakse aktiivse testri programmi parameetrid, mida kasutatakse testi ajal ja seejärel muudetakse indeks, mis määrab mitmendast testri mälupesast alustada uute testide salvestamisel nii, et vana testi üle ei salvestataks. Testi salvestamine võimaldab vajadusel kiiremalt vahetada erinevate testri programmide vahel. See on vajalik näiteks juhtudel kui toote

testimise ajal on vaja vahepeal testida mõned eksemplarid teist toodet (siin täiesti tavapärane juhtum). Sel juhul ei ole vaja iga kord testrit uuesti programmeerida vaid saab mälust laadida vajamineva toote programmi.

Salvestatud programmi laadimiseks tekib nupu ’Load Settings’ vajutamisel klassi ’LoadProgram’ aken, mis lubab valida salvestatud programmide vahel. Programmi laadimisel salvestatakse aktiivse programmi struktuur (’struct’) ’_currentTest’ üle laetava programmi omaga.
Struktuur (’_currentTest’) sisaldab järgmisi andmeid:

- (string) AutoName – testri programmi nimi (automaatse testi nimi).
- (int) AutoTestN – automaatse testi mälupesa number testris.
- (string[]) StartedTest – andmed, mida on vaja hiljem logisse kirjutada.
- (bool) TestIsProgrammed – näitab, kas testi testrisse programmeerimine lõpetati edukalt (muidu võib juhtuda, et testri programmeerimine katkes kuid struktuuris on juba salvestatud andmeid).
- (TestList[]) TestParams – andmed iga manuaalse testi kohta.
    - (string) Relays – releede asendid, kasutatakse vaid programmi versioonis, mis kasutamiseks koos rakisega.
    - (string) TestDescription – testi kirjeldus testi parameetrite failist. Mõeldud testijale enne testi näitamiseks, sisaldab juhiseid, mida teha enne testi algust.
    - (string) TestName – manuaalse testi nimi.
    - (int) TestNo – mälupesa number testris kuhu test salvestada.
    - (string) TestParameters – testi parameetrid, võetud testi parameetrite failist testi testrisse programmeerimise ajal.
    - (int) TestingTime – testi parameetrite järgi testimiseks kuluv aeg. Mõeldud taimerite jaoks, et õigel ajal kontrollida, kas test on pärast testi aja möödumist lõppenud.
- Meetod ’ public override string ToString() { return AutoName; }’, mis käsu ToString peale annab tagasi muutuja ’AutoName’.

Salvestatud programmi andmetes on kõik vajalikud parameetrid, et sellega toodet testida: muutujad programmi enda kasutamiseks ja testri mälupesade numbrid, kuhu testid on salvestatud.

3.3.4. Taustal toimuvad toimingud

Testri programmeerimine (meetod ’worker_program_DoWork’) ja testimine (meetod ’worker_test_DoWork’) toimuvad kasutades taustal töötamist (’background worker’) (L7.1 Programmi fail 2: MainWin.cs). Nende pikkade toimingute taustal tegemine võimaldab samal ajal kasutajaliidest kasutada ja ka neid toiminguid katkestada. Kui teha pikki toiminguid samas ’threadis’ kui kasutajaliides, siis ei saaks toimingut katkestada, sest samal ajal kasutajaliidese nuppude vajutamine ei saa teha toiminguid kuni kestev tehing on lõppenud.

3.3.5. Kasutajaliidese muutmine taustaprotsessist

Selleks, et taustal toimuvate protsesside ajal (testri programmeerimine ja testimine) oleks võimalik muuta kasutajaliidest on vaja need kutsed kasutajaliidese muutmiseks viia põhiprotsessile (’thread’). Selleks on ’InvokeC’ klass (L7.1 Programmi fail 4: InvokeC.cs, lk 147), mille kaudu saab neid muuta.

3.3.6. Kasutatava keele muutmine

Programmi kirjutamise lõppfaasis otsustasin lisada ka võimaluse muuta programmi keelt. Selleks asendasin kuvatava teksti avaldamisel tavalise konstantse stringi (’constant string’) muutujaga ’Resources.Name’. Asendused tegin programmi ’ ReSharper’ [18] abil. Tekst (’Value’) on salvestatud koos muutujaga ’Name’ faili ’Resources.resx’ (Sele 7.30) (Sele 7.32). Keele vahetamiseks tuleks tõlkida ära see fail ja kasutada, siis faili varianti, mis soovitud keeles. Täielikuks keele muutmiseks tuleks ära tõlkida ka kõigi klasside juurde kuuluvad ’.resx’ failid (Sele 7.31). Lihtsamalt saab keele vahetust kui erinevate keelte ’.resx’ failid nimetada kujul ’klassi_nimi.keele_kood.resx’ (näiteks Briti inglise keele puhul oleks keele_kood en-GB) ning valida keel programmi seadetest.

4. TESTIMISRAKISE EHITUS

4.1. Üldine ehitus ja nõuded

Kogu elektroonika on paigaldatud isoleerivasse karpi, et kaitsta testijat elektrilöögi eest. Karbis on ka ventiil kuna karbi suurus on piisav ja nii on see kaitstud tolmu eest. Ventiili poolt läbi summuti välja lastava õhu eemaldamiseks on ventiil paigaldatud summutitega karbi seina lähedale ja seina puuritud väikesed augud. Ventiil on eraldi konstruktsioonil, kuhu läheb ka testitav toode.

Et paremini isoleerida kõrgepinge ja juhtiv madala pingega osa on nad erinevatel trükkplaatidel (juhtimisosa plaat ja releede plaat). Erinevatele plaatidele paigutamise vajaduse tingis ka kõrgepinge releede suurus.

Rakise ehitusel on arvestatud, et selle saaks kergelt ühendada Sefelec testeriga. Sel eesmärgil ei ole kasutatud eraldi juhet kõrgepinge jaoks, vaid ühendati testri otsik karbiga.

Rakise mudeli illustratsioonid on välja toodud lk246 ja lk247.

4.2. Silinder ja ventiil

4.2.1. Pistiku liigutamiseks vajalik jõud

Silindri valikul tuleb arvestada, et ventiil suudaks kindlalt pistikut sisse ja välja liigutada 6 bar (0,6 *MPa*, ettevõtte suruõhu rõhk) rõhu juures.

Silindri vajaliku jõu saab arutada valemiga (4.1), kus *m* on kaal, mis on vajalik pistiku liigutamiseks.

silindri vajaliku liigutamiseks.

$$F = m * g \qquad (4.1)$$

, kus *F* – jõud silindri liigutamiseks,

m – silindri liigutamíseks vajalik mass,

g – raskuskiirendus, $g = 9{,}8\ \frac{m}{s^2}$.

Katsetasin pistiku pesasse lükkamiseks ja pesast eemaldamiseks vajaminevat jõudu rippkaaluga (kalibreeritud). Tehtud katsete põhjal on pistiku sisse lükkamiseks vaja 3,5 kuni 4,8 kg (Tabel 4.1) suurust raskust ehk suurim jõud, mida võib vaja minna on $4{,}8 * 9{,}8 = 47{,}1$ N ja keskmine jõud on $4{,}45 * 9{,}8 = 43{,}7$ N (kasutatud valemit (4.1)). Pistiku välja tõmbamiseks on vaja 3,2 kuni 4,1 *kg* (Tabel 4.1) vastavat jõudu ehk suurim jõud, mida vaja võib minna on $4{,}1 * 9{,}8 = 40{,}2$ N ja keskmine on $3{,}7 * 9{,}8 = 36{,}2$ N (kasutatud valemit (4.1)). Silindri liikumisulatus peab olema vähemalt 2 cm.

Tabel 4.1. Pistiku liigutamiseks vajalikud jõud.

Test	Jõuga võrdsustatud kaal pistiku sisse lükkamiseks m/kg	Jõuga võrdsustatud kaal pistiku välja tõmbamiseks m/*kg*
1	4,1	3,8
2	3,9	3,3
3	4,8	4,1
4	4,7	4,1
5	4,5	4,0
6	4,6	3,8
7	4,2	3,2
8	3,5	3,3
9	4,5	3,6
10	4,4	3,7
Keskmine	4,45	3,7

4.2.2. Silindri valik

Lisaks on veel mõned nõuded, millele silinder peab vastama [19]:

- Võimalikult kompaktne disain. Kuna töötamise jõud on väikesed ja külgsuunalised jõud on väikesed, siis pole vaja täissuuruses silindrit. Kompaktsus aitab ka kokku hoida ruumi testimiskohas.
- Juhtimisega, et pistik ei pöörleks ümber oma telje.
- Võimalusel otsalülitid.

Ettevõttes on olemas SMC MGQM 12-40 silinder, mis peaks sobima (Sele 7.13). Silinder on kompaktne ja kasutab kahte kolbi, seega ei ole probleemiks pistiku nihkumine. Silindrile saab küll lisada otsa andurid, kuid need on kallid (30 eurot tükk) ning on võimalik saada hakkama ka ilma. Silindri jõu saab arvutada valemi (4.2) ja (4.3) alusel (esimese valemiga silindri välja liikumise ehk pistiku pesasse lükkamise jõu ja teise valemiga silindri sisse liikumise ehk pistiku pesast välja tõmbamise jõu).

$$F_{out} = \pi * \frac{D_b^2}{4} * p^{air} \quad (4.2)$$

, kus F_{out} – silindri jõud välja liikumisel [N],

D_b – silindri kolvi läbimõõt [mm], D_B = 12 m [20].

ρ_{air} – suruõhu rõhk [MPa], ettevõttes NOTE Pärnu on ρ_{air} = 0,6 MPa.

$$F_{out} = \pi * \frac{12^2}{4} * 0{,}6 = 67{,}9\ N$$

$$F_{in} = \pi * \frac{D_b^2 - D_r^2}{4} * p^{air} \quad (4.3)$$

, kus F_{in} – silindri jõud sisse liikumisel [F],

D_r – silindri varda läbimõõt [mm], D_r = 6 mm [20].

$$F_{in} = \pi * \frac{12^2 - 6^2}{4} * 0{,}6 = 50{,}9\ N$$

Silindri jõud väljaliikumisel on 67,9 N ja sisse liikumisel 50,9 N. Varuteguri saab arvutada valemi (4.4) abil.

$$k = \frac{F_{cyl}}{F_{need}} \quad (4.4)$$

, kus k – varutegur [],

F_{cyl} – silindri jõud [N],

F_{need} – vajaminev jõud [N].

Silindri väljaliikumisel oleks varutegur arvestades maksimaalset silindri (F_{cyl} = 67,9 N) ja vajaminevat jõudu (F_{need} – 47,1 N):

$$k = \frac{67{,}9}{47{,}1} = 1{,}4$$

Silindri sisse liikumisel oleks varutegur arvestades maksimaalset silindri (F_{cyl} = 50,9 N) ja vajaminevat jõudu (F_{need} – 40,2 N):

$$k = \frac{50,9}{40,2} = 1,3$$

Antud silindril tundub olevat piisav varu, eriti arvestades, et see on saadud võrreldes katseliselt saadud maksimaalse jõuga. Reaalselt võib muidugi olla pistikuid, mis vajavad suuremat jõudu, kuid need ei ole ilmselt nii suured, et valmistada probleeme. Samuti ei ole otstarbekas kasutada suurema jõuga silindrit, sest niimoodi võivad plaat või pistik viga saada.

4.2.3. Silindri tagasiside

Silindri asendi kontrollimiseks saab kasutada otsalüliteid, mille saaks ühendada kontrolleri IO pinnidega. Sel juhul saaks automaatselt kontrollida, kas silinder on liikunud nõutud asendisse. Otsalülitite hind antud silindrile oleks olnud 30 eurot tüki kohta, vaja oleks läinud kahte. Minu juhendaja ettevõttest oli nõus, et see ei tasu ennast reaalselt ära, sest lisakontroll, mille testija peab tegema oleks minimaalne. Lisaks ei kahjusta silindri mittetäielik liikumine mingil viisil toodet. Seetõttu sai programmi lisatud testija sisendit nõudev paus, kus testija peab veenduma, et silinder on liikunud täielikult ja sisestama selle programmi.

4.2.4. Ventiil

Silindrit varustatakse õhuga parker p2l-av511es [21] ventiili (Sele 7.14) kaudu, mis on ettevõttes olemas ja mille näitajad on:

- Maksimaalne rõhk 10 MPA.
- Väljundite/olekute arv 5/2. Ventiil jagab õhku kahele väljundile. On monostabiilne, s.t. juhtsignaali puudumisel laseb õhku ühte väljundisse (sinna sunnib teda vedru).

4.2.5. Ventiili solenoid

Silindrit juhitaks solenoidiga parker p2e-kv31j [22] (Sele 7.14), mis oli samuti ettevõttes olemas ja mille näitajad on:

- Toitepinge 230 VAC (50 Hz).
- Maksimaalne võimsus 1,6 VA.

Ventiili saab juhtida kasutades võrgupinget, lülitades vajadusel relee abil kokku solenoidi kuum juhe.

4.3. Ohutusnõuded

Rakise ehitamisel tuli arvestada, et kasutatakse kõrget pinget ja testija peab selle eest kaitstud olema. Selleks kasutasin ABB veekindlat harukarpi IP55 reitinguga (IP55 junction box 00 858 [23]). See on vastupidav kuumusele ja tulele (kuni 960 *°C*), mõõtmed on 310x240x110 mm. Et testija ei saaks karpi katsudes viga (voolu) võib läbi kasti olla ainult plastikust osasid. Samal põhjusel on rakis suletud, nii et testimise ajal toodet puutuda ei saa.

Kuna rakis sisaldab ka liikuvaid osasid, siis peab olema välistatud võimalus, et testija kuhugi vahele kinni jääb. Sellega arvestades on rakise disainimisel arvestatud, et silindri liikuvad osad oleksid varjatud.

4.4. Elektroonika ja muude osade paigutus karbis

Paigutusel oli põhiliseks eesmärgiks mahutada ära kaks plaati elektroonikaga ja ventiil nii, et jääks ruumi ka testri kõrgepinge otsiku kasti ulatuva osa (umbes 80 mm) jaoks. Muud komponendid on väiksemad neid saab asetada teiste vahele (Sele 7.17).

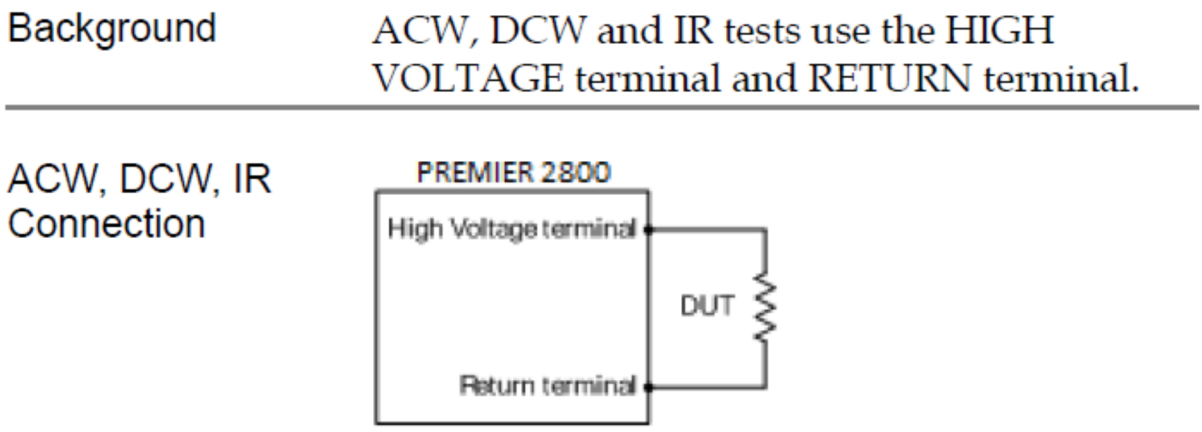
Kasti ühte otsa jäävad elektroonikaga plaadid (Sele 7.18 ja Sele 7.19), kusjuures juhtimisosaga plaat saab kõrgemale, sest sellel on komponente mõlemal pool ja sealt alt lähevad läbi ka pneumaatika voolikud. Ventiil jääb ühte nurka (Sele 7.20), summutitega seina poole ja lisasin seinasse ka väikesed augud, et summutitest tulev õhk pigem välja liiguks. Vabaks jääb üks nurk kuhu saab asetada ühenduse testri kõrgepinge otsiku jaoks (Sele 7.12, Sele 7.10). Mikrokontrolleri saab asetada ventiili kohale (Sele 7.20). Otsustasin kontrolleri alla panna

plaadi, millele see kinnitub. See kaitseb ka kontrollerit ventiili eest (kuigi otseselt mingit ohtu oodata ei ole, on parem olla ettevaatlik).

Kasti kaane sees on ühes ääres 3 auku (Sele 7.23), kust lähevad läbi LED'id, mis näitavad testimise olekut. LED'ide poolsel küljel on ka augud läbi mille kinnitatakse silindri rakis kaane külge.

4.5. Ühendus testriga

Valitud toodete puhul (Sele 7.3 ja Sele 7.5) võib vaja minna kõrgepinge (alalis- või vahelduvvoolu) või isolatsiooni testi. Kõigi puhul on vaja testriga samasugust ühendust (Sele 4.1). Seega on vaja rakisega ühendada vaid kõrgepinge otsik („High Voltage probe", Sele 7.8) ja maanduse otsik („Return probe", Sele 7.9). Maanduse otsiku jaoks on olemas pistik, mis sarnane [24]. Kõrgepinge otsiku saab ühendada surudes see vastu juhtivat pinda, mis on ühendatud juhtmega (Sele 7.10).Polt on vedru abil surutud vastu testri kõrgepinge otsikut, mida hoitakse kinnitusega kinni (Sele 7.12). See lahendus võimaldab kiirelt tester rakisega ühendada ning elektrit juhtivad osad on testija eest varjatud.



Sele 4.1. Toote ühendus testriga ([17] *lk27).*

Rakise siseselt on ühendus vastavalt (L3), kus:

- SV1a – PE grupp.
- SV1b – IN grupp.
- SV1c – OUT grupp.
- SV2a – testri maanduse otsik.
- SV2b –testri kõrgepinge otsik.

Toote ühendused (Sele 7.11) on jaotatud kolme gruppi: IN, OUT ja PE. Testis mõõdetakse nende vahelist takistust või vastupidavust kõrgepingele. Seda, milliste gruppide vahel mõõtmist teostatakse määrab test, tavaliselt on tootel kaks testi, mis teostab mõõtmist kahe erineva grupi vahel. Testide erinevad ühendused teostatakse rakise sees releede abil. Edasi on IN, OUT ja PE ühendused ühendatud tootele ühendatava pistiku pinnidega (kokku 15) (ühendus on nagu Sele 7.24 peal). Seda, milliste pinnidega grupid ühendatakse sõltub tootest ja iga toote jaoks on eraldi pistik, millel on teistest erinevad ühendused rakise väljundite ja tootele ühenduva pistiku vahel. Erinevalt rakise sisestest ühendustest, millest kasutatakse sama toote jaoks erinevaid, on ühendus rakise ja toote vahel sama ühe toote iga testi jaoks. Erinevaid pistikuid kasutatakse, sest kliendiga jõudsime kokkuleppele, et ei ole mõtet kasutada lisareleesid ning pistiku vahetamise aeg ei ole väga pikk. Selleks, et kasutada ühte pistikut oleks vaja olnud lisaks 4 (sobiks ainult olemasolevate toodete jaoks) kõrgepingereleed. Uute toodete jaoks sobiva ühe pistikuga rakise ehitamine oleks keeruline. Selleks oleks vaja 30 kõrgepinge releed (ühe pooluseline), et pistiku iga pinni (kokku 15) saaks ühendada maanduse või kõrgepinge väljundiga. See lahendus nõuaks teistsugust lahendust ja vajalikud releed võtaksid liiga palju ruumi. Seega sai otsustatud, et tuleb kasutada erievate ühendustega pistikuid, et saaks testida ka uusi tooteid.

4.6. Silindri rakise ehitus

Silindri ja toote jaoks sai ehitatud rakis (joonised, leht 1) (Sele 7.15 ja Sele 7.16), kuhu saaks lihtsalt toote sisse libistada ja mis hoiaks toodet ka kinni kui silindri abil pistikut toote peasse surutakse või välja tõmmatakse. Toote plaadi ääred jäävad seinas olevatesse soontesse ja toodet hoiab paigal peale toote sisse lükkamist lisatav otsa plaat, mille liigutamine on kerge.

Silindri õhu portidele on kinnitatud vooluregulaatorid, et reguleerida silindri liikumise kiirust. Neist lähevad õhuvoolikud edasi elektroonika karbi seina küljes olevatesse pneumaatilistesse läbiviikudesse (plastikust süda, et ükski metallosa ei nähtav nii seest- kui ka väljastpoolt), kust lähevad voolikud edasi ventiili. Silindri küljes olevast pistikust lähevad juhtmed silindri alt (jalgade vahelt) elektroonika karbi küljes olevasse pistikusse.

4.6.1. Alusplaat

Alusplaat (joonised, leht 6) on rakist kokku hoidev osa, mis kinnitab rakise ka elektroonikat sisaldavale karbile.

4.6.2. Küljeplaat

Küljeplaadid (joonised, leht 4)hoiavad toote plaati kinni mõlemas oleva soone abil, plaadid on peegelpildis. Ülesandeks on ka katta silindri liikuvad osad.

4.6.3. Katteplaat

Katteplaadi (joonised, leht 3) ülesandeks on katta toode ja silindri liikuv osa, et testija ei saaks neid testi ajal puutuda.

4.6.4. Silindri jalg

Silindri jalad (joonised, leht 7) on silindri õigele kõrgusele asetamiseks. Ühendus alusplaadiga on lihtne, et silindrit saaks koos jalgadega kiiresti eemaldada. Jala ja silindri vahele on jäetud seibid, et saaks vajadusel silindri kõrgust vähendada ja on võimalik ka silindri kõrgust suurendada seibide lisamisega.

4.6.5. Otsaplaat

Otsaplaat (joonised, leht 2) (Sele 7.21) ei lase testijal testi ajal toodet puutuda ja hoiab ka toodet kinni, et see pistiku pesasse lükkamise ajal ei liiguks. Kaks soont plaadi allumises otsas on tootel olevate LED’ide pärast. Plaadi mõõtmed peavad avadesse täpselt mahtuma kuid

liikumine ei tohi olla raske. Väike lokse on lubatav kui toote plaat selle tõttu testimise ajal eriti ei liigu.

4.6.6. Pistiku hoidja

Plaat (joonised, leht 5) (Sele 7.22), mis kinnitatakse silindri kelgu külge ja millele kinnitub pistik. Pistik on eemaldatav, et seda vahetada saaks. Plaadi materjalike on metall ja ta on suhteliselt paks (5mm), et pistiku asend selle paindumise tõttu ei muutuks.

5. ELEKTROONIKA

5.1. Eritingimused

Kasutatavate testide maksimaalne pinge on 1 kV vahelduvvoolu, kusjuures maksimaalne vool on vaid 6 mA. Kliendiga läbi rääkides otsustasime, et rakisel võiks olla ka varu juhuks kui tulevikus võib tootmisse tulla uusi tooteid, mis nõuavad kõrgemat testi pinget. Seda arvestades valisime 3 kV ja 10 mA (suurele voolule ei ole vajadust, testi raskuse määrab pinge) kui nõuded, millele rakis peab vastama. Vajalik varutegur on leitud paragrahvis 6.1.

Kõrge pinge nõuab piisavaid vahekaugusi ja isolatsiooni, et ei toimuks läbilööki. Vool on suhteliselt madal, see ei sea juhtmetele ja radadele eriti suuri mõõtmete nõudmisi. Hea oleks ka kõrgepinge osa ja madalama pingega juhtimisosa eraldamine. Kuna kõrgepingereleed on küllaltki suured, siis on optimaalne asetada kõrgepinge osad teisele trükkplaadile.

5.2. Nõuded trükkplaatidele ja juhtmetele

5.2.1. Nõuded trükkplaadile ja radade disainile

Trükkplaatide dielektriline tugevus on 25,6 kV/mm (650 V/mil) [25] [26] ja rohkem. [27]soovitab arvestada plaadi dielektriliseks tugevuseks 11,8 kV/mm (300 V/mil), mis arvestab ka plaadi amortiseerumist kasutuse jooksul. Siit näeb, et tavaline trükkplaat (1,5 mm paksune, arvestades ka vase pindu võik võtta arvutusteks 1,4 mm) on piisav (vajalik oleks $\frac{5}{11{,}8} =$ 0,42 mm paksune plaat). Varutegur on selgelt piisav (üle kolme).

Elektriradade vahede valikule on mitmeid erinevad soovitusi [28] [27]. Parima arvutus-meetodina tundub valem (5.1).

$$s = \frac{U_{rms} * 3}{k_E} = \frac{U * 3}{k_E * \sqrt{2}} \qquad (5.1)$$

, kus U – lubatav pinge [V],

U_{rms} – lubatava pinge ruutkeskmine [V],

k_E – lubatav pinge 1 mm vahe kohta [V/mm].

Kui arvestada, et lubatav pinge peaks olema 5 kV ja kasutades konservatiivsemat hinnangut [14], siis k_E = 2953 V/mm.

$$s = \frac{5000 * 3}{2953 * \sqrt{2}} = 3{,}6\,mm$$

Kõrgepinge rakendustes tuleb trükkplaadi radasid kujundades arvestada läbilöögi ja koroona tekkimise võimalusega [27]. Läbilöök tekib kui pinge kahe punkti vahel on suurem kui isolatsiooni materjali võime sellele vastu pidada. Peamine toote läbipõlemise põhjus on koroona, sest see põhjustab isolatsiooni nõrgenemist. Koroona tekib kui elektriline vool ei löö kahe elektroodi vahelisest isolatsioonist täielikult läbi. Sel juhul ioniseeritakse elektronide poolt gaasi molekulid, millest eralduvad uued elektronid, mis ioniseerivad veel molekule. Elektriväli kiirendab elektronide liikumist ja tekib suur hulk vabu elektrone. Elektriväli suunab need elektronid elektroodide poole ja tekitab sellega voolu läbi isolatsiooni. Juhendi [27] järgi tuleks hoiduda teravatest nurkadest, nii radade pööretel kui patjadel. Õige kumeruste kasutamine võib vajalikku vahekaugust vähendada kolm korda. Kuna mul ei ole suuri kogemusi kõrgepinge radade disainimisel, siis võtan arvesse soovitused ja teen radade pöörded kumeratena, aga radade vahekaugusena kasutan siiski ettevaatlikult suurt väärtust (4 mm).

5.2.2. Nõuded juhtmele

Kõrgepinge jaoks on parimaks juhtme materjaliks alumiinium (kui vaadelda vaid mõistliku hinnaga materjale) [29] [30]. Alumiiniumi puuduseks vase ees on väiksem elektriline juhtivus ja kehvemad mehhaanilised omadused, kuid eelisteks on hind ja kõrgepinge juhtimise omadused (peamiselt koroona tekkimise vähendamine). Kuna rakendus kasutab väga kõrget pinget, siis oleks mõistlik kasutada materjali, mis selle jaoks parem ehk alumiiniumi.

Juhtme minimaalse läbimõõdu määrab maksimaalne kantav vool. Kõrgepinget juhtivad juhtmed peavad suutma kanda vaid 15 mA. Selle põhjal sobiks [31] [32] AWG39 ehk 0,0889 mm läbimõõduga juhe. Relee juhtsignaali kandvad juhtmed peavad kannatama kuni 30 mA ja selleks sobiks AWG36 ehk 0,127 mm läbimõõduga juhe. Siiski oleks efektiivsem kasutada suuremat juhet, sest ruum ei ole piiravaks asjaoluks ja suuremal juhtmel on väiksem takistus. Mõistlik oleks kasutada näiteks 1 mm läbimõõduga juhet ehk AWG18.

5.2.3. Nõuded juhtme kattele

Juhtmete kate peab andma piisava isolatsiooni, et see kaitseks kõrge pinge eest. Katte materjaliks kasutatakse peamiselt silikooni, polüuretaani, polüetüleeni ja fluoropolümeere [33]. Neist esimesed kolm on laiemalt kasutatavad. Nende eelised ja puudused on ära toodud all (Tabel 5.1) (Tabel 5.2) (Tabel 5.3).

Tabel 5.1. Silikooni eelised ja puudused.

lk5 [33]	**Eelised**	**Puudused**
Elektrilised	Dielektriline tugevus, vastupidavus koroonale.	Dielektriline konstant.
Mehhaanilised	Painduv madalatel temperatuuridel.	Madal vastupidavus lõikele, kõrge hõõrdekonstant, kõrge erikaal.
Keskkonna	Kõrge vastupidavus kiirgusele.	Tekitab silikoonõli, madal vastupidavus õlile, kehv tekstuur.
Rakenduse piirangud	Madala profiiliga pakkimine.	Kaal, vajalik paks isolatsioon.

Tabel 5.2. Polüuretaani eelised ja puudused.

([33] lk 5)	**Eelised**	**Puudused**
Elektrilised	Elektriline suutlikus.	Dielektriline vastupidavus pingele.
Mehhaanilised	Vastupidavus lõikele, vastupidavus abrasioonile, paindlikkus.	Kõrge paindlikkusega variant on kehva tekstuuriga.
Keskkonna	Vastupidav lahustitele, vastupidav UV suhtes, vastupidav kiirgusele, vastupidav hallitusele.	Vastupidavus temperatuurile ja reostusele.
Rakenduse piirangud	Kasutatakse peamiselt ümbrisena.	

Tabel 5.3. Polüetüleeni eelised ja puudused.

([33] lk 6)	**Eelised**	**Puudused**
Elektrilised	Dielektriline konstant, isolatsiooni takistus.	
Mehhaanilised	Vastupidavus abrasioonile, lai valik erinevaid variante.	Abrasioonile vastupidavad variandid on jäigad.
Keskkonna	Keemiline vastupidavus, madal hõõrdekonstant.	Vastupidavus temperatuurile.
Rakenduse piirangud		Paindlikkus.

Arvestades erinevate materjalide eeliseid ja puudusi oleks parimaks kattematerjaliks silikoon. see on piisavalt paindlik ja hea isolatsiooniga. Paksust isolatsiooni kihist tingitud suur juhtme mõõt ei ole takistuseks ning juhe peab siiski olema paindlik.

5.3. Elektriskeem

5.3.1. Juhtimisosa plaat

Lülitatavaid komponente (LED'id ja releed) juhitakse AVR mikrokontrolleri väljundutest, kuid kuna nende vooluvarustus ei ole piisav, tuleb kasutada võimenduseks muid komponente. Lisaks on vahepeal (enne võimendavat elementi) optosidestid, et pakkuda galvaanilist eraldatust mikrokontrolleri ja kõrgepingereleede (ning muude elementide) vahel. Optisidesteid võiks ka võimendava elemendina kasutada, aga antud juhul valitutel ei ole väljundvool (paragrahv 6.3) sisendtingimustel piisav, et juhtida kõrgepingereleesid (ega ka teisi elemente). Seetõttu kasutan Darlingtoni transistoritega elementi ULN2803A [34] (tellitud Farnellist [35]). ULN2803A on võimas, kõrge voolutarbimisega (kuni 500 mA) elementide juhtimiseks mõeldud kiip. Saades sisendisse kõrge pinge nivoo, muudetakse sellele vastav väljund madalaks ja vooli liigub ühise kõrge pinge nivooga pinni ja madalale nivoole seatud pinni vahel. Seega eraldab mikrokontrollerit kontrollitavatest elementidest optosidestid ja ULN2803A. Igasuguse rikke korral peaks mikrokontroller olema kindalt kaitstud.

5.3.2. Releede plaat

Rakise sees on vaja ümber lülitada erinevaid ühendusi. Vajalikke variante on kaks kuid kasutades ühe poolusega releesid (kahe poolusega kõrgepinge releed on palju kallimad, odavam on kasutada kahte ühe poolusega releed ühe kahe pooluselise asemel) on vaja nelja releed. Kuna kasutatavad pinged on kõrged, siis tuleb kasutada kõrgepinge releesid. Vajalikud ühendused on välja toodud varem (paragrahv 4.5) ja realiseeritud lahendus lisades (L3).

5.4. Komponentide paigutus

Komponendid paigutasin kahele plaadile, et oleks võimalik eraldada kõrgepingel töötav osa juhtimisosast. Kõrgepingel töötavale osale jääb vaid neli kõrgepingereleed, mis võtavad ka oma suuruse tõttu enda alla enamuse plaadist (100x160 mm). Kuna ühendusi pole väga palju, siis saab kõik rajad paigutada ühele poole ja ühepoolne plaat sobib. Kontrolliplaadi peal on ühendusi nii palju, et vaja on kahepoolset plaati (sama suurusega). Komponendid paigutan ühele poolele (ülemine pool), samale poolele jääb ka maanduspind („ground pour“). Teisele poolele jäävad signaalirajad.

Plaatideks sai valitud oomipoe fotolakiga kaetud plaadid (üks ühepoolne [36] ja teine kahepoolne [37]). Plaadid sobivad prototüübi tegemiseks hästi, sest ettevõttes on selle ilmutamiseks vajalik UV lambiga karp olemas. Kuna ilmutmine võtab rohkem aega kui eemaldatavat vaske on palju, siis arvestasin sellega disainimisel ning jätsin suured vasepiirkonnad voolu ja maandusradadele (’power pour’ ja ’ground pour’). Signaalirajad disainisin vastavalt töökindluse põhimõtetele.
Plaatide valmistamine toimus järgmiselt:

- Plaadi radade kujunduse printimine kilele (tindiprinteriga, see kaitseb paremini UV kiirguse eest).
- Plaadi paljastamine UV kiirgusele. Kile läbipaistvatele piirkondade (kohad kuhu vaske ei jää) alla jäävatelt kohtade kaitsekiht muutub UV kiirguse mõjul eemaldatavaks.
- Kaitsva kihi eemaldamine UV kiirgusele paljastatud kohtadest seebikivi lahusega [38].
- Vase eemaldamine kaitsekihita kohtadest naatriumpersulfaadi lahusega [39].

6. ELEKTROONIKA KOMPONENTIDE VALIK

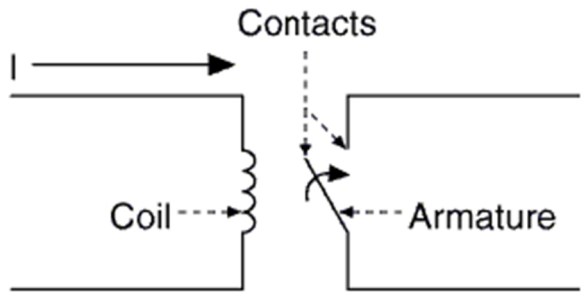
6.1. Kõrgepinge releed

6.1.1. Relee tüüp

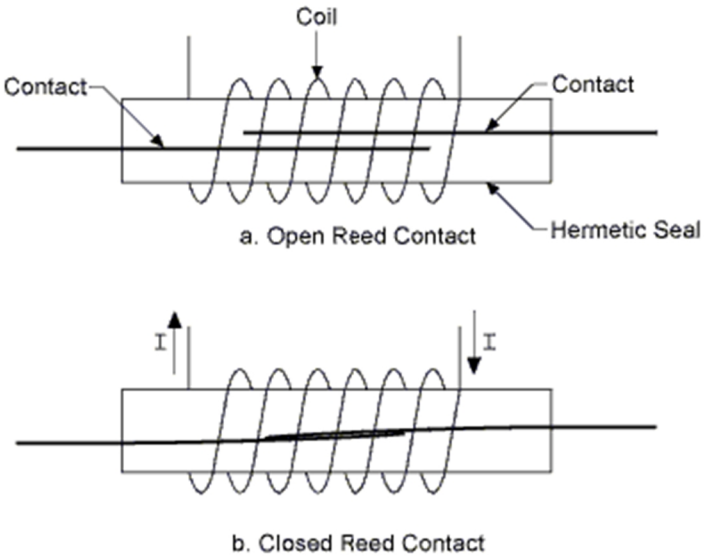
Kuni 5 *kV* releede puhul võib kasutada veel avatud arhitektuuri („open relays“), kuid kõrgem pinge nõuab juba erinevaid lahendusi [40] [41]. Enimkasutatavad on vaakumreleed ja gaasiga (mis annab õhust parem dielektrilise isolatsiooni) täidetud releed.

Kõige sagedamini kasutatavad releede tüübid on [42]:

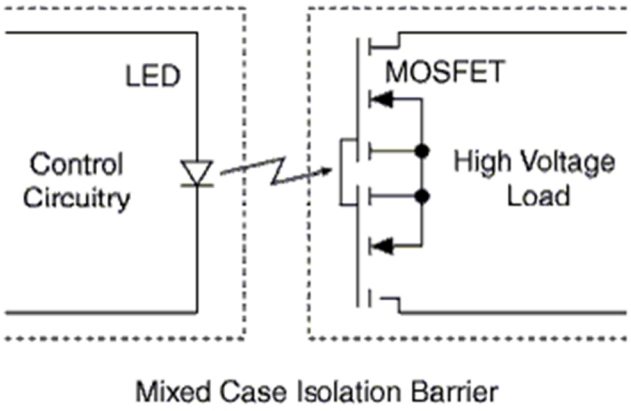
1) Elektromehhaaniline relee (Sele 6.1) – magnetpool liigutab armatuuri, mis avab ja sulgeb kontakte.
2) Reedi relee (Sele 6.2) – pool on ümber kontaktide. Kui pool vool lasta, siis see magnetiseerub ja sulgeb kontaktid.
3) Pooljuhtrelee (Sele 6.3) – Koosneb valgustundlikust MOSFET seadest (kontaktid) ja LEDist.
4) FET lüliti – koosneb CMOS transistoridest. Sisend juhib transistoride baase.



Sele 6.1. Elektromehhaanilise relee tööpõhimõte.



Sele 6.2. Reedi relee tööpõhimõte.



Sele 6.3. Pooljuhtrelee tööpõhimõte

Kõigil on oma puudused ja eelised [42] [43] [44], mida näitab Tabel 6.1.

Tabel 6.1. Relee tüüpide eelised ja puudused.

Eelised	Puudused
Elektromehhaaniline relee (EM relee)	
• Lai valik variante erinevatele sagedustele ja võimsustele. • Kontaktid on omavahel isoleeritud ja galvaaniliselt isoleeritud juhtimisest. • Robustne	• Aeglane lülitamine. • Lühike eluiga. • Vastupidavam läbilöögile. • Häälekas lülitamine.
Reed relee	
• Väiksem ja kergem kui EM relee. • Kiire lülitamine. • Ei vaja minimaalset koormust.	• Läbilöögi oht (voolu hüpe, mis üle lubatud parameetrite võib kontakte kahjustada). • Ei sobi täpseteks madalapingelisteks kasutusteks (relee termiline EMF võib tulemusi märgatavalt mõjutada).
Pooljuhtrelee	
• Kiire lülitamine. • Pikk eluiga. • Juhtimine ja kontaktid on galvaaniliselt isoleeritud. • Väiksem ja kergem kui EM relee. • Pole liikuvaid osasid.	• Suur kontaktide takistus. • Läbilöögi oht (voolu hüpe, mis üle lubatud parameetrite võib kontakte kahjustada). • Puudub kontaktide mehhaaniline eraldatus
Fet lüliti	
• Väga kiire lülitamine • Väga väike • Pikk eluiga.	• Suur kontaktide takistus. • Läbilöögi oht (voolu hüpe, mis üle lubatud parameetrite võib kontakte kahjustada). • Puudub füüsiline isolatsioon (sobib vaid madalale pingele).

Kõige paremini sobib valitud rakenduseks Reedi relee. See on peaaegu sama hea kui pooljuhtrelee, aga on palju odavam (lisaks on kõrgepinge pooljuhtreleed eriti kallid ja ei ole ka lihtsalt saadavad). EM relee ei oleks antud rakenduses töökindel, sest enamus kõrgepinge EM

releesid vajavad vahetevahel kuuma lülitamist (lülitamist, kui kontaktides jookseb vool), et kontakte puhastada. Reedi relee eeliseks on ka minimaalse koormuse nõude puudumine. See on kasulik, sest tester laseb alguses väljundit aeglaselt tõstes ka madalamat pinget (arvestades, et vool on alati madal on ka võimsus sel juhul madal).

6.1.2. Relee vajalikud parameetrid

Relee peab vastu pidama 3 kV pingele ja 10 mA voolutugevusele (vahelduvvool). Voolule vastu pidamine ei ole probleem, sest peaaegu kõik kõrgepingereleed kannatavad sellest rohkem voolu. Arvestades, et testitav toode sisaldab ka mahtuvuslike ja induktiivseid komponente, ei tohi uskuda relee voolutugevuse reitingut [45] [46]. Sel juhul tuleks kasutada vaid maksimaalselt 30 % relee lubatud voolutugevusest, ehk relee peaks võimaldama $\frac{10}{0,3} = 33,3$ mA voolu. Samuti peab arvestama vooluhüpetega pinge rakendamise alguses (millele Reedi relee on suhteliselt vastuvõtlik), mis mahtuvuslike komponentide tõttu võib olla kuni 40 korda suurem. Seega peaks relee kannatama $10 * 40 = 400$ mA vooluhüppeid. Ka relee pingereiting peaks olema kõrgem kasutatavast pingest [47], 1,5 kuni 5 korda [48] või isegi 10 korda mootori puhul [49]. Tavalise kasutuse puhul takistusliku koormusega on konservatiivsete juhiste järgi soovitatav kasutada vaid 50 % relee maksimaalsest pinge reitingust. Arvestades, et testitavatel toodetel ei ole suuri induktiivseid komponente võib kasutada ohutustegurit 2. S.t. relee opereerimise pinge peab olema vähemalt $2 * 3 = 6$ kV.

6.1.3. Relee valik

Kõrgepinge releesid müüakse pigem tellimusena kuid alla 10 kV reitinguga releesid on saadaval ka veebipoodides nagu Farnell [50], RS [51] ja Onlinecomponents [52]. Kõik müüvad sarnaseid releesid, peamiselt Cynergy3 (saadaval kõigis), Meder Electronic (Farnell, Onlinecomponents), Coto Technology (Farnell, Onlinecomponents). Odavaimad hinnad on Farnellis, lisaks räägib selle kasuks fakt, et Note OÜ saab sealt allahindlust. Valida oli kolme erineva relee vahel (kõik erinevatelt tootjatelt) (Tabel 6.2). Kõik releed on normaalselt avatud.

Tabel 6.2. Võimalike releede võrdlus.

	CYNERGY3-DAT71210 [53] [54] **(**Sele 7.27**)**	**COTO TECHNOLOGY - 5501-12-1** [55] [56] **(**Sele 7.28**)**	**STANDEXMEDER - H12-1A83** [57] [58] **(**Sele 7.29**)**
Juhtimise pinge, U_i/V	12	12	12
Maksimaalne lülitamise pinge, U_o/kV	7	7,5	7,5
Maksimaalne lülitamise vool, I_o/A	2	3	3
Pooli takistus, R_c/Ω	150	175	340
Kontaktide takistus, R_o/MΩ	100	-	150
Lülitamise aeg (lülitamine/vabastamine), t_s/ms	3/2	3/3	3/1,5
Lülitamise pinge, U_s/V	>9	3,75-18	-
Vabastamise pinge, U_r/V	<1,25	0,5-2	-
Eeldatav eluiga, operatsiooni	10^6	10^8	$5*10^7$
Mõõdud, x/mm	18,8;60;15,8	19,05;71,12;18,03	30;29,1;18
Hind, hi/EUR	33,53	44,37	55,85

Kõik releed vastavad vajaliku pinge (6 kV) ja voolu (400 mA) nõuetele, ka eeldatav eluiga on kõigil hea kuid DAT71210 on natuke lühema elueaga. Lülitamise aeg on piisavalt lühike ja kõigil sarnane, 5501-12-1 kasuks on madalam lülitamise pinge kuid see ei ole väga tähtis aspekt. Kuna kõik releed vastavad täielikult seatud nõudmistele ja ükski neist ei ole teistest palju parem, siis on mõistlik valida odavaim ehk DAT71210 (relee välimust näeb Sele 7.19 pealt).

6.2. Mikrokontroller

Mikrokontroller ei pea olema väga võimas kuid on nõuded, mida see peab rahuldama:

1. Võimalik programmeerida C või C++ keeles (tavaliselt kõigil IDE abil võimalik).
2. Ühendus arvutiga USB pordi kaudu.
3. Vähemalt 8 väljundit.
4. Programmeeritav mälu vähemalt 64 kilobaiti (arvestatud, et sarnase otstarbega programmide suurus on vahemikus 20-30 kilobaiti ning lisades varu vajalike muude funktsioonide jaoks).
5. Võimalik tellida Farnellist (ettevõtte soov)

Tingimustele vastavad järgmised tooted:

- ATMEGA2560 [59]
- LPCXPRESSO [60]
- AT90USBKEY [61]
- ATSAM4S-XPLD [62]

Valikutest võib kohe eemaldada esimese kõrgema hinna tõttu. Ülejäänute võrdluses (tabel Tabel 6.3) leidsin, et parimaks on AT90USBKEY. Otsustavaks sai IO pinnidele võimaldatav vool, AT90USBKEY on ainus, mis võimaldab juhtida 8 optosidestit (igaüks neist vajab üle 5 mA voolu). Kõik vaadeldavad mikrokontrollerid omavad piisavalt sisendeid/väljundeid ja on piisavalt kiired, 32 bitisel arhitektuuril on teatud eelised kuid need on minimaalsed. Kõik variandid suudavad anda piisavalt voolu kõigi IO pinnide peale kokku, kuid vaid AT90USBKEY (välimust näeb Sele 7.20 pealt) omab piisavalt pinne, mis suudavad eraldi anda piisavat kõrget voolu.

Tabel 6.3. Mikrokontrollerite võrdlus.

Nimi	LPCXPRESSO [60] [63]	ATSAM4S-XPLD [62] [64]	AT90USBKEY - AT90USB1287 [61] [65]
Hind	€21.44	€26.98	€23.92
Arhitektuur	32bit	8bit	32bit
Sisendite/väljundite arv	80	47	48
IO maksimaalne vool (sisse/välja), I_{max}/mA	20 (2 pinni), 4 (ülejäänud) / 20 (1 pin), 4 (ülejäänud)	30 (2 pinni), 4 (9 pinni), 2 (ülejäänud) / sama, mis sisse	20 / 10
IO kogu vool (välja), I_{max_tot}/mA	100	100	100
Kiirus, f/MHz	50	120	16

6.3. Optosidesti

6.3.1. Optosidesti tööpõhimõte

Optosidesti on vajalik, et juhtiv osa oleks eraldatud kõrgepingereleedest. Optosidesti kannab signaali üle elektrilise barjääri kasutades infrapunast kiirgavat dioodi ja fototransistori. Diood põleb heledusega, mis sõltub toitepingest ja voolust. Kui dioodilt paistab fototransistorile piisavalt valgust, siis see muundab selle vooluks, mida võimendatakse transistori voolu võimenduse teguri („current gain ratio“) võrra. Optosidesti puhul näitab fototransistori voolu (väljundvoolu) sõltuvust dioodi voolust CTR ehk voolu ülekandmise faktor. Tegelikult on optosidestil kaks CTR-i kuid praegusel juhul on vaja vaid lineaarse režiimi oma. Optosidesti töötab lineaarses režiimis kui transistoril on pinge suurem, sel juhul on see mitteküllastunud olekus ja CTR sõltuvus dioodi voolust on peaaegu lineaarne. Kui transistoril on madal pinge (alla 1 V, tavaliselt 0,4 V), siis on transistor küllastunud olekus, CTR sõltuvus dioodi voolust on mittelineaarne ja optosidesti töötab lülitina (digitaalloogika režiim). Kuid kuna antud olukorras on vaja kõrgemat pinget, siis kasutame vaid lineaarset režiimi.

6.3.2. Optosidesti valik

Optosidestit juhib mikrokontrolleri IO pin, mis suudab anda kuni 10 mA pingel 3,3 V.

Nõuded optosidestile on:

1. Suudab väljundisse anda 0,93 mA.
2. Lekkevool on alla 50 µA.
3. Kannatab väljundis 12 V pinget.

Olemas on Vishay CNY17-3 [66], seega kasutan seda.

Transistor kannatab kuni 70 *V* kollektor-emitter pinget (Sele 6.4), seega sobib kasutamiseks 12 V juures. Ka kollektori maksimaalne pidev vool (50 mA) on piisav.

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
INPUT				
Reverse voltage		V_R	6	V
Forward current		I_F	60	mA
Forward surge current	$t_p \leq 10$ µs	I_{FSM}	2.5	A
LED power dissipation	at 25 °C	P_{diss}	100	mW
OUTPUT				
Collector emitter breakdown voltage		BV_{CEO}	70	V
Emitter base breakdown voltage		BV_{EBO}	7	V
Collector current		I_C	50	mA
	$t_p/T = 0.5$, $t_p \leq 10$ ms	I_C	100	mA
Power dissipation		P_{diss}	150	mW
COUPLER				
Isolation test voltage between emitter and detector	t = 1 min	V_{ISO}	5000	V_{RMS}
Storage temperature		T_{stg}	-55 to +150	°C
Operating temperature		T_{amb}	-55 to +110	°C
Soldering temperature [1]	2 mm from case, ≤ 10 s	T_{sld}	260	°C
Total power dissipation		P_{diss}	250	mW

Sele 6.4. CNY17-3 maksimaalsed võimed. [47]

6.3.3. Optosidesti arvutused

Antud andmed võib võtta arvutustel aluseks, sest nagu näha graafikult (Sele 6.5) on väljundvoolu muutus väljundpinge muutudes väike kui dioodi vool on väike (antud juhul kindlalt alla 10 mA). 1 mA juures (5 V) on CTR vähemalt 34 % ja 10 mA juures vähemalt 100 %. Sel juhul on 5 mA (mõistlik oletus) juures, lugedes CTR ja sisendvoolu suhe lineaarseks, CTR 63 % (Sele 6.6). 5 mA juures oleks (võttes keskkonna temperatuuriks 25 °C) nCTR (normaliseeritud voolu ülekande faktor, see tähendab, et CTR on viidud valitud keskkonna temperatuuri ja sisendvoolu tingimustele) 0,9 (Sele 6.7).

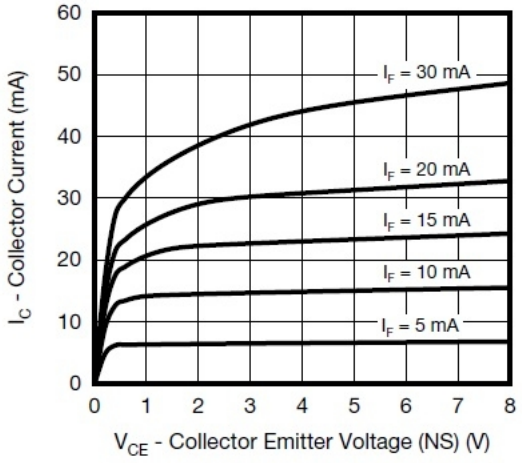


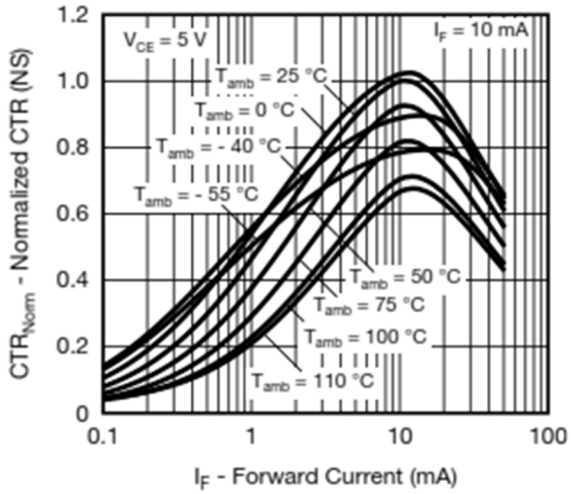
Fig. 6 - Collector Current vs. Collector Emitter Voltage (NS)

Sele 6.5. Kollektori voolu ja emitteri pinge suhe [47].

CURRENT TRANSFER RATIO (T_{amb} = 25 °C, unless otherwise specified)

PARAMETER	TEST CONDITION	PART	SYMBOL	MIN.	TYP.	MAX.	UNIT
I_C/I_F	V_{CE} = 5 V, I_F = 10 mA	CNY17-1	CTR	40		80	%
		CNY17-2	CTR	63		125	%
		CNY17-3	CTR	100		200	%
		CNY17-4	CTR	160		320	%
	V_{CE} = 5 V, I_F = 1 mA	CNY17-1	CTR	13	30		%
		CNY17-2	CTR	22	45		%
		CNY17-3	CTR	34	70		%
		CNY17-4	CTR	56	90		%

Sele 6.6. CNY17-3 CTR. [47]



Sele 6.7. CNY17-3 nCTR sõltuvus voolust. [47]

Kuna väljundvool sõltub sisendvoolust CTR teguri kaudu, siis saab selle leida valemi (6.1) abil [67] [68].

$$I_c = I_F * CTR * nCTR \quad (6.1)$$

, kus I_c – transistori kollektori vool ehk optosidesti väljundvool [mA],

I_F – LEDi edaspidine vool ehk optosidesti sisendvool [mA].

$$I_c = 5 * 0{,}63 * 0{,}9 = 2{,}84\ mA$$

2,84 mA on liiga suur vool. 3 mA suuruse sisendvoolu puhul oleks CTR ≈ 49 % ja nCTR ≈ 0,75. Sel juhul:

$$I_c = 3 * 0{,}49 * 0{,}75 = 1{,}1\ mA$$

1,1 mA on sobiv. Et mikrokontrolleri väljundist tulev vool oleks sobilik, peab vahele paigutama sobiliku takisti, mille väärtuse leiab valemiga (6.2).

$$R = \frac{U - U_F}{I_F} \quad (6.2)$$

, kus R – vajaliku takisti suurus [Ω],

U – mikrokontrolleri väljundpinge [V], U = 3,3 V,

U_F – optosidesti dioodi edaspidine pingelang [V], U_F = 1,65 V (Sele 6.8).

$$R = \frac{3{,}3 - 1{,}65}{3 * 10^{-3}} = 550\ \Omega$$

Standardsuurustes on lähim suurem takisti 560 oomi. Sel juhul leib sisendvoolu valemiga (6.3).

$$I_F = \frac{U - U_F}{R} \quad (6.3)$$

$$I_F = \frac{3{,}3 - 1{,}65}{560} = 0{,}0029\ A = 2{,}9\ mA$$

Ja väljundvool oleks (arvestades, et kuna voolu muutus oli vaid väike, siis kehtivad eelnevad parameetrid):

$$I_c = 2{,}9 * 0{,}49 * 0{,}75 = 1{,}07\ mA$$

See on lubatud voolu piires (tüüpilise ja maksimaalse lülitamise sisendvoolu vahel).

ELECTRICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)							
PARAMETER	**TEST CONDITION**	**PART**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
OUTPUT							
Collector emitter capacitance	V_{CE} = 5 V, f = 1 MHz		C_{CE}		5.2		pF
Collector base capacitance	V_{CE} = 5 V, f = 1 MHz		C_{CB}		6.5		pF
Emitter base capacitance	V_{CE} = 5 V, f = 1 MHz		C_{EB}		7.5		pF
Thermal resistance			R_{th}		500		K/W
COUPLER							
Collector emitter, saturation voltage	V_F = 10 mA, I_C = 2.5 mA		V_{CEsat}		0.25	0.4	V
Coupling capacitance			C_C		0.6		pF
Collector emitter, leakage current	V_{CE} = 10 V	CNY17-1	I_{CEO}		2	50	nA
		CNY17-2	I_{CEO}		2	50	nA
		CNY17-3	I_{CEO}		5	100	nA
		CNY17-4	I_{CEO}		5	100	nA

Sele 6.8. CNY17-3 elektrilised omadused. [47]

6.4. Pingemuundur

Et valida kõige optimaalsema võimsusega pingemuundur, on vaja teada kui suur võib olla skeemi osade voolu tarbimine.

Releede voolu tarbimine oleneb nende juhtimismähise takistusest ja toite pingest ning on leitav valemiga (6.4).

$$I_r = \frac{U_S}{R_c} \qquad (6.4)$$

, kus I_r – relee voolutarbimine [A],

U_s – relee toitepinge [V], U_s = 12 V,

R_c – relee juhtmähise takistus [Ω].

Kõrgepingereleel (Tabel 6.2) on takistuseks 150 Ω, selle järgi on voolutarbimiseks:

$$I_r = \frac{12}{150} = 0{,}08\ A = 80\ mA$$

Ventiili solenoidi juhtimise releel (paragrahv 6.6.1) on takistuseks 330 Ω, selle järgi on voolutarbimiseks:

$$I_r = \frac{12}{330} = 0{,}0364\ A\ \approx 37\ mA$$

Tabel 6.4. Toitevoolu võimalik suurus.

Skeemi osa (arv)	**Voolu tarbimine kokku [*mA*]**
Punane LED (1)	14,7 (paragrahv 6.5)
Roheline ja kollane LED (2)	36,2 (paragrahv 6.5)
Kõrgepinge releed (4)	320
Relee ventiili solenoidi juhtimiseks (1)	36,4
Optosidesti väljundi vool ULN2803 juhtimiseks (8)	8,6
Kokku:	415,9

Kuigi skeemi kõik osad ei tööta mitte kunagi koos (kõigi osade lülitamine toimub järjestikuselt mitte paralleelselt kuid esineb kattuvusi, kus järgmine lülitus algab kui eelmine veel voolu tarbib), peaks siiski arvestama selle võimalusega. Samuti tuleb igaks juhuks kasutada ohutustegurit, milleks valin 1,5. Seega peaks pingmuundur suutma anda $415{,}9 * 1{,}5 = 623{,}9$ mA. Kuid suure tõenäosusega võib eeldada, et tarbimine ei ole kunagi suurem kui 400 mA. Muunduri valimisel võtan siiski minimaalseks voolu parameetriks 623,9 mA.

Sobivad muundurid on toodud all (Tabel 6.5). Kõigi muundurite väljundpinge on 12 VDC ja valitud on vaid kaetud variantide hulgast.

Tabel 6.5. Sobivad pingemuundurid.

	VIGORTRONIX - VTX-214-010-112 [69] [70]	RECOM POWER - RAC10-12SC/277 [71] [72]	XP POWER - ECE10US12 [73] [74]
Sisendpinge piirkond, U_{in}/VAC	90-265	80-305	85-264
Maksimaalne väljundvool, I_{out}/mA	830	840	830
Maksimaalne võimsus, P_{max}/W	10	10	10
Efektiivsus (minimaalne), η/%	75	80	83
Sisend/väljund isolatsioon, U_{iso}/VAC	5075	3750 (1 minut)	4000
Lisad	Ülevõimsuse, ülepinge ja lühise kaitse	Ülepinge, ülevoolu ja lühise kaitse	Ülevõimsuse, ülepinge ja lühise kaitse
Hind, hi/€	8,83	24,02	26,46

VIGORTRONIX jääb teistele küll natuke efektiivsuses alla kuid mitmeid kordi odavam hind teeb ta kõige optimaalsemaks variandiks.

6.5. LEDide valik

Vaja on kolme LEDi: punast, rohelist ja kollast. Valitud LEDid valisin olemasolevate seast ja valituteks said:

- Kollane – Everlight 333-2UYD/S530-A2 (Super Yellow) [75]
- Roheline – Kingbright L-53SGD-5V (Super Bright Green) [76]
- Punane – Kingbright L-53ID-5V (High Efficiency Red) [77]

Tabel 6.6. LEDide valik.

	Kollane	Roheline	Punane
Pingelang, U_F/V	2	6	6
Maksimaalne vool, U_{max}/mA	25	37 (12V juures)	37 (12V juures)
Kasutatav vool (arvestab väikest varutegurit), I_F/mA	15	20	20
Arvutatud takistus, R_{LED}/Ω	667	300	300
Sobiv takisti, R_s/Ω	681	332	332
LEDi voolutarbimine valitud takistiga, I/mA	14,7	18,1	18,1

Takisti, mis tuleks LED' ette panna, et voolutarbimine oleks vajalik leib valemiga (6.5).

$$R_{LED} = \frac{U_S - U_F}{I_F} * 1000 \qquad (6.5)$$

, kus R_{LED} – LEDi ees olev takisti [Ω],

U_F – LEDi pingelang [V], (Tabel 6.6).,

U_S – toitepinge [V], U_S = 12 V

I_F – LEDi vool [mA], (Tabel 6.6).

Kollane LED: $R_{LED} = \frac{12-2}{15} * 1000 = 667\ \Omega$. Sobilik võimalikest suurustest on 681 Ω.

Roheline LED: $R_{LED} = \frac{12-6}{20} * 1000 = 300\ \Omega$. Sobilik võimalikest suurustest on 332 Ω.

Punane LED: $R_{LED} = \frac{12-6}{20} * 1000 = 300\ \Omega$. Sobilik võimalikest suurustest on 332 Ω.

Tegeliku LEDi voolutarbimise kasutades standardsuuruses takistit leiab valemiga (6.6).

$$I_{F\_tegelik} = \frac{U_S - U_F}{R_{LED\_tegelik}} * 1000 \quad (6.6)$$

, kus $I_{F\_tegelik}$ – tegelik LEDi läbiv vool [mA],

$R_{LED\_tegelik}$ – valitud takisti [Ω], (Tabel 6.6).

Kollane LED:

$$\mathrm{I_{F\_tegelik}} = \frac{12-2}{681} * 1000 = 14{,}7\ mA$$

Roheline LED:

$$\mathrm{I_{F\_tegelik}} = \frac{12-6}{332} * 1000 = 18{,}1\ mA$$

Punane LED:

$$\mathrm{I_{F\_tegelik}} = \frac{12-6}{332} * 1000 = 18{,}1\ mA$$

6.6. Muud komponendid

6.6.1. Ventiili solenoidi juhtimine

Relee ventiili solenoidi juhtimiseks on Schrack RTS3L012 [78]. Relee on 230 VAC voolu lülitamiseks 12 V juhtimis pingega. Valituks osutus antud relee, sest see oli kohapeal olemas. Relee on piisavalt võimas, ta lubab kuni 20 A pidevat voolu ja 4000 W voolu lülitamist. Ventiili solenoid, mille lülitamiseks releed kasutatakse tarbib vaid 1,6 W.

6.6.2. Takistite valik

Vajalikud takistid saan NOTE Pärnu laost. Valik vastavalt kättesaadavusele, arvestades, et takisti reiting oleks piisav. Võimsuse, mida takisti peab taluma leiab valemiga (6.7).

$$P = U * I * k \quad (6.7)$$

, kus P – vajalik takisti võimsuse reiting [mW],

U – takisti pinge [V],

I – takistit läbiv vool [mA],

k – varutegur[].

Mikrokontrolleri portidega ühendatud takistid peavad olema vähemalt ($U = 3{,}3$ V ; $I = 2{,}9$ mA; $k = 2$, valisin suurema väärtuse, sest kasutatavad võimsused on väikesed):

$P = 3{,}3 * 2{,}9 * 2 = 19{,}2\ mW$

Kollase LEDi takisti võimsuse reiting peaks olema($U = 12$ V; $I = 14{,}7$ mA; $k = 2$):

$P = 12 * 14{,}7 * 2 = 352{,}8mW$

Punase ja rohelise LEDi takistite võimsuse reiting peaks olema ($U = 12$ V; $I = 18{,}1$ mA; $k = 2$):

$P = 12 * 18{,}1 * 2 = 434{,}4mW$

Saada on vajaliku väärtusega takistid, mille võimsus reiting on 600 mW. Kasutan neid.

6.6.3. Draiver

ULN2803A valisin, sest see on kõige laiemalt kasutatav Darlingtoni transistoritega draiver. Elemendi ülesanne on varustada juhitavaid elemente piisava vooluga kuna optosidesti väljundvool ei ole antud tingimusel piisav. Elemendid ühendatakse ULN2803A väljundporti, mis muutub madalaks kui vastav sisendport muudetakse kõrgeks ja neelab läbi juhiava elemendi voolu.

7. PROGRAMM RAKISE JUHTIMISEKS

7.1. Mikrokontrolleri programm

7.1.1. Programmeerimiskeel

AT90USBKEY peal olev AT90USB1287 kiip lubab kasutada nii C kui C++ keelt. C++ kasutamine teeks programmeerimise palju kergemaks kuid ka C abil saab kõik vajaliku tehtud. Otsustasin C kasuks, sest siis saab kasutada LUFA („Lightweight USB Framework for AVRs“ – väikesemahuline USB raamistik AVR'idele) [79] „Virtual Serial Device“ projekti. LUFA on C keeles kirjutatud vabavaraline teekide kogum, mis lubab suhelda USB ühendusega AVR-iga kasutades virtuaalset jadaporti. Proovisin küll muuta seda projekti C++ põhiseks, aga selle tulemusel hakkasid tekkima mõned probleemid ja otsustasin kasutada siiski C keelt. See võimaldab mikrokontrolleriga suhelda ilma USB ühendust loomata (seega pole vaja ka luua USB draiverit), selle asemel võib luua palju lihtsama jadapordi ühenduse.
Programmis on kasutusel LUFA versioon 140302. LUFA toetub AVR-GCC kompilaatorile, ning kasutan ka WinAVR teeke, versioon on 20100110.

7.1.2. LEDi vilgutamine

Kollase LEDi vilgutamiseks (näitab testi toimumist) on kasutatud kontrolleri enda kella ja taimerit. Kogu kood on failis VirtualSerial.c (L7.3 Programmi fail 20: VirtualSerial.c, lk 204). Taimeri kasutamiseks tuleb see kõigepealt üles seada (Programmi kood 7.1). CTC režiimi kasutamine lubab ise katkestusi (’interrupt’) esile kutsuda ja see töötab paralleelselt CPU muude töödega. Edasi seatakse taimeri pikkus võrreldava väärtusena (tuleb valida vastavalt vajatavale ajale, oleneb ajast, CPU sagedusest ja CPU skaalast), arvestades CPU skaalat (see oleneb mudelist, kasutataval 64). Lõpuks tuleb taimer tööle panna. Taimeri aja läbi saamisel teostatakse meetod ’ ISR(TIMER1_COMPA_vect)’ (Programmi kood 7.2), kuhu lisatud kood muudab kollase LEDi pordi oleku vastupidiseks.

Programmi kood 7.1. Taimeri üles seadmine (L7.3 Programmi fail 20: VirtualSerial.c, lk 204).

```
TCCR1B |= (1 << WGM12); // Configure timer 1 for CTC mode
TIMSK1 |= (1 << OCIE1A); // Enable CTC interrupt
sei(); //  Enable global interrupts
OCR1A   = 25000; // Set CTC compare value, with a prescaler of 64
TCCR1B |= ((1 << CS10) | (1 << CS11)); // Start timer at Fcpu/64
```

Programmi kood 7.2. Meetod ISR (L7.3 Programmi fail 20: VirtualSerial.c, lk 204):

```
ISR(TIMER1_COMPA_vect)
{
      if (yellowBlink == 1){//do on timer completion
            C_FLIPBIT(ly7);} // Toggle yellow LED
}
```

7.1.3. Andmete lugemine puhvrist

Toimub meetodiga ’GetData’ (Programmi kood 7.3). Käsuga ’ if(fgets(getString, CDC_TXRX_EPSIZE, &USBSerialStream)!=NULL)’ saab lugeda puhvrist andmed ja juhul kui selles on sisu tehakse edasi järgmisi toiminguid. Saadud käsk lahutatakse osadeks ’ char** tokens=str_split(getString, ’ ’);’ abil. Sama käsk saadetakse ka tagasi prefiksiga ’mir ’, et saaks veenduda, et käsk jõudis kohale. Tühikuga eraldatud sõnad salvestatakse erinevatesse mälukohtadesse. Kui esimene sõna on ’gg’, siis käivitatakse meetod ’ SendFeedback(*(tokens+1));’, millele antakse parameetriks teine sõna. Vastasel juhul kui esimene sõna on ’ss’, käivitatakse meetod ’ SetPort(*(tokens+1), *(tokens+2));’, millele antakse parameetriks teine ja kolmas sõna.

Programmi kood 7.3. Meetod GetData (L7.3 Programmi fail 20: VirtualSerial.c, lk 204):

```
bool GetData(void)
{
	char getString[CDC_TXRX_EPSIZE];
	char sendString[CDC_TXRX_EPSIZE];
	if(fgets(getString, CDC_TXRX_EPSIZE, &USBSerialStream)!=NULL)
	{//data has command, param1, (param2)
		C_FLIPBIT(led1);//AVR board LED, blinks show data received
		sprintf(sendString, " mir %s\r\n", getString);
		fputs(sendString, &USBSerialStream);//mirror command back
		char** tokens=str_split(getString, ' ');//split command by delimiter
		if(strncmp(*(tokens), "gg", strlen("gg"))==0){//command is get state
			SendFeedback(*(tokens+1));}
		else if(strncmp(*(tokens), "ss", strlen("ss"))==0){//command is send
state
			SetPort(*(tokens+1), *(tokens+2));}
		return true;
	}
	else//no data recieved
		return false;
}
```

7.1.4. Pordi oleku saatmine

AVR pordi oleku saatmiseks arvutisse on ’SendFeedback’ meetod (Programmi kood 7.4). Meetod võtab argumendiks pordi nime, mille põhjal annab vastuseks stringi, mis sisaldab pordi nime ja selle olekut. Vastus saadetakse ’fputs(answerString, &USBSerialStream);’ käsuga.

Programmi kood 7.4. Meetod SendFeedback (L7.3 Programmi fail 20: VirtualSerial.c, lk 204). Osa koodi eemaldatud:

```
void SendFeedback(char* switched_)
{
	char* answerString=NULL;//string with port state
	/*checks incoming string for each port name*/
	if (strncmp(switched_, "pr0", strlen("pr0")) == 0){
		if (C_CHECKBIT(pr0)) answerString = " st pr0 1 \r\n";//if port bit
set high
		else answerString = " st pr0 0 \r\n";}//if port bit set low
	else if(strncmp(switched_, "hv1", strlen("hv1"))==0){
		if(C_CHECKBIT(hv1)) answerString=" st hv1 1 \r\n";
		else answerString=" st hv1 0 \r\n";}
	else if(strncmp(switched_, "hv2", strlen("hv2"))==0){
		if(C_CHECKBIT(hv2)) answerString=" st hv2 1 \r\n";
		else answerString=" st hv2 0 \r\n";}
	//repeating pattern…
	else answerString=" err st na \r\n";
	fputs(answerString, &USBSerialStream);//send answer over stream
}
```

7.1.5. Pordi oleku muutmine

AVR pordi olekut muudetakse ’SetPort’ meetodiga (Programmi kood 7.5). Meetod võtab argumentideks pordi nime ja oleku. Argumendina saadav port muudetakse valitud olekusse. Kui pordi nimena tuli ’a3’ muudetakse kõigi portide olek valituks. Vastus saadetakse nagu varem (paragrahv 7.1.4).

Programmi kood 7.5. Meetod SetPort (L7.3 Programmi fail 20: VirtualSerial.c, lk 204). Osa koodi eemaldatud:

```
void SetPort(char* port, char* state)
{
	char* answerString=NULL;//answer to send afet setting
	int i=-1;//init at unused value
	if(strncmp(state, "1", strlen("1"))==0) i=1;//set i according to state
	else if(strncmp(state, "0", strlen("0"))==0) i=0;
	if (i>=0)
	{
		if(strncmp(port, "a3", strlen("a3"))==0){//command to set all used
ports
			if(i==1) {ChangeAllPorts(true); answerString=" new all 1
\r\n";}
			else {ChangeAllPorts(false); answerString=" new all 0 \r\n";}}
		/*set port bit according to port name and given state*/
		else if(strncmp(port, "pr0", strlen("pr0"))==0){
			if(i==1) {C_SETBIT(pr0); answerString=" new pr0 1 \r\n";}
			else {C_CLEARBIT(pr0); answerString=" new pr0 0 \r\n";}}
		//repeating pattern…
		else if(strncmp(port, "ly7", strlen("ly7"))==0){
			if(i==1) {yellowBlink=1; C_SETBIT(ly7); answerString=" new ly7
1 \r\n";}
			else {yellowBlink=0; C_CLEARBIT(ly7); answerString=" new ly7 0
\r\n";}}
		else answerString=" err new na \r\n";//command not correct, send
error
	}
	else{//state value wriong
		answerString=" err new unk \r\n";}
	fputs(answerString, &USBSerialStream);//send answer over stream
}
```

Pordi oleku muutmiseks on makrod ’C_SETBIT(pordi_nimi)’ ja ’C_CHECKBIT(pordi_nimi)’ (Programmi fail 22: avr035.h, lk 216). Makro seab antud nime all defineeritud (L7.3 Programmi fail 21: VirtualSerial.h, lk 213) pordi vastavalt makrole, kas kõrgeks või madalaks. Ka kollase LEDi vilgutamiseks kasutatakse sarnast makrot ’ C_FLIPBIT(pordi_nimi)’.

7.2. Programm arvutis

Programmi algoritm on toodud välja lisades (Sele 7.26).

7.2.1. Testri programmi salvestamine

Rakisega programmi puhul otsustasin loobuda testri programmi salvestamise võimalusest. Peamiseks põhjuseks oli selle väike kasu võrreldes tekitavate probleemidega. Kirjutatud kood ei töötanud korralikult uue programmiga (koos rakisega) ja otsustasin seda mitte kasutada, sest saadav kasu on väiksem kui ilma rakiseta programmi puhul. Kasu on väike sest rakisega testitavate toodete puhul on paljudel toodetel samade parameetritega testid ja nende puhul, kus parameetrites on erinevusi, on ka pistiku ühendused (paragrahv 4.5) erinevad.

7.2.2. 'SafeSerialPort' klass

Kuna programmi testimise ajal tekkis raskusi AVR kontrolleri ühendusega, siis tuli lisada 'SafeSerialPort' klass. Probleemid tekkisid kasutatava 'Virtual Serial Device' projekti draiveriga, mille tõttu vahel kadus ühendus kontrolleriga ja tavalise jadapordi meetodid ei olnud selle lahendamiseks piisavad. 'SafeSerialPort' klass sulgeb jadapordi ühenduse tõhusamalt ja niimoodi saab vajadusel pordi sulgeda ning seejärel uuesti avada ilma programmi sulgemise vajaduseta.

Juhuks kui pordi lihtsalt sulgemine ei aita, lisasin meetodi 'ResetAvr'. Meetod sulgeb AVR kontrolleriga ühendatud pordi ja palub kasutajal korraks selle juhe välja tõmmata ning seejärel uuesti sisse panna. Meetodi kood on MainWin (L7.2 Programmi fail 16: MainWin.cs, lk 197) klassis.

KOKKUVÕTE

Töö sai tehtud NOTE Pärnu OÜ ja kliendi soovil. See pidi kiirendama toodete testimist ja tegema selle testijale lihtsamaks.

Peale töö alustamist sai selgeks, et täisautomaatse rakise valmistamine läheks liiga kalliks ning palju otstarbekam on kasutada vahetatavaid osasid. Näiteks selle asemel, et kasutada palju kalleid kõrgepinge releesid, saab kasutada vahetatavaid erinevate ühendustega pistikuid.

Esiteks sai valmis kirjutatud programm testri juhtimiseks, mis kiirendas testimist, sest testija ei pea enam iga testi jaoks otsima programmi testri mälust. Lisaks, tagasiside testijatelt ütles, et see vähendas eksimise võimalusi, kus testija valib vale programmi. Kuna programmi abil testides valib testija testi asemel toote nimega testi parameetrite faili, on eksimise võimalus palju väiksem.

Järgmiseks sai valmistatud rakise projekt ja see ka valmis ehitatud ning selle jaoks programm modifitseeritud. Kahjuks nõuab ka rakis testijalt tagasisidet (testija peab vaatama, kas pistik on korralikult ühendatud) kuid Note nõustus, et vajalikud andurid oleksid olnud liiga kallid ja viivis selle tõttu ei ole suur. Rakise kasutuselevõtuga kaob testija vajadus muuta programme testide vahel ja tõsta ümber mõõteotsi. Selle asemel peab ta (esimesel korral eelneb mõõteotsade ühendamine rakisega ja toote valik programmis) sisestama toote rakisesse, sulgema rakise, alustama testi ja veenduma, et silinder on liikunud lõpuni. Seejärel teostatakse kõik tootele vajalikud testid ilma testija abi vajamata.

Programm sisestab andmed logisse ja see võimaldab hiljem vaadata toodete parameetrite muutumist aja jooksul. Lisaks salvestatakse ka programmi teated ja see võimaldab hiljem hinnata programmi töökindlust.

Töö tegemise käigus selgus, et kontrolleri draiver ei ole väga töökindel ja nõuab vahetevahel USB juhtme korraks eemaldamist. See on küll ebameeldiv viivis kuid olukorra parandamiseks oleks vaja spetsiaalselt kirjutatud draiverit, mis nõuaks palju paremaid programmeerimise oskusi kui mehhatroonika õppekava annab. Note teisi rakiseid vaadates sain teada, et see probleem on tavaline, sest testrite programme loovad enamasti ikkagi mehhatroonikud mitte programmeerijad.

Kokkuvõtvalt, saavutatud tulemus on rahuldav. Oleks muidugi olnud parem kui tester oleks saanud täisautomaatne kuid takistuseks sai Note ja kliendi huvi hoida hind madalamal.

SUMMARY

The work was completed according to the wishes of NOTE Pärnu OÜ and its client. It was supposed to make the testing faster and easier for the tester.

After starting the project, it became obvious that making a fully automatic rig would be too expensive and using exchangeable parts is much more optimal. For example, instead of using a lot of expensive high voltage relays, interchangeable connectors with different connections can be used.

At first, a program to control the tester was written. It made the testing faster because the tester didn't have to choose a program for every test the product needs. In addition, it reduced the chance of the tester making a mistake choosing the test. The program uses test parameter files that are named according to the products name, this makes the chance of a mistake much smaller.

Next, the project for the rig was created, used to build it and the program was modified to use with the rig. Unfortunately, the rig still needs feedback from the tester (the tester must check if the connector is correctly attached) but Note agreed that necessary sensors were too expensive and the delay caused by this is small. Using the rig, the tester no longer needs to change programs in between tests and reorganize measurement leads. Instead, he only needs to (first time he also needs to connect measurement leads to the tester and choose the product in the program) insert the product, close the rig and verify that the cylinder has moved to the end. Then, all tests needed for the product are executed automatically.

The program will automatically write the testing data into a log and therefore, it's later possible to see how the product parameters change over time. In addition, program messages are also saved and this can later be used to evaluate the programs reliability.

While constructing the rig, I discovered that he controllers driver is not very reliable and sometimes needs the USB wire to be removed. Although this is an uncomfortable delay, to resolve the situation a custom driver would be needed and that is beyond the programming skills of a mechatronics student. While exploring other testing rigs in Note, I discovered that it's a common problem because the testers are usually created by mechatronics not programmers.

Is summary, the achieved result is satisfactory. It would have been better to make a fully automatic tester but Note and its client wanted a lower cost.

KASUTATUD KIRJANDUS

[1] Sefelec Premier 2804 (Sefelec) *[WWW]* http://www.sefelec.com/en/produit.php?produit=2804 (03.03.2014).

[2] Sefelec Premier series operating manual

[3] A practical guide to dielectric testing (Chroma Systems Solutions) *[WWW]* http://www.chromausa.com/pdf/app-notes/AN-A Practical Guide to Dielectric Testing-092007.pdf (25.02.2014).

[4] Dielectric withstand test (Associated Research) *[WWW]* http://www.asresearch.com/support/faqs/dielectric-withstand.aspx (12.04.2014).

[5] AC vs. DC dielectric withstand testing (J. Lind) *[WWW]* http://www.compwest.com/Library/hipothud.pdf (25.02.2014).

[6] Hipot / electrical safety testing FAQ's (Chroma Systems Solutions) *[WWW]* http://www.hipotsafetytest.com/faqs.php (12.04.2014).

[7] The operator's guide to electrical safety compliance testing (Associated Research) 2004 *[WWW]* http://www.varsatech.com/Operator's Guide.pdf

[8] Testing cables with high voltage (Cirris Systems) *[WWW]* http://www.cirris.com/testing/guidelines/hipot_testing.html (12.04.2014).

[9] Dielectric withstand test (Sefelec) *[WWW]* http://www.sefelec.com/en/dielectricwithstandtest.php (15.04.2014).

[10] Insulation resistance testing *[WWW]* http://www.insulationresistancetesting.co.uk/ (15.04.2014).

[11] Electrical Safety Testing – Help file for most popular standard (Sefelec) *[WWW]* http://www.sefelec.com/en/documents/Helpfileformostpopularstandardsv.4_000.pdf (11.12.2014).

[12] Dielectric Strength Test (Sefelec) *[WWW]* http://www.sefelec.com/en/dielectric-strength-test.php (11.12.2014).

[13] Dielectric Withstand Tester (Sefelec) *[WWW]* http://www.sefelec.com/en/dielectric-withstand-tester.php (11.12.2014).

[14] Insulation Resistance Test (Sefelec) *[WWW]* http://www.sefelec.com/en/insulation-resistance-test.php (11.12.2014).

[15] Ground Bond Test (Sefelec) *[WWW]* http://www.sefelec.com/en/ground-bond-test.php (11.12.2014).

[16] Line Leakage Test (Sefelec) *[WWW]* http://www.sefelec.com/en/line-leakage-test.php (11.12.2014).

[17] Sefelec Premier 2800 manuaal (Sefelec) *[WWW]* http://www.sefelec.fr/upload/produits/manuels/pent4252-premier-28xx-version-013-en-sl.pdf (03.03.2014).

[18] ReSharper programm *[WWW]* https://www.jetbrains.com/resharper (22.12.2014).

[19] Guidelines for selecting pneumatic cylinders (Kenneth Korane) *[WWW]* http://machinedesign.com/news/guidelines-selecting-pneumatic-cylinders (29.05.2014).

[20] Compact guide cylinder - Series MGQ (SMC) *[WWW]* http://content2.smcetech.com/pdf/MGQ.pdf (22.04.2014).

[21] Directional control valves (Parker Pneumatics) *[WWW]* http://www.pneumatic.nl/pdf/parker/p2l viking valves.pdf (15.12.2014).

[22] parker p2e-kv31j (Parker) *[WWW]* http://www.parker.com/portal/site/PARKER/menuitem.bb22d5a82bbb5b147cf26710237ad1ca/?vgnextoid=a2d9b5bbec622110VgnVCM10000032a71dacRCRD&vgnextfmt=default&vgnextcatid=2058199&vgnextcat=SOLENOIDS - P2E SERIES&vgnextdiv=687819&vgnextpartno=P2E-KV31J1&Wtky (15.05.2014).

[23] ABB junction box *[WWW]* http://www05.abb.com/global/scot/scot209.nsf/veritydisplay/5d35360733842688c1256e270058633c/$file/1slt000035d00.pdf (30.05.2014).

[24] Pomona 72913-0 Banana Jack *[WWW]* http://www.newark.com/pomona/72913-0/banana-jack-36a-solder-black/dp/74K0047?utm_campaign=bazaarvoice&utm_medium=SearchVoice&utm_content=Default&utm_source=RatingsAndReviews (25.05.2014).

[25] FR-4 andmeleht (Plastics International) *[WWW]* https://www.plasticsintl.com/datasheets/Phenolic_G10_FR4.pdf (23.04.2014).

[26] PCB laminaadi omadused (Epec) *[WWW]* http://www.epectec.com/pcb/laminate/ (12.04.2014).

[27] High voltage printed circuit design & manufacturing notebook (R. Taezwell and K. Bahl) 2004 *[WWW]* http://www.magazines007.com/pdf/High-Voltage-PCDesign.pdf (12.04.2014).

[28] High voltage spacing (E. Mayerhoff) *[WWW]* http://www.highvoltageconnection.com/articles/highvoltagespacing.pdf (12.04.2014).

[29] Cable Interconnect “System” Design and Component Selection manual: Section A Conductors (Storm Products Company) *[WWW]* http://www.stormcable.com/uploads/wire_conductors.pdf

[30] Selecting an electrical wire (KeyWolf) *[WWW]* http://www.keywolf.com/Selecting_An_Electrical_Wire.php (12.04.2014).

[31] Wire gauge and current limits including skin depth and strength (PowerStream Technology) *[WWW]* http://www.powerstream.com/Wire_Size.htm (22.04.214AD).

[32] AWG cable description *[WWW]* http://www.interfacebus.com/Copper_Wire_AWG_SIze.html (22.04.2014).

[33] Selecting the right cable system for your environment (Paul Warren) 2010 *[WWW]* https://www.gore.com/MungoBlobs/598/801/SelectingtheRightCable_08-11.pdf

[34] ULN280xA andmeleht (STMicroelectronics) *[WWW]* http://www.farnell.com/datasheets/1690352.pdf (28.04.2014).

[35] ULN2803A Farnellis *[WWW]* http://ee.farnell.com/stmicroelectronics/uln2803a/darlington-array-8npn-2803-dip18/dp/1094428 (28.04.2014).

[36] Fotolakiga ühepoolne trükkplaat Oomipoes *[WWW]* http://www.oomipood.ee/en/product/fr4100x160/3500/board-single-sided-1-5mm-l-160mm-w-100mm&s=trükkplaat (22.12.2014).

[37] Fotolakiga kahepoolne trükkplaat Oomipoes *[WWW]* http://www.oomipood.ee/en/product/fr4100x160/3535/board-double-sided-copper-light-sensitive-coating-l-160mm&s=trükkplaat (22.04.2014).

[38] Seebikivi lahus Oomipoes *[WWW]* http://www.oomipood.ee/en/product/dp50/positive-developer-for-pcb&s=naoh (22.12.2014).

[39] Naatriumpersulfaadi lahus Oomipoes *[WWW]* http://www.oomipood.ee/en/product/b327/chemical-agent-etcher-sodium-persulfate-bag-100g (22.12.2014).

[40] V. Gurevich. Electric relays Principles and Applications, 3rd ed. Taylor & Francis, 2006

[41] High voltage relays information (IHS Engineering 360) *[WWW]* http://www.globalspec.com/learnmore/electrical_electronic_components/relays_timers/high_voltage_relays (22.04.2014).

[42] How to choose the right relay (National Instruments) 2012 *[WWW]* http://www.ni.com/white-paper/2774/en/ (28.03.2014).

[43] Reed relay vs. other relays (Meder electronic) *[WWW]* http://www.meder.com/fileadmin/meder/pdf/en/Technical_Documents/Reedrelay_vs._Solid-State.pdf (02.05.2014).

[44] Renesas solid state relays for ATE applications (V. N. Tran, S. Larry, and W. Z. Jiang) *[WWW]* http://www.cel.com/pdf/appnotes/an3008.pdf

[45] Guidelines of relay choosing (Hongfa EPC) *[WWW]* http://www.hf-relay.com/english/jsyfw/jdqzs/200605/68.html (16.04.2014).

[46] Tips for selecting relays (Machine Design: Joseph Zintel) *[WWW]* http://machinedesign.com/archive/tips-selecting-relays (16.04.2014).

[47] L. J. Blackburn and T. J. Domin. Protective relaying principles and applications, 3rd ed. CRC Press, 2007

[48] High voltage relays - detailed information (Ross Engineering Corp) *[WWW]* http://www.rossengineeringcorp.com/products/control/hv-relays/hv-relays-detailed-info.html (16.04.2014).

[49] How to pick a relay (Leach International) *[WWW]* http://www.leachintl2.com/english/english2/vol6/properties/how5.htm (22.04.2014).

[50] Farnell *[WWW]* http://ee.farnell.com/ (04.04.2014).

[51] RS *[WWW]* http://ee.rsdelivers.com/?&cm_mmc=World-Selector-Page-_-Online-Referral-_-MainWorldMap-201211-_-MainWorldMap (04.04.2014).

[52] Onlinecomponents *[WWW]* http://www.onlinecomponents.com/ (04.04.2014).

[53] Cynergy3 - DAT71210 (Cynergy3) *[WWW]* http://ee.farnell.com/cynergy3/dat71210/relay-reed-pcb-12v/dp/1882600 (27.04.2014).

[54] Cynergy3 - DAT71210 andmeleht (Cynergy3) *[WWW]* http://www.farnell.com/datasheets/1295539.pdf (26.04.2014).

[55] Coto Technology - 5501-12-1 (Coto Technology) *[WWW]* http://ee.farnell.com/coto-technology/5501-12-1/reed-relay-spst-no-12vdc-3a-thd/dp/4447750 (26.04.2014).

[56] Coto Technology - 5501-12-1 andmeleht (Coto Technology) *[WWW]* http://www.farnell.com/datasheets/164032.pdf (26.04.2014).

[57] Standexmeder - H12-1A83 (Standexmeder) *[WWW]* http://ee.farnell.com/standexmeder/h12-1a83/relay-reed-high-voltage-12vdc/dp/1079507 (26.04.2014).

[58] Standexmeder - H12-1A83 andmeleht (Meder electronics) *[WWW]* http://www.farnell.com/datasheets/91208.pdf (26.04.2014).

[59] ATMEGA2560 (Atmel) *[WWW]* http://ee.farnell.com/arduino/a000067/atmega2560-arduino-mega2560-rev3/dp/2212779 (02.05.2014).

[60] LPCXPRESSO (NXP) *[WWW]* http://ee.farnell.com/nxp/om13000/dev-board-lpcxpresso-lpc1769/dp/1825878 (02.05.2014).

[61] AT90USBKEY - AT90USB1287 (Atmel) *[WWW]* http://ee.farnell.com/atmel/at90usbkey/at90usb1287-usb-jtag-demo-board/dp/1455078 (02.05.2014).

[62] ATSAM4S-XPLD (Atmel) *[WWW]* http://ee.farnell.com/atmel/atsam4s-xpld/eval-sam4s16-xplained-kit/dp/2215343 (02.05.2014).

[63] LPCXPRESSO andmeleht (NXP) *[WWW]* http://www.farnell.com/datasheets/1803698.pdf (02.05.2014).

[64] ATSAM4S-XPLD andmeleht (Atmel) *[WWW]* http://www.farnell.com/datasheets/1674393.pdf (02.05.2014).

[65] AT90USBKEY - AT90USB1287 andmeleht (Atmel) *[WWW]* http://www.farnell.com/datasheets/5088.pdf (02.05.2014).

[66] CNY17-3 andmeleht (Vishay Semiconductors) *[WWW]* http://www.vishay.com/docs/83606/cny17.pdf

[67] How to use optocoupler normalized curves, 2011 *[WWW]* http://www.vishay.com/docs/83706/83706.pdf

[68] Dealing with low-current optocouplers (Christophe Basso) 1999 *[WWW]* http://powerelectronics.com/markets/dealing-low-current-optocouplers (15.05.2014).

[69] Vigortronix VTX-214-010-112 pingemuundur (Vigortronix) *[WWW]* http://ee.farnell.com/vigortronix/vtx-214-010-112/ac-dc-conv-fixed-1-o-p-10w-12v/dp/2401054 (29.05.2014).

[70] Vigortronix VTX-214-010 andmeleht (Vigortronix) *[WWW]* http://www.farnell.com/datasheets/1803742.pdf (29.05.2014).

[71] Recom RAC10-12SC/277 pingemuundur (Recom) *[WWW]* http://ee.farnell.com/recom-power/rac10-12sc-277/ac-dc-conv-fixed-10w-12v-0-84a/dp/2440313 (29.05.2014).

[72] Recom RAC10-C/277 andmeleht (Recom) *[WWW]* http://www.farnell.com/datasheets/1842302.pdf (29.05.2014).

[73] XP Power ECE10US12 pingemuundur (XP Power) *[WWW]* http://ee.farnell.com/xp-power/ece10us12/psu-encapsulated-10w-single-output/dp/2099456 (29.05.2014).

[74] XP Power ECE andmeleht (XP Power) *[WWW]* http://www.xppower.com/pdfs/SF_ECE05-10.pdf (29.05.2014).

[75] Everlight 333-2UYD/S530-A2 andmeleht (Everlight) *[WWW]* http://www.mouser.com/ds/2/143/333-2UYD-S530-A2-188338.pdf (22.05.2014).

[76] Kingbright L-53SGD-5V andmeleht (Kingbright) *[WWW]* http://www.farnell.com/datasheets/67078.pdf (22.05.2014).

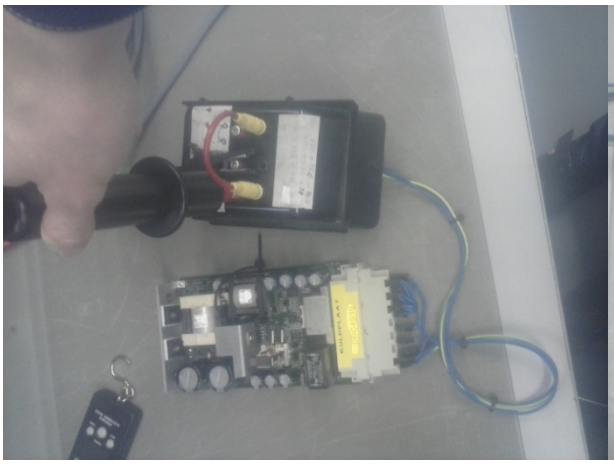
[77] Kingbright L-53ID-5V andmeleht (Kingbright) *[WWW]* http://www.farnell.com/datasheets/67073.pdf (22.05.2014).

[78] Schrack RTS3L012 relee andmeleht (Schrack) *[WWW]* http://www.farnell.com/datasheets/1792654.pdf (15.05.2014).

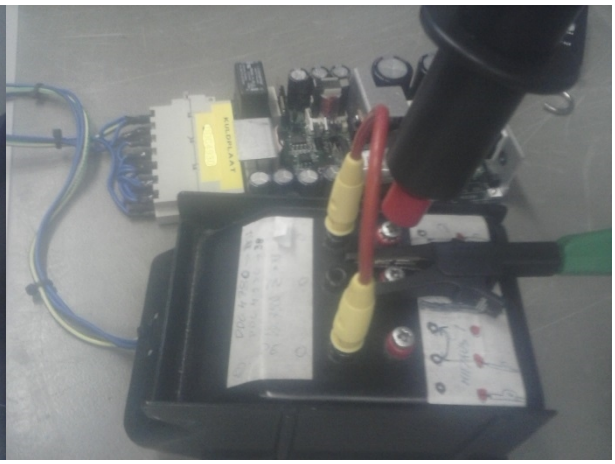
[79] LUFA projekti koduleht *[WWW]* http://www.fourwalledcubicle.com/LUFA.php (15.04.2014).

LISAD

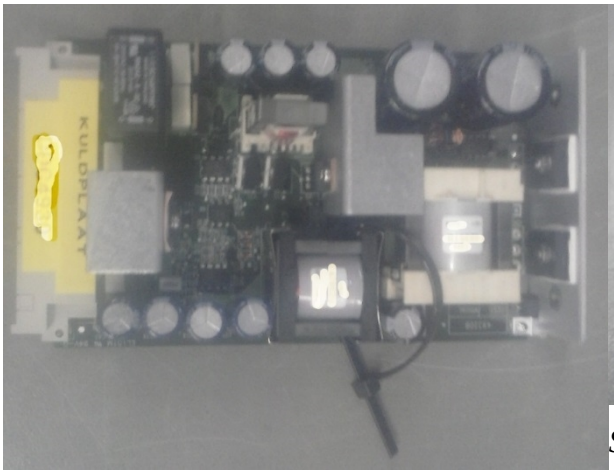
L1. Seled



Sele 7.1. Testimine varem (toode 1).

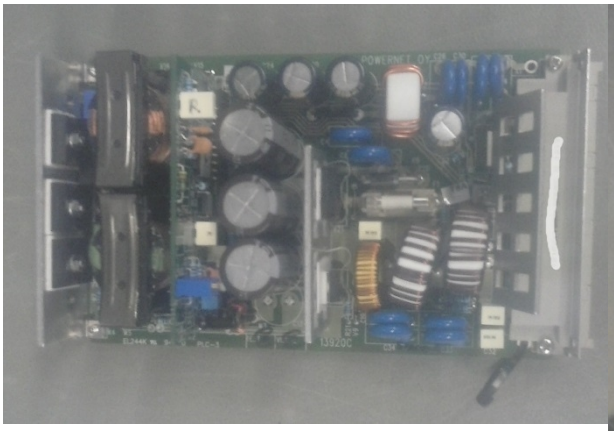


Sele 7.2. Testimine varem (toode 1).

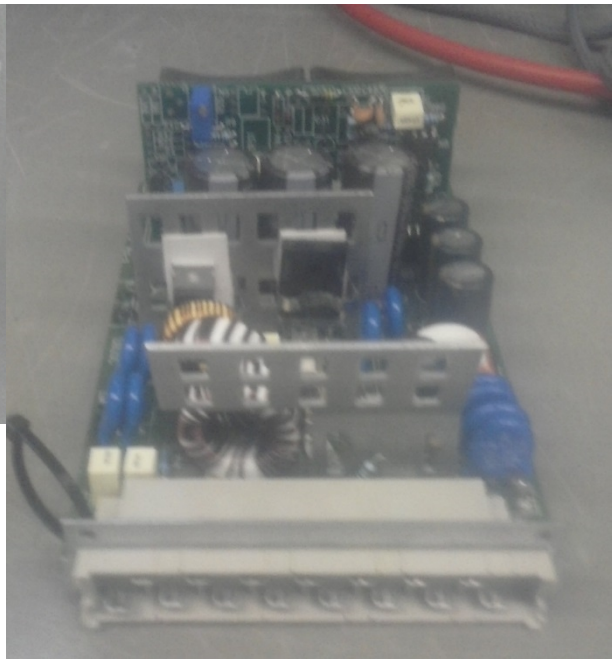


Sele 7.3. Toode 1.

Sele 7.4. Toode 1.



Sele 7.5. Toode 2.

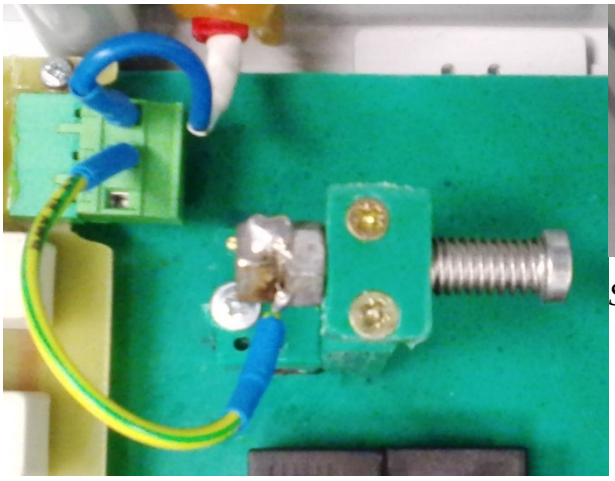


Sele 7.6. Toode 2.

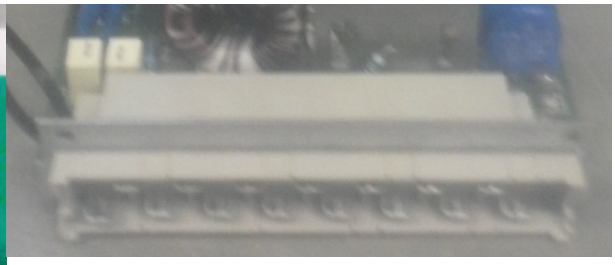
Sele 7.7. Sefelec Premier 2804.

Sele 7.8. Testri kõrgepinge otsik.

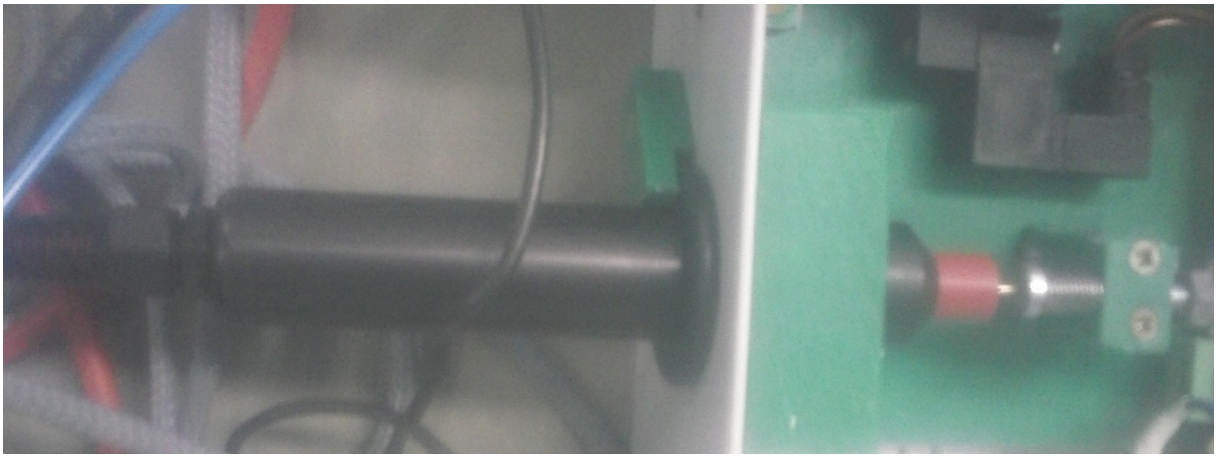
Sele 7.9. Testri maanduse otsik.



Sele 7.10. Rakise ühendus kõrgepinge otsiku jaoks.

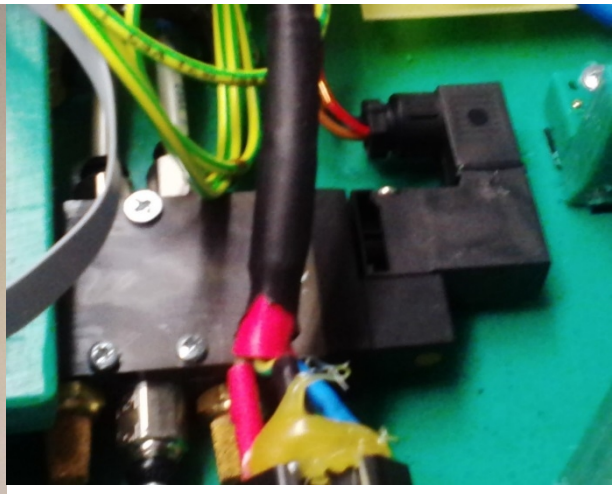


Sele 7.11. Toote pistik.



Sele 7.12. Rakise ühendamine kõrgepinge otsikuga.

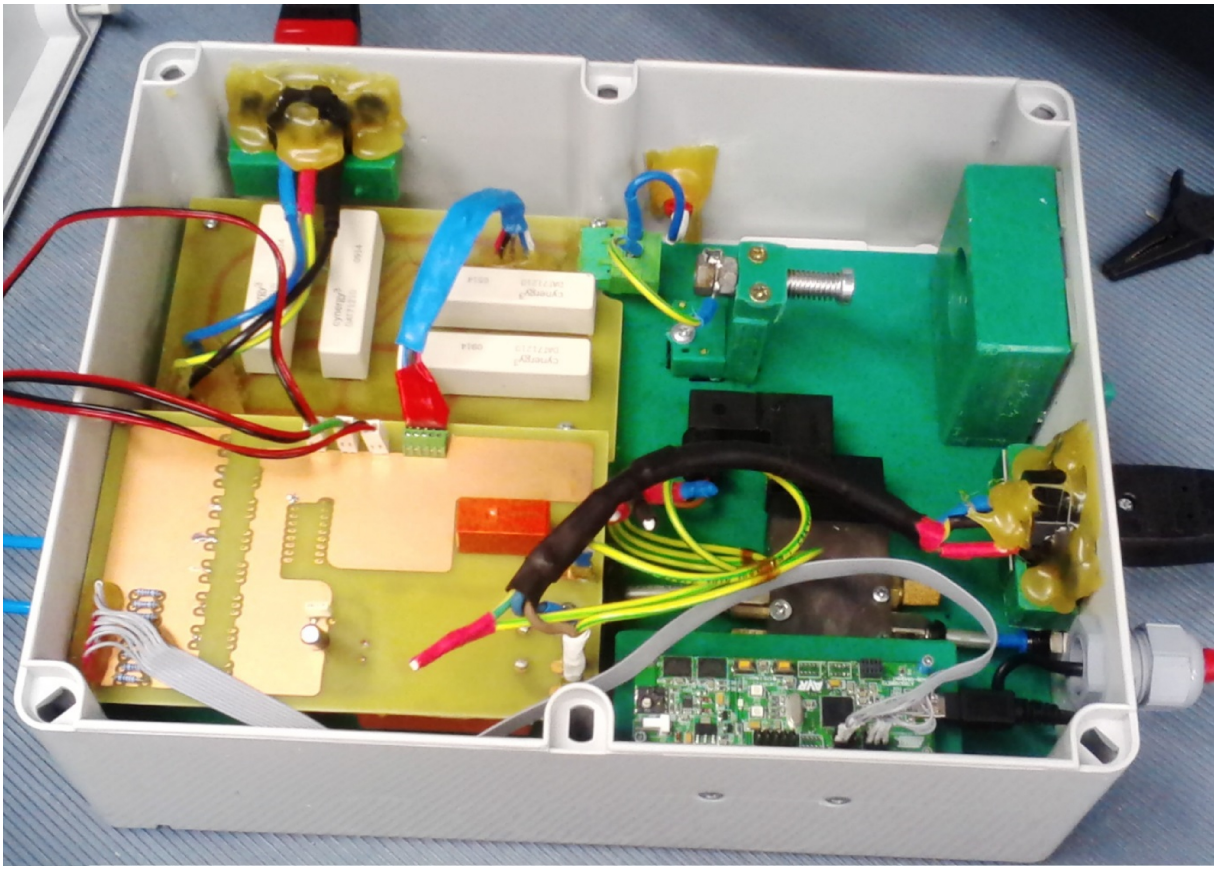
Sele 7.13. Kasutatav silinder.



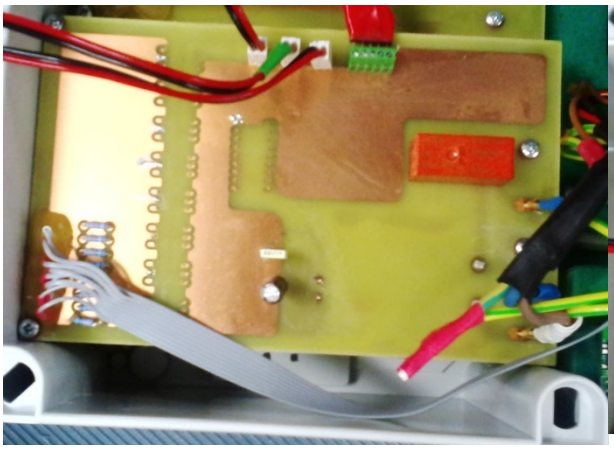
Sele 7.14. Ventiil ja solenoid rakises.

Sele 7.15. Silindri rakis.

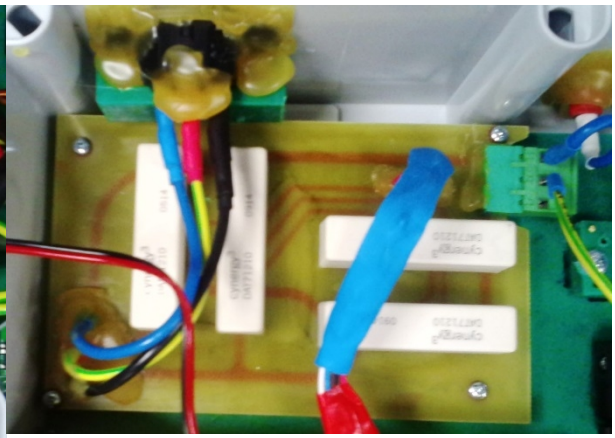
Sele 7.16. Silindri rakise sulgemine.



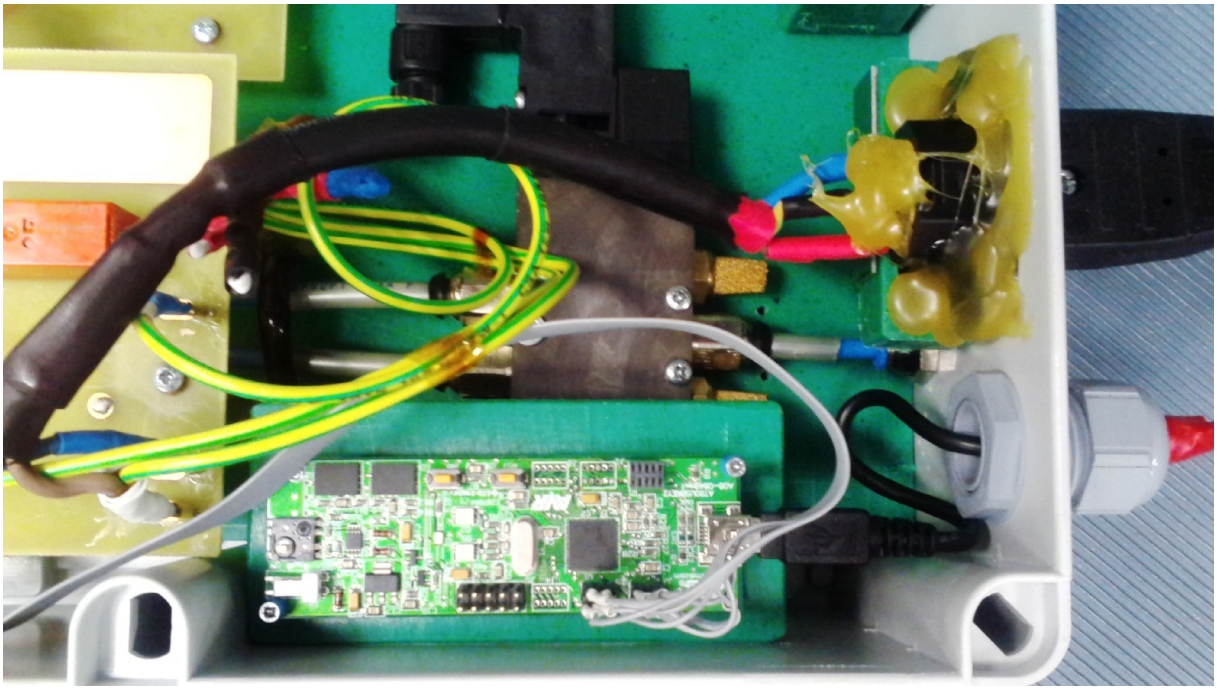
Sele 7.17. Elektroonika karbi sisu.



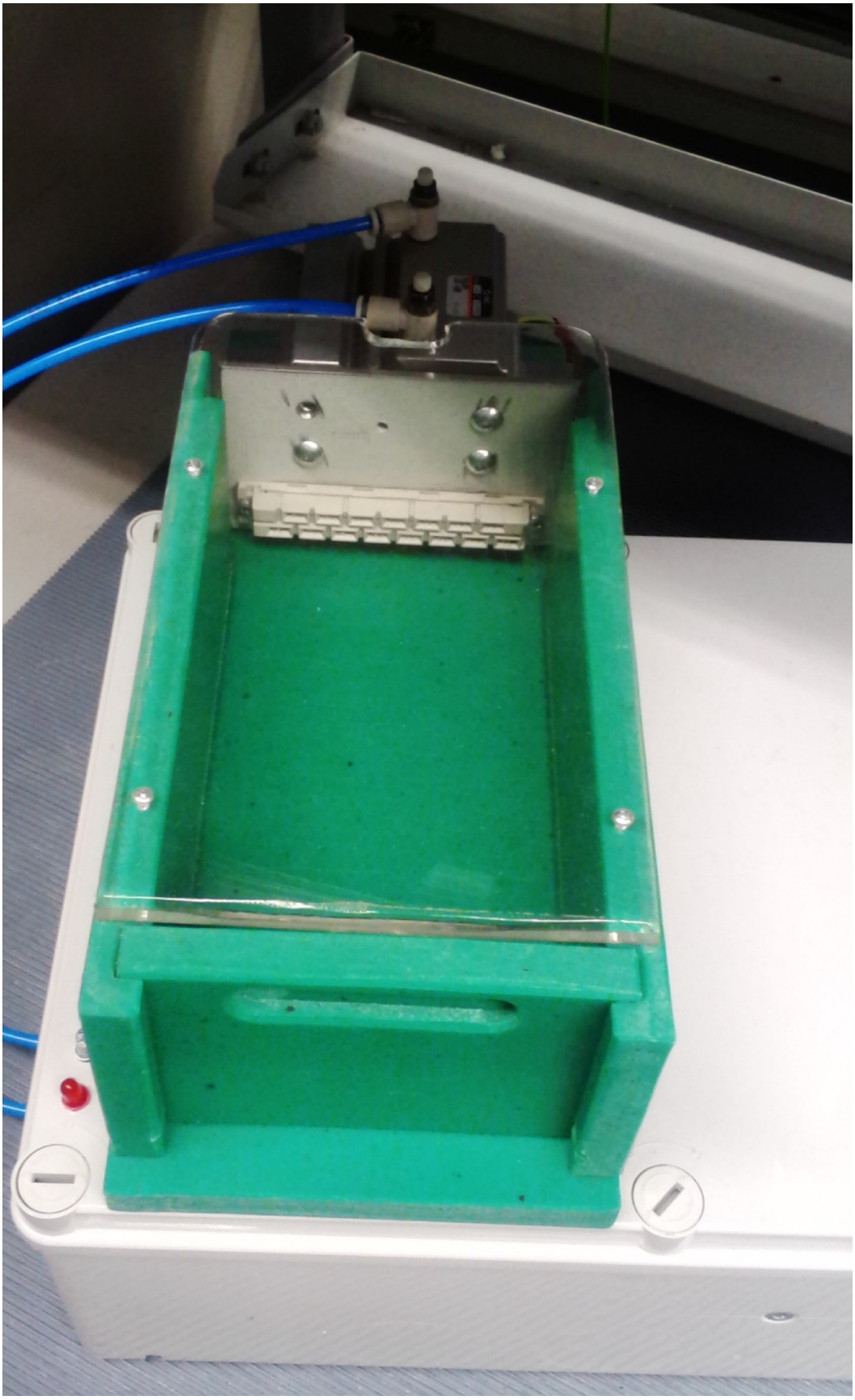
Sele 7.18. Elektroonikaga (juhtimise) plaadi paigutus.



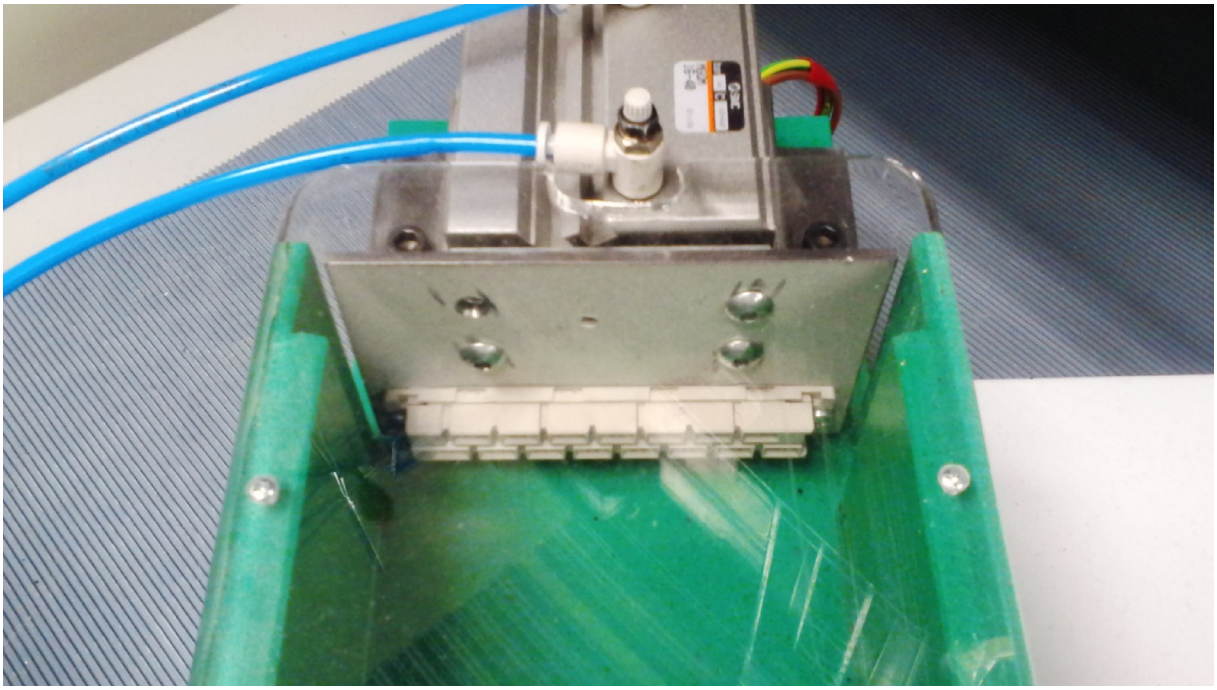
Sele 7.19. Elektroonikaga (releede) plaadi paigutus.



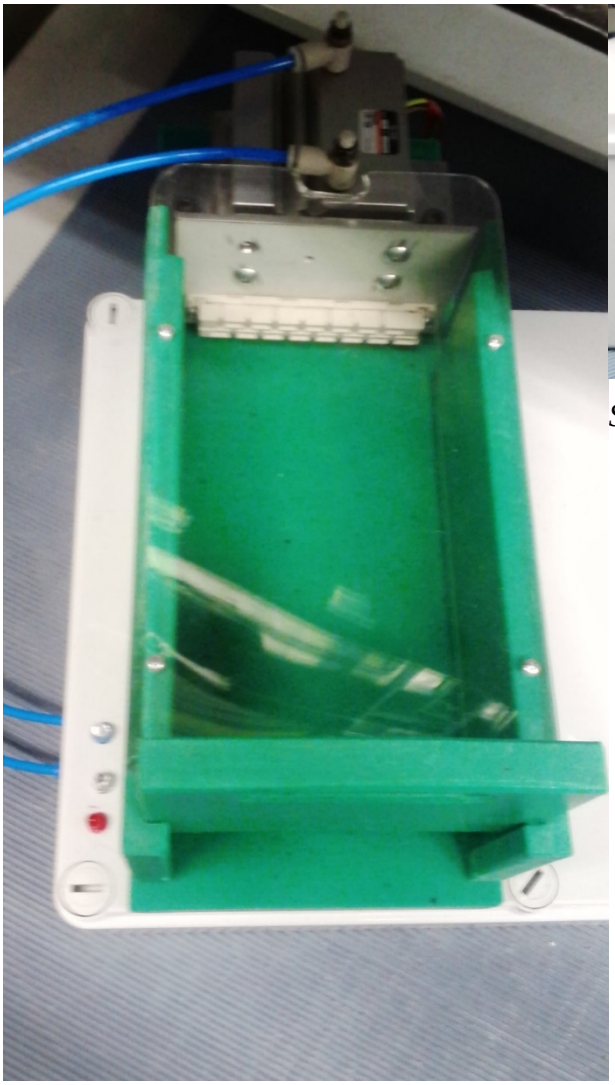
Sele 7.20. Ventiili ja mikrokontrolleri asukoht rakises.



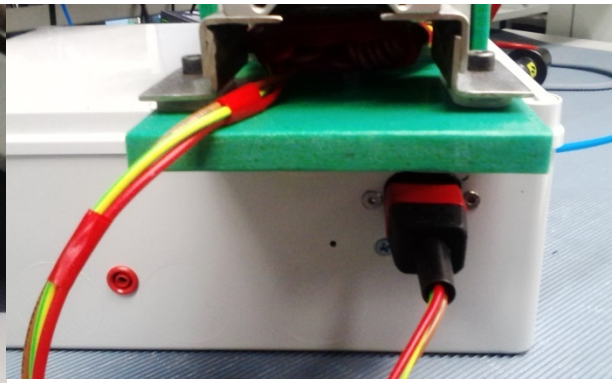
Sele 7.21. Rakise otsaplaadi koht.



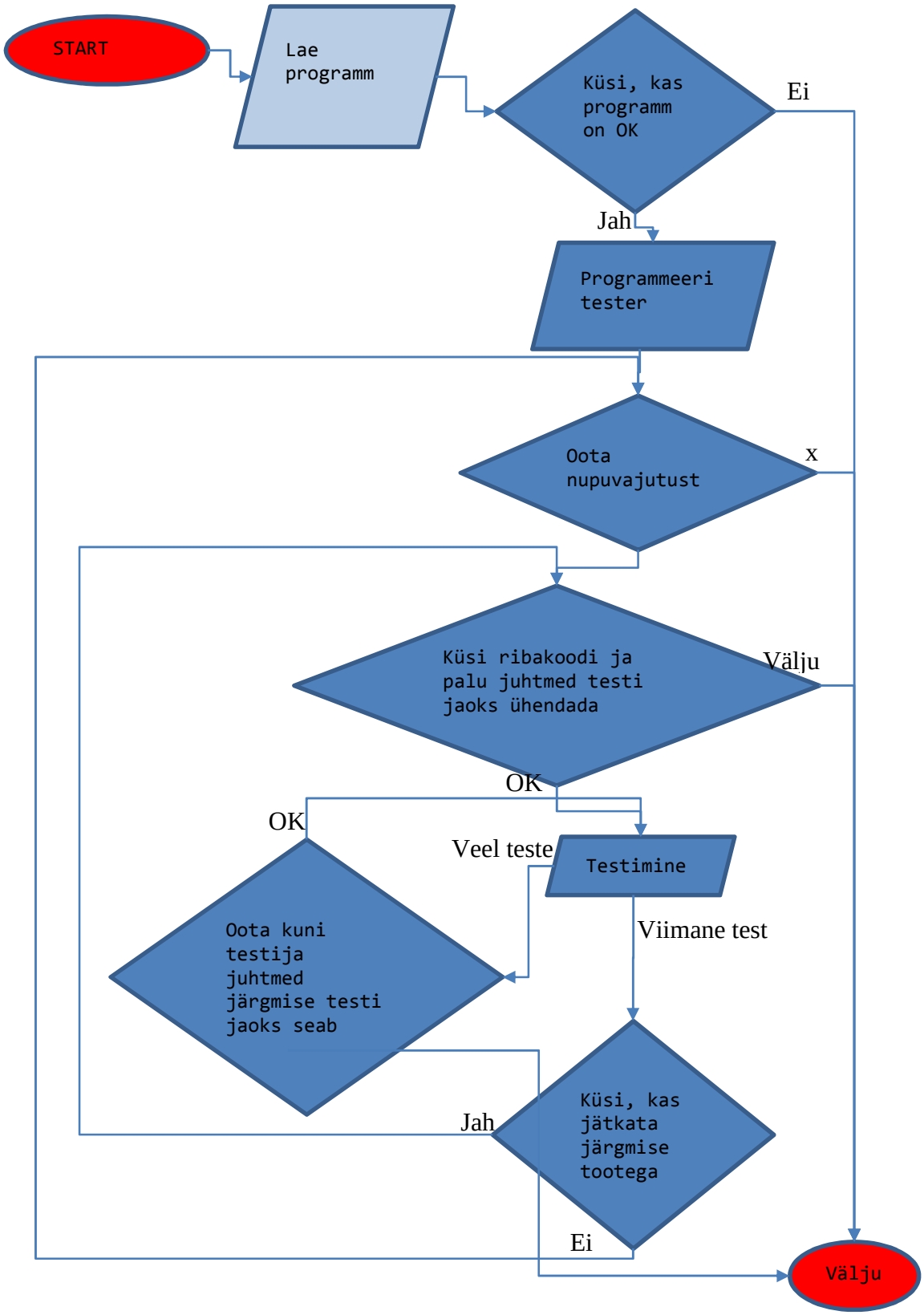
Sele 7.22. Rakise pistik ja selle hoidik.



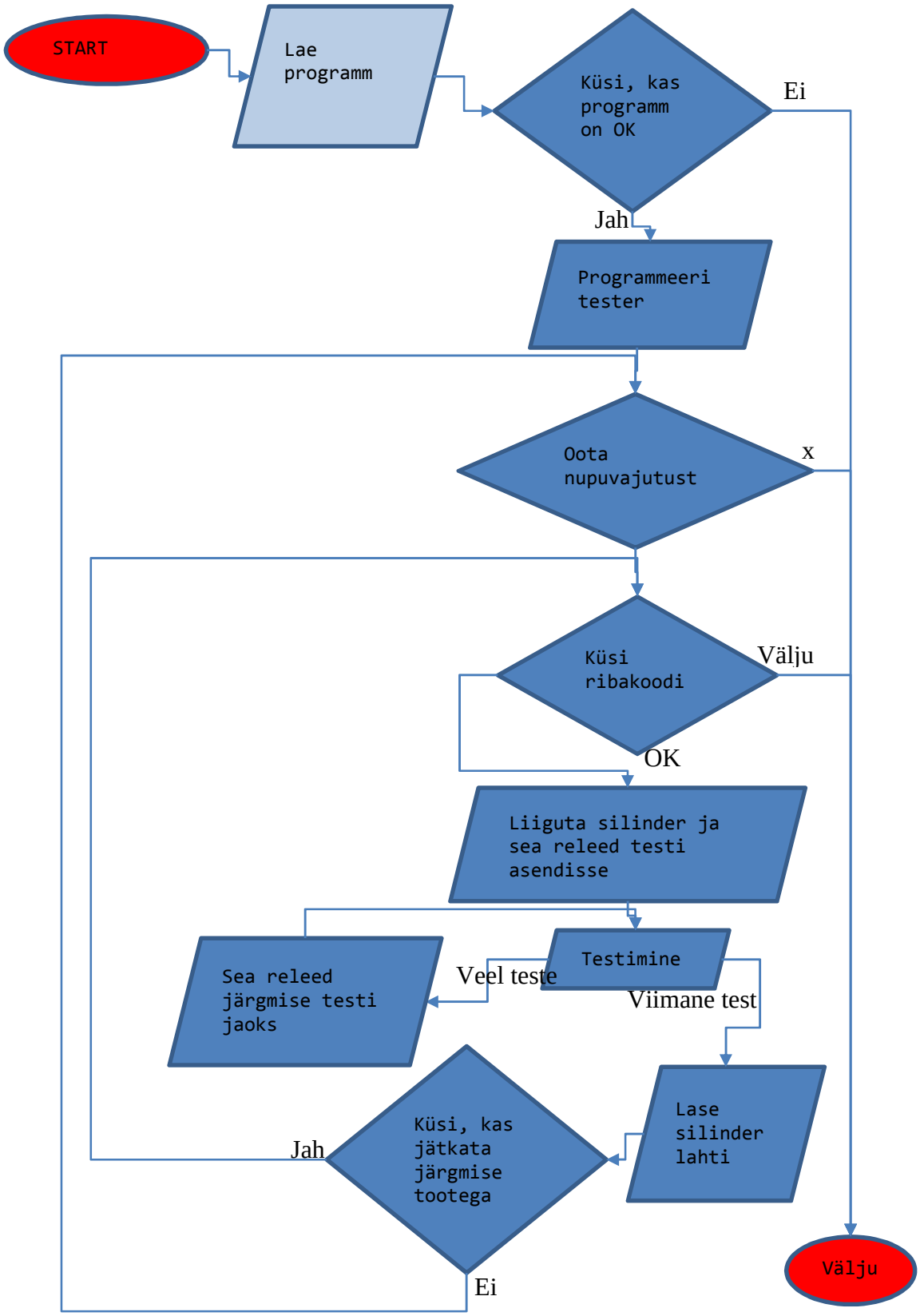
Sele 7.23. Rakis pealtvaatest.



Sele 7.24. Silindri tagune ühendus.



Sele 7.25. Algoritm ilma rakiseta programmile.

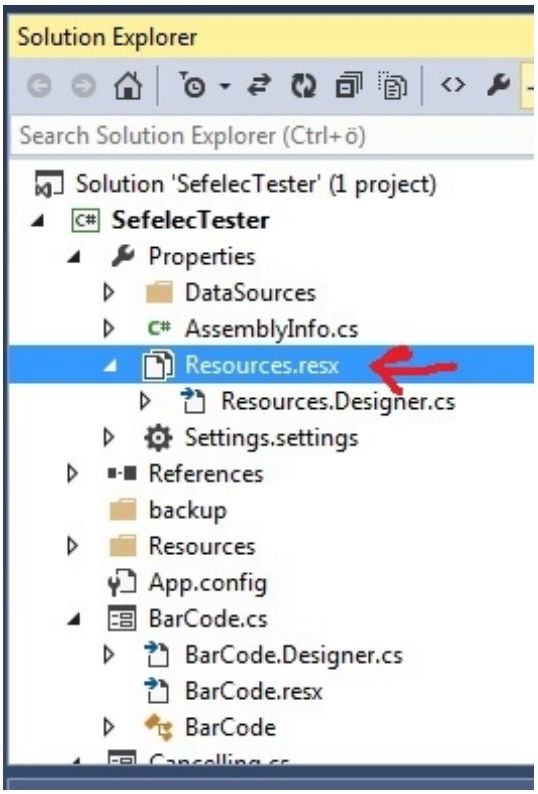


Sele 7.26. Algoritm rakisega programmile.

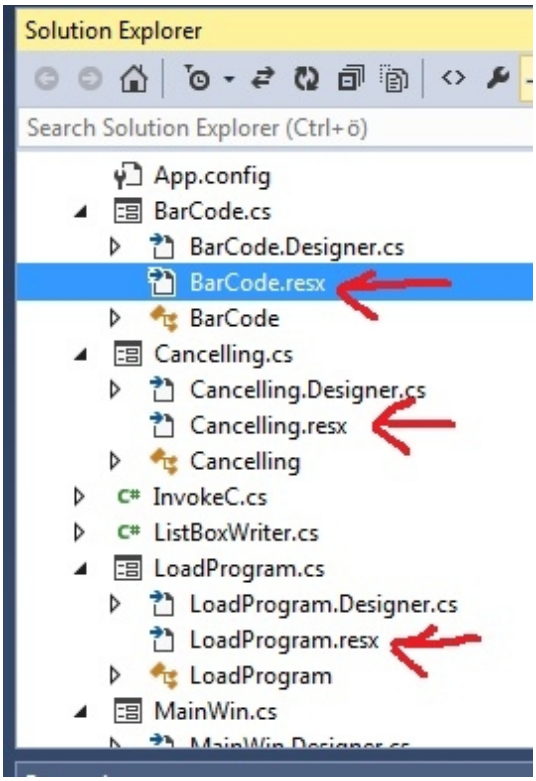
Sele 7.27. CYNERGY3 - DAT71210.

Sele 7.28. COTO TECHNOLOGY - 5501-12-1.

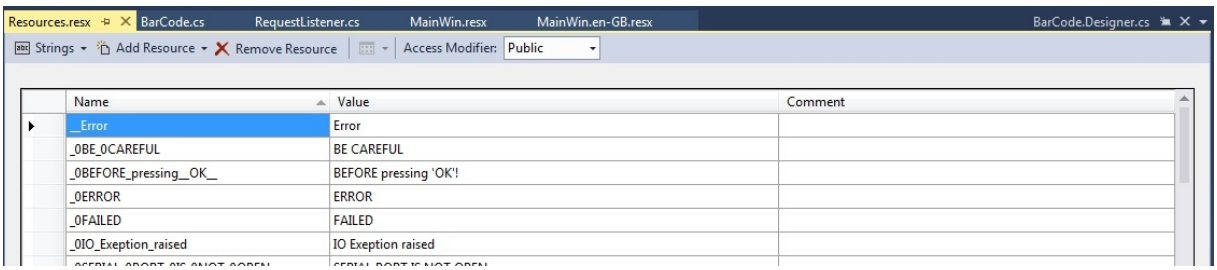
Sele 7.29. STANDEXMEDER - H12-1A83.



Sele 7.30. Programmi resx faili asukoht.

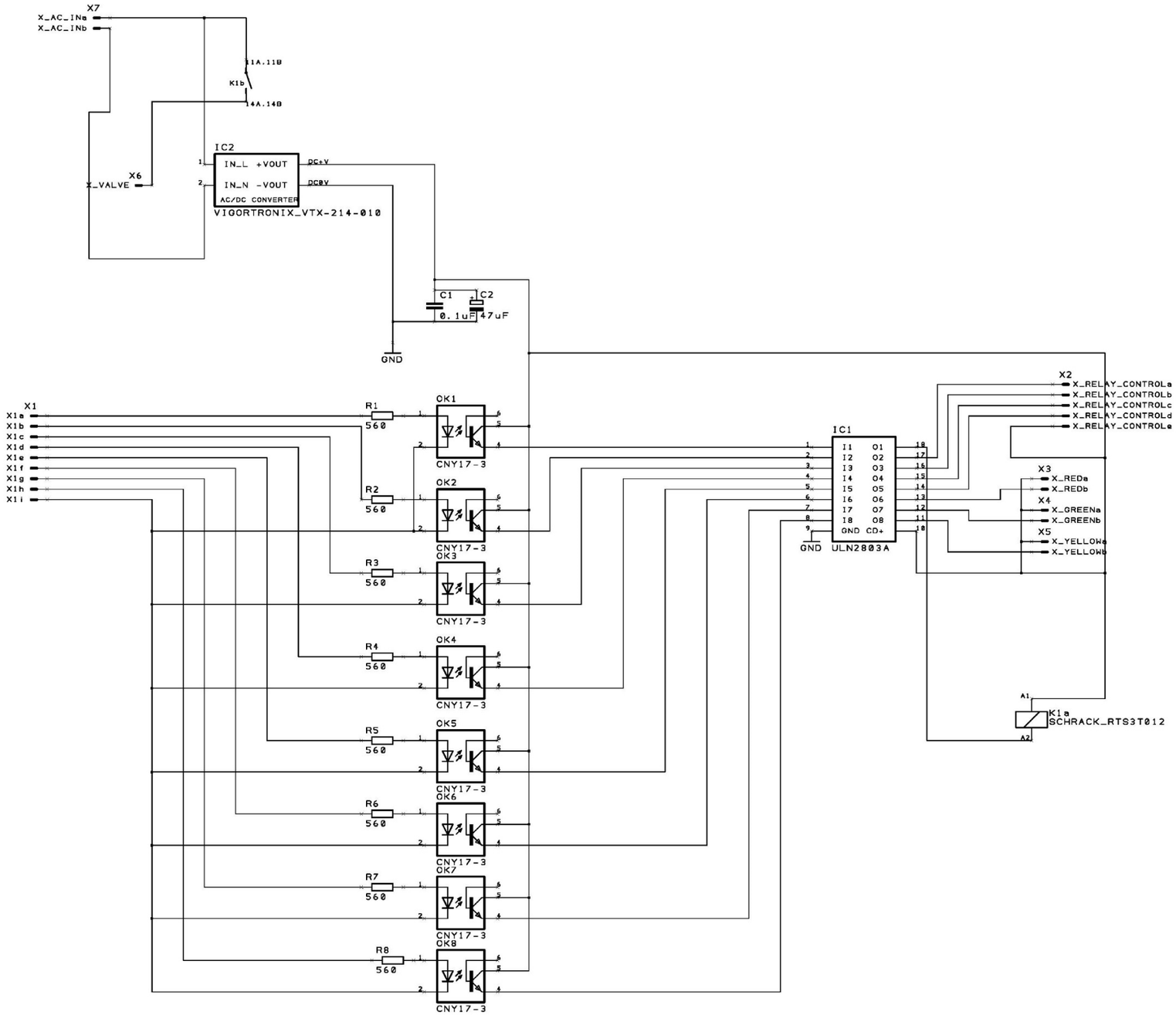


Sele 7.31. Programmi klasside resx faili asukohad.

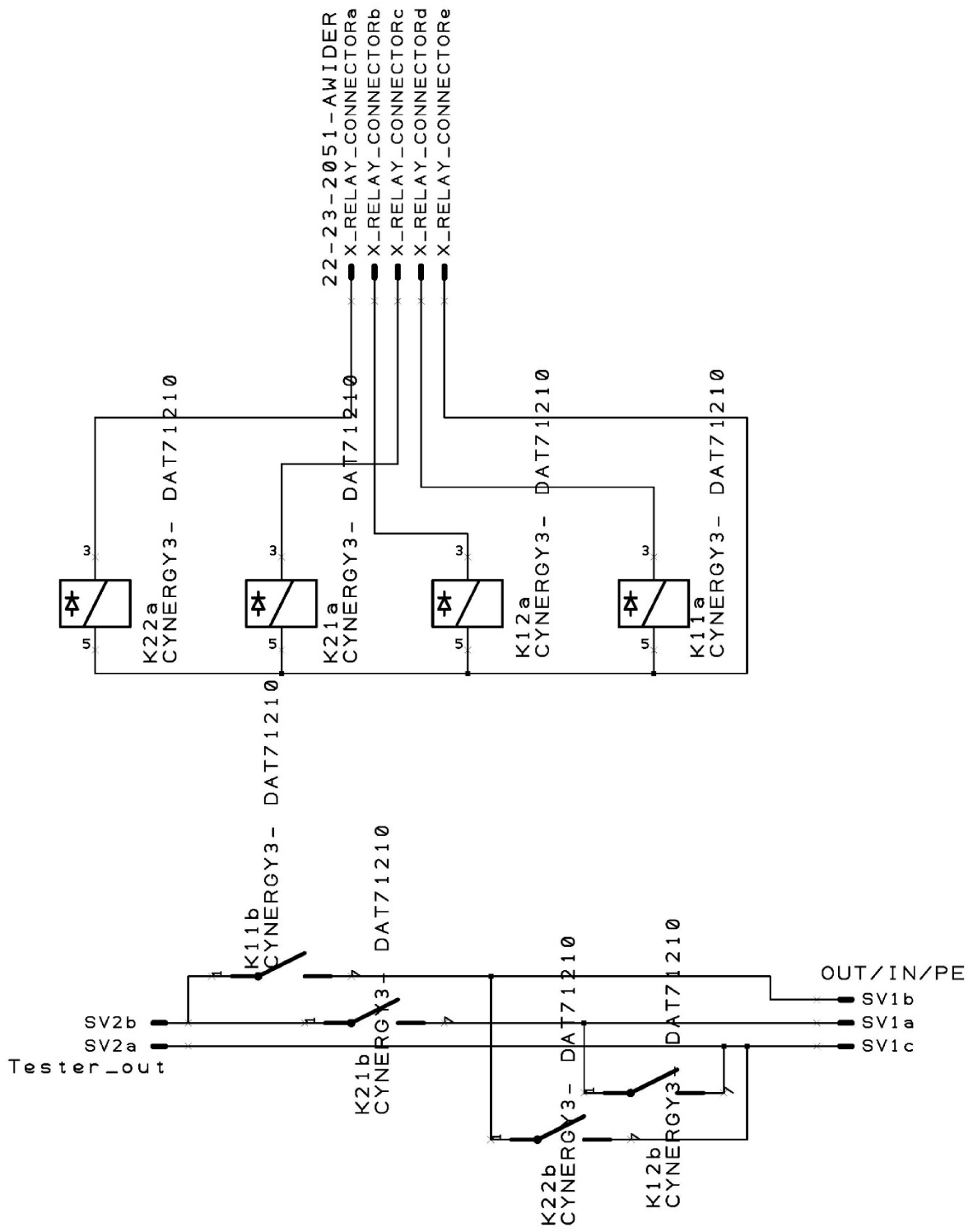


Sele 7.32. Resx faili sisu.

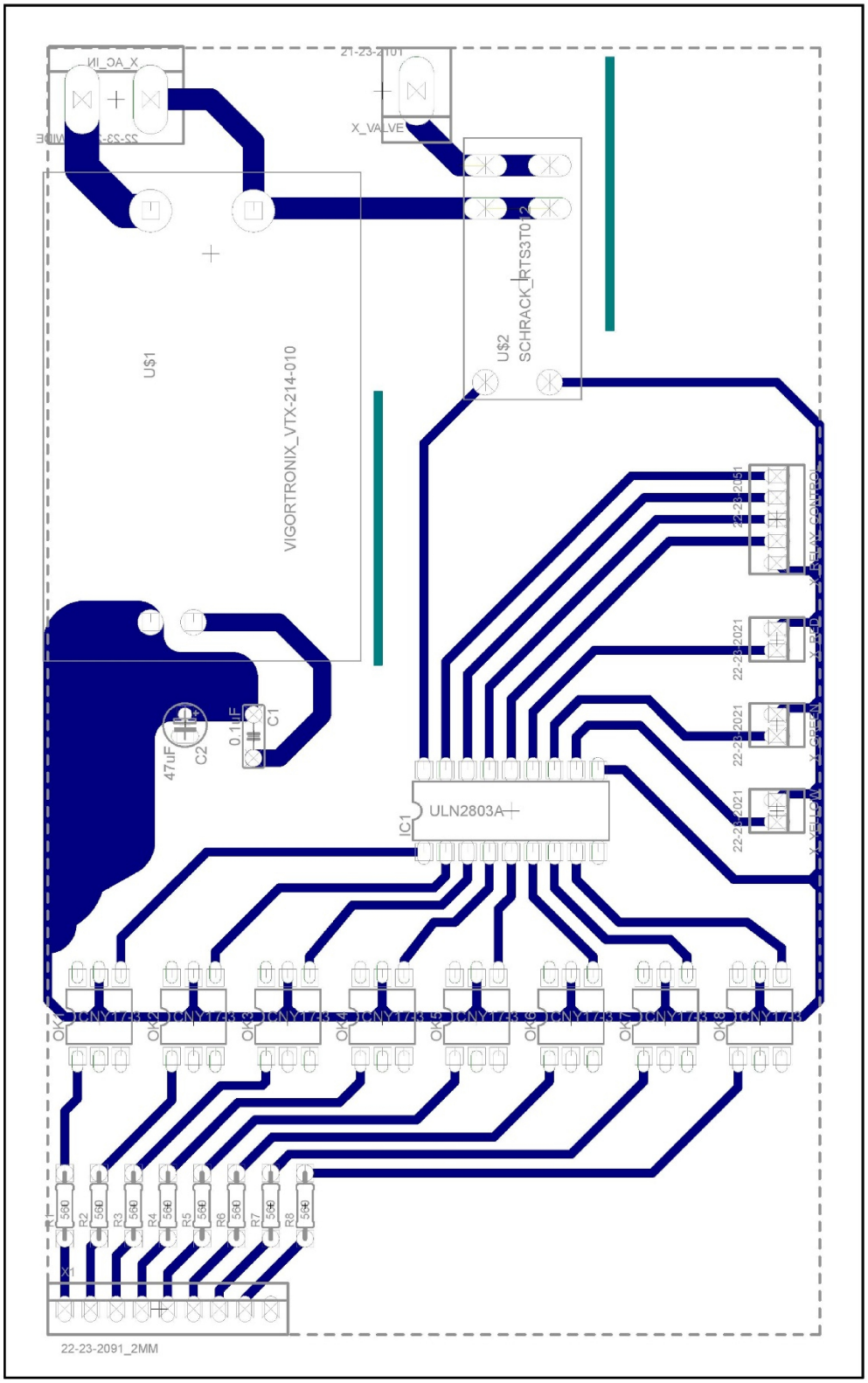
L2. Juhtimisosa elektriskeem



L3. Kõrgepingereleede osa elektriskeem

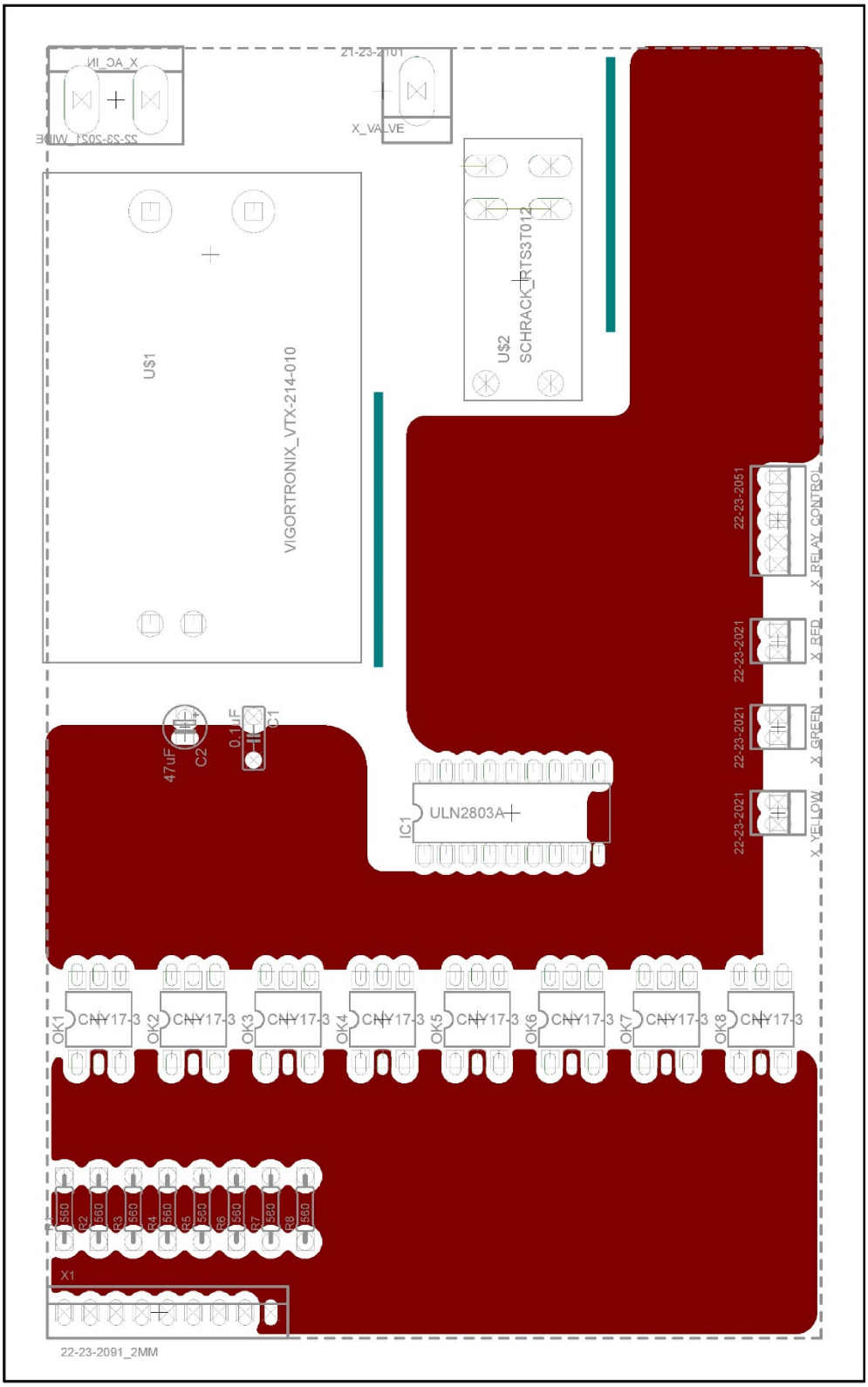


L4. Juhtimisosa PCB laotus (alumine pool)



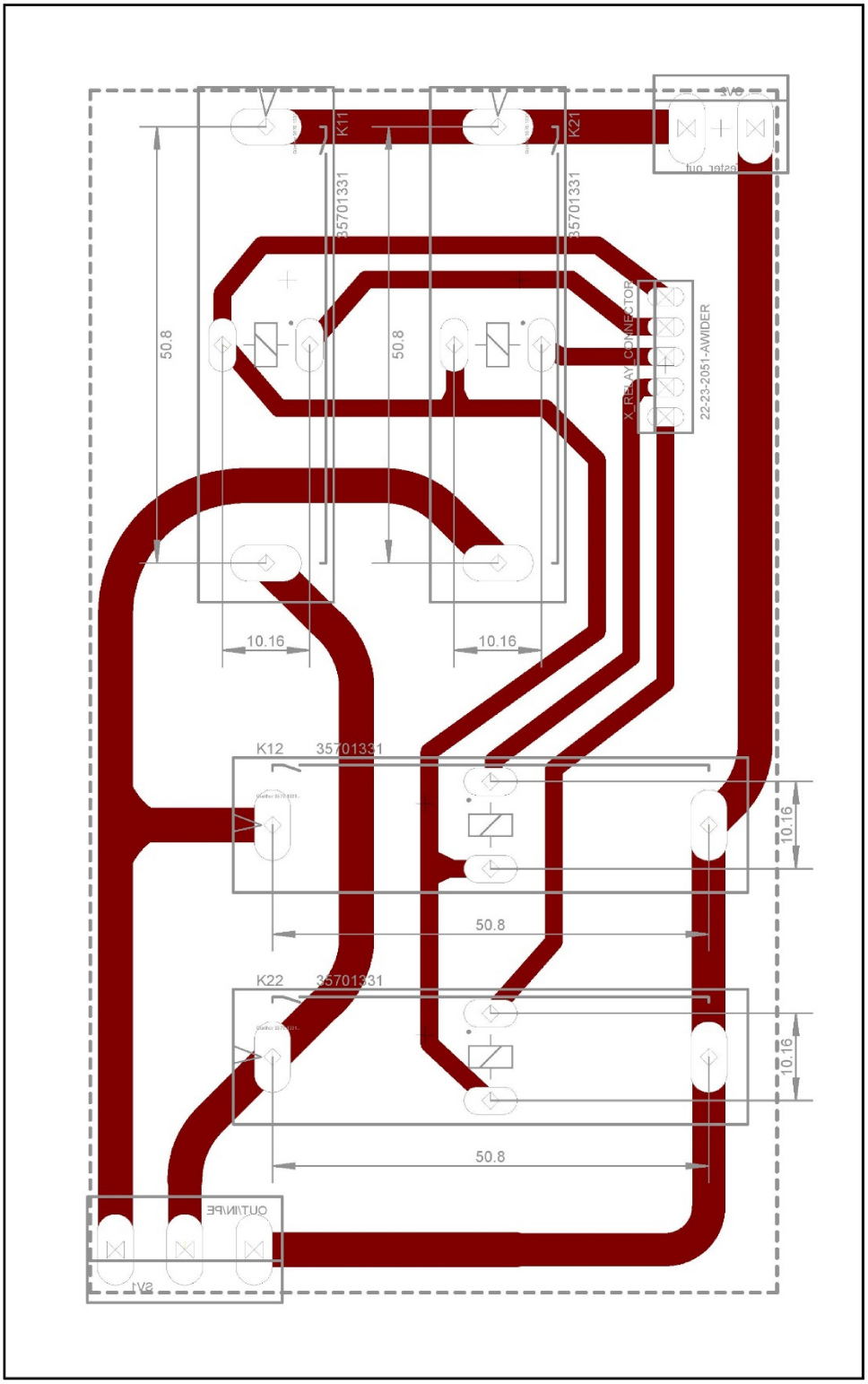
5.11.2014 22:27:05 f=1.75 E:\9Backups\Dropbox\Loputoo\schematics\relay_6t2.brd

L5. Juhtimisosa PCB laotus (ülemine pool)



5.11.2014 22:27:52 f=1.75 E:\9Backups\Dropbox\Loputoo\schematics\relay_6t2.brd

L6. Kõrgepingereleede osa PCB laotus



5.11.2014 22:22:26 f=1.75 E:\9Backups\Dropbox\Loputoo\schematics\relay_3_relays-2t.brd

L7. Programmide koodid (programmi failid)

L7.1. Programm testri juhtimiseks

Välja toon kõik '.cs' failid:

Programmi fail 1: Program.cs

```
using SefelecTester.Properties;
using System;
using System.Windows.Forms;

namespace SefelecTester
{
    internal static class Program
    {
        private static MainWin _myForm;

        /// <summary>
        ///     The main entry point for the application.
        /// </summary>
```

```
        [STAThread] private static void Main()
        {
            AppDomain.CurrentDomain.UnhandledException +=
CurrentDomain_UnhandledException;
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            _myForm = new MainWin();
            Application.Run(_myForm);
        }

        private static void CurrentDomain_UnhandledException
            (object sender, UnhandledExceptionEventArgs e)
        {
            try
            {
                var ex = (Exception)e.ExceptionObject;
                var tempor = new MessageBoxOk(Resources.Whoops + ". " +
Resources.Please_contact_the_developers_with_the_following + ":\n" +
Resources.Unhandeled_exception + ":\n" + ex.Message + ex.StackTrace,
Resources.Unhandeled_exception + ". " + Resources.Fatal_error + ".");
                tempor.ShowDialog();
                _myForm.WriteDebug();
            }
            finally{
                Application.Exit();}
        }
    }
}
```

Programmi fail 2: MainWin.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Globalization;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Text;
using System.Threading;
```

```
using System.Timers;
using System.Windows.Forms;
using SefelecTester.Properties;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public partial class MainWin : Form
    {
        private const string Line1 = "ACW;blank;0.1;50;0;42;0;1;1;OFF;0;ON;only
debug test";//parameter strings to default the program
        private const string Line2 = "DCW;blank;0.1;#;0;11;0;1;1;OFF;0;ON;only
debug test";
        private const string Line3 = "IR;blank;0.1;#;1;9999;0;1;1;OFF;#;OFF;only
debug test";
        private const string Line4 = "GB;blank;3;50;0;650;0;1;#;OFF;0;OFF;only
debug test";
        private const int ParamsMain = 12;
        private readonly ManualResetEvent _busy1 = new
ManualResetEvent(false);//events to reset backgroundworkers
        private readonly ManualResetEvent _busy2 = new ManualResetEvent(false);
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private readonly List<SavedTest> _listOfSavedTests = new
List<SavedTest>();//list of saved tests that can be loaded later
        private readonly Timer _myTimer = new Timer();//timer for test
        public string StartTime = null;
        private int _autoTestNo = 100;
        private int _autoTestNoMin = 99;
        private Cancelling _cancel;
        private int _controllerNo;
        private SavedTest _currentTest;
        private bool _elapsed;
        private bool _keyElapsed = true; //KEYPRESS EVENT
        private int _manualTestNo = 100;
        private int _manualTestNoMin = 99;
        private bool _pause;
        private TestList[] _programList;
        private string _scannedSerialNumber = "00000";
        private string _selectedProductPath;
        private string _serialCom;//Sefelec tester COM port
        private string _serialNumber = "00000";
```

```
        private List<string> _startedtest = new List<string>();
        private int _testNo;
        private TextWriter _writer; //for TextBoxWriter
        private bool _zerocheck;

        public MainWin()
        {
            InitializeComponent();
            Wri.Wri_m(this);
            FormClosing += Form1_FormClosing;
        }

        private void MainWin_Load(object sender, EventArgs e) //load to form
        {
            //KEYPRESS EVENT commands are used to ignore enter key from barcode
reader
            //look to BarCode.cs for more information
            Show(); //show main window
            _writer = new ListBoxWriter(DebugWin); // Instantiate the writer
            Console.SetOut(_writer); // Redirect the out Console stream
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            worker_program.ProgressChanged += worker_program_ProgressChanged;//to
report backgroundworker progress
            worker_program.RunWorkerCompleted +=
worker_program_RunWorkerCompleted;//runs when backgroundworker completed
            worker_test.ProgressChanged += worker_test_ProgressChanged;
            worker_test.RunWorkerCompleted += worker_test_RunWorkerCompleted;
            OutputTabs.SelectedIndex = 0;//choose first output tab
            if (Debugger.IsAttached){//advanced options when debugging
                OutputTabs.SelectedIndex = 2;//show debug window
                checkBox_advanced.Enabled = true;
                checkBox_advanced.Show();}//show advanced options
            Wri.ConWrite(Resources.Program_is_starting, 1);
            StartTime = GetDateTimetoFilename();
            button_setup_Click(null, null);
            _currentTest.TestIsProgrammed = false;
            _currentTest.AutoName = Resources._name;
            _currentTest.AutoTestN = 100;
            _currentTest.TestParams = null;
```

```
        }

        private void Form1_FormClosing(Object sender, FormClosingEventArgs e)
//close main form
        {
            Wri.ConWrite(Resources.Close_button_pressed);
            e.Cancel = true;//cancel backgroundworkers
            ExitProgram();
        }

        private void ExitProgram()
        {
            TreeNode[] node = treeView1.Nodes.Find(Resources.Write_log,
true);//find nodes with 'Write Log' name
            if (node.Any()){
                SetNodeImage(treeView1.Nodes[0].Nodes[2], 1);} //log node to in
progress
            CancelWorker1();//cancel both backgroundworkers
            CancelWorker2();
            try{
                worker_program.Dispose();//dispose of both backgroundworkers
                worker_test.Dispose();}
            catch (Exception ex){
                Wri.ConWrite(Resources.Exception_raised + " (" + ex.Message + "):
" + ex);
                WriteDebugAfterEx();}
            try{
                if (serialPort1.IsOpen)
                    serialPort1.Close();}//close serial port
            catch (IOException){}
            CloseForms();
            Wri.ConWrite(Resources.Writing_logs);
            WriteLogCsv("TESTING FINISHED");
            Wri.ConWrite(Resources.Program_Exit, 1);
            WriteDebug();
            if (node.Any()){
                SetNodeImage(treeView1.Nodes[0].Nodes[2], 3);}//log node to
completed
            this.SynchronizedInvoke(Dispose);//dispose of window
            Application.Exit();//exit application
        }
```

```
        private void CloseForms()
        {
            for (int i = Application.OpenForms.Count - 1; i >= 0; i--)//find all
open windows
            {
                if (Application.OpenForms[i].Name == "MainWin") continue;//close
all windows, except main
                if (Application.OpenForms[i].InvokeRequired){
                    int i1 = i;
                    DebugWin.Invoke(new MethodInvoker(() =>
Application.OpenForms[i1].Close()));}
                else Application.OpenForms[i].Close();
            }
        }

        private bool OpenPopup()
        {
            while (worker_program.IsBusy) CancelWorker1();
            if (serialPort1.IsOpen) serialPort1.Close();//close serial port
            var popup = new StartupSettings(); //start StartupSettings form
            DialogResult dialogresult = popup.ShowDialog();
            if (dialogresult == DialogResult.OK) //get dialogue result from popup
            {
                label2.Text = Resources.Programming_started;
                _controllerNo = popup.ControllerNumber;//get controller number
form StartupSettings window
                Wri.ConWrite(Resources.Controller_number_is + ": " +
_controllerNo, 1);
                _serialCom = popup.SelectedCom;//get serial port number form
StartupSettings window
                _selectedProductPath = popup.SelectedProductPath;//get parameters
file path form StartupSettings window
                _zerocheck = popup.Zerocheck;//get zerocheck option form
StartupSettings window
                int controller = popup.ControllerNumber;
                _startedtest = new List<string> {"STARTED TESTING;TESTER ID:;" +
controller};
                string[] toPass = {popup.SelectedCom,
popup.SelectedProductPath};//string to send to backgroundworker
                try{
                    worker_program.RunWorkerAsync(toPass);} //this will run all
Transmitting protocol coding at background thread
```

```
                catch (Exception ex)
                {
                    Wri.ConWrite(Resources.Problem_starting + " <worker_program>"
+ ". " + Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                    WriteDebugAfterEx();
                    var tempor = new MessageBoxOk(Resources.Problem_starting + "
<worker_program>", Resources._0ERROR);
                    tempor.ShowDialog();
                }
                popup.Dispose(); //close popup form
            }
            else if (dialogresult == DialogResult.Cancel){
                return false;}
            return true;
        }

        private void SetupTesting()
        {
            _myTimer.Elapsed += OnTimer;
            _myTimer.Interval = 1000;
            _myTimer.Enabled = true;
            _myTimer.Elapsed += OnTimer;
        }

        private void StartTesting()
        {
            while (worker_test.IsBusy) CancelWorker2();//if worker if busy
            Wri.ConWrite(Resources.Started_test_no + ": " + _testNo++);
            foreach (TreeNode thisNode in
ReturnChildNodes(treeView1.Nodes[0].Nodes[1]))
                SetNodeImage(thisNode, 0);//set all treeview nodes to start
position
            SetImage(0);
            try{
                worker_test.RunWorkerAsync();} //this will run all Transmitting
protocol coding at background thread
            catch (Exception ex)
            {
                Wri.ConWrite(Resources.Problem_starting + " <worker_test>" + ". "
+ Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                WriteDebugAfterEx();
```

```
                var tempor = new MessageBoxOk(Resources.Problem_starting + "
<worker_test>", Resources._0ERROR);
                tempor.ShowDialog();
            }
        }

        public bool OpenComPort(string com) //serial port stuff
        {
            if (!serialPort1.IsOpen)
            {//serial port not open already
                try{
                    serialPort1.PortName = com;//take serial port name from
StartupSettings window
                    serialPort1.BaudRate = 115200;
                    serialPort1.Open();}//open serial port
                catch (Exception ex){
                    Wri.ConWrite(Resources.Problem_opening_COM_port + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                    WriteDebugAfterEx();
                    var tempor = new MessageBoxOk(Resources.APPLICATION_WILL_EXIT
+ ". " + Resources._0ERROR + ": " + ex, Resources.APPLICATION_WILL_EXIT + ". " +
Resources._0ERROR + " (" + Resources.with_COM_port + ")" + "");
                    tempor.ShowDialog();
                    ExitProgram();}
            }
            else{
                Wri.ConWrite(Resources.Serial_port_already_open);}
            if (serialPort1.IsOpen){//serial port already open
                Wri.ConWrite(Resources.Serial_port + " " + com + " " +
Resources._connected, 1);
                return true;}
            Wri.ConWrite(Resources.Cannot_connect_to_serial_port + ": " + com, 2);
            return false;
        }

        private void buttonTemp_Click(object sender, EventArgs e)
        {
            Wri.ConWrite(Resources.New_instance_of_testing_started);
            StartTesting();//start testing procedure
        }
```

```
        private void RemoveEmptyLines(string path1, string path2)
        {//for removing empty lines from parameters file
            File.WriteAllText(@path2, string.Empty); //clear file contents
            bool newF = false;
            int lineN = 0;
            try
            {
                using (var streamReader = new StreamReader(@path1))
                using (var streamWriter = new StreamWriter(@path2))
                {
                    string line;
                    while ((line = streamReader.ReadLine()) != null)//while there
is content in line (even if just newline)
                    {
                        if (string.IsNullOrWhiteSpace(line) ||
string.IsNullOrWhiteSpace(line.Split(';')[0])) continue;//if line empty or first
parameter empty then do nothing
                        if (newF)
                            streamWriter.Write("\n");//if has content then write
to new file
                        else
                            newF = true;
                        if (lineN == 0 || line.Split(';')[0] == "end") //if first
line(name) or end
                        {
                            streamWriter.Write(line.Split(';')[0]); //write only
first word
                            if (line.Split(';')[0] == "end"){
                                break;}
                        }
                        else{
                            streamWriter.Write(line);} //writes without newline
character
                        lineN++;
                    }
                }
            }
            catch (Exception ex)
            {
                string exc;
                if (ex is FileNotFoundException)//no such file, wrong path
                    exc = Resources.The_file_is_not_found;
```

```
                else if (ex is IOException)//can't access file
                    exc = Resources.The_file_cannot_be_accessed;
                else if (ex is FileFormatException || ex is
FileLoadException)//can't load file
                    exc = Resources.The_file_cannot_be_loaded;
                else
                    exc = Resources.Some_other_error;
                Wri.ConWrite(Resources.Remove_empty_lines + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                var errorEm = new
MessageBoxAlt(Resources.Error_removing_empty_lines_from_the_tests_file,
Resources.There_was_a_problem_removing_empty_lines_from_the_tests_file + ":\n" +
path1 + "\n" + path2 + "\n" + exc + "\n" +
Resources.Press__yes__if_the_problem_is_resolved_to_try_again + ".\n" +
Resources.Press__no__to_exit_the_program + "\n" + ex, true, false);
                DialogResult exit = errorEm.ShowDialog();
                WriteDebugAfterEx();
                if (exit == DialogResult.Yes){
                    RemoveEmptyLines(path1, path2);}//try again
                else{
                    ExitProgram();}
            }
        }

        public void WriteDebug()
        {
            var list = new List<Listboxes>();//list of listboxes
            string path = Directory.GetCurrentDirectory();
            path += @"\debug_info";
            if (!Directory.Exists(path)){//if no directory
                Directory.CreateDirectory(path);}//then create it
            list.Add(new Listboxes {File = path + @"\debug" + StartTime + ".txt",
Box = DebugWin});//add all lsitboxes to list
            list.Add(new Listboxes {File = path + @"\error" + StartTime + ".txt",
Box = ErrorConsole});
            list.Add(new Listboxes {File = path + @"\output" + StartTime + ".txt",
Box = OutputConsole});
            foreach (Listboxes outP in list)//do for every item in list (every
listbox)
            {
                Wri.ConWrite(Resources.Write + ": " + outP.File);
                try
                {
```

```
                    using (var streamWriter = new StreamWriter(@outP.File))
                    {
                        foreach (object item in outP.Box.Items){//do for each line
in listbox
                                streamWriter.Write(item.ToString());}//add to file
                    }
                }
                catch (Exception ex)
                {
                    string exc;
                    if (ex is FileNotFoundException)
                        exc = Resources.The_file_is_not_found;
                    else if (ex is IOException)
                        exc = Resources.The_file_cannot_be_accessed;
                    else if (ex is FileFormatException || ex is FileLoadException)
                        exc = Resources.The_file_cannot_be_loaded;
                    else
                        exc = Resources.Some_other_error;
                    Wri.ConWrite(Resources.Writing_debug + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                    WriteDebugAfterEx();
                    var tempor = new
MessageBoxOk(Resources.There_was_a_problem_writing_debug_file + ":\n" + outP.File
+ "\n" + Resources.File_will_not_be_written + "\n" + exc + "\n" + ex,
Resources.Warning);
                    tempor.ShowDialog();
                }
            }
        }


        public void WriteDebugAfterEx()
        {//alternative debug file
            string path = Directory.GetCurrentDirectory();
            path += @"\debug_info";
            if (!Directory.Exists(path)){//if directory doesn't exist
                Directory.CreateDirectory(path);}//create it
            path += @"\debug_Ex" + StartTime + ".txt";//add time program was
started to file
            Wri.ConWrite(Resources.Write + ": " + path);
            File.WriteAllText(@path, string.Empty); //clear file contents
            try
```

```
            {
                using (var streamWriter = new StreamWriter(@path))
                {
                    foreach (object item in DebugWin.Items){//all lines in debug
listbox
                        streamWriter.Write(item.ToString());}//write to line
                }
            }
            catch (Exception ex)
            {
                string exc;
                if (ex is FileNotFoundException)
                    exc = Resources.The_file_is_not_found;
                else if (ex is IOException)
                    exc = Resources.The_file_cannot_be_accessed;
                else if (ex is FileFormatException || ex is FileLoadException)
                    exc = Resources.The_file_cannot_be_loaded;
                else
                    exc = Resources.Some_other_error;
                Wri.ConWrite(Resources.Write + " DebugEx" + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                WriteDebugAfterEx();
                var tempor = new
MessageBoxOk(Resources.There_was_a_problem_writing_debug_file + ":\n" + path +
"\n" + Resources.File_will_not_be_written + "\n" + exc + "\n" + ex,
Resources.Warning);
                tempor.ShowDialog();
            }
        }


        private void OnTimer(Object source, ElapsedEventArgs e) { _elapsed = true;
}


        private static string IntTo3String(int no)
        {//convert int to string, used to send to tester
            string str = no.ToString();
            while (str.Length < 3){
                str = "0" + str;}
            return str;
        }
```

```
        private static bool CancelPending(BackgroundWorker worker)
        {//chacks if backgroundworker has pending cancel
            if (!worker.CancellationPending) return false;
            Wri.ConWrite(Resources.Cancel_pending + ": " + worker);
            return true;
        }

        private void WriteLog(string text)
        {//write string to log file
            string filePath = Directory.GetCurrentDirectory();
            filePath += @"\Testing_Log.txt";
            try
            {
                using (StreamWriter streamWriter = File.AppendText(@filePath)){
                    streamWriter.WriteLine(text.Replace(Environment.NewLine,
"""));}//append to file
            }
            catch (Exception ex)
            {
                string exc;
                if (ex is FileNotFoundException)
                    exc = Resources.The_file_is_not_found;
                else if (ex is IOException)
                    exc = Resources.The_file_cannot_be_accessed;
                else if (ex is FileFormatException || ex is FileLoadException)
                    exc = Resources.The_file_cannot_be_loaded;
                else
                    exc = Resources.Some_other_error;
                Wri.ConWrite(Resources.Writing_log + ". " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);
                var errorEm = new MessageBoxAlt(Resources.Error_writing_the_log,
Resources.There_was_a_problem_writing_the_file + ":\n" + filePath + "\n" + exc +
"\n" + Resources.Press__yes__if_the_problem_is_resolved_to_try_again + ".\n" +
Resources.Press__no__to_exit_the_program + "\n" + ex, true, false);
                DialogResult exit = errorEm.ShowDialog();
                WriteDebugAfterEx();
                if (exit == DialogResult.Yes){
                    WriteLog(text);}//try again
                else{
                    ExitProgram();}
            }
```

```
        }

        private void WriteLogCsv(string text)
        {//write log as csv as well as txt
            string filePath = Directory.GetCurrentDirectory();
            filePath += @"\Testing_Log.csv";
            string text2 = text.Replace(';', '\t');//in txt > use tab instead of ;
            text2 += ("\t" + GetDateTime());//add time to line
            WriteLog(text2);//write to txt
            try
            {
                using (StreamWriter streamWriter = File.AppendText(@filePath)){
                    streamWriter.WriteLine(text.Replace(Environment.NewLine, "") +
";" + GetDateTime());}//append to csv and add time
            }
            catch (Exception ex)
            {
                string exc;
                if (ex is FileNotFoundException)
                    exc = Resources.The_file_is_not_found;
                else if (ex is IOException)
                    exc = Resources.The_file_cannot_be_accessed;
                else if (ex is FileFormatException || ex is FileLoadException)
                    exc = Resources.The_file_cannot_be_loaded;
                else
                    exc = Resources.Some_other_error;
                Wri.ConWrite(Resources.Writing_log + ". " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);
                var errorEm = new MessageBoxAlt(Resources.Error_writing_the_log,
Resources.There_was_a_problem_writing_the_file + ":\n" + filePath + "\n" + exc +
"\n" + Resources.Press__yes__if_the_problem_is_resolved_to_try_again + ".\n" +
Resources.Press__no__to_exit_the_program + "\n" + ex, true, false);
                DialogResult exit = errorEm.ShowDialog();
                WriteDebugAfterEx();
                if (exit == DialogResult.Yes){
                    WriteLogCsv(text);}//try again
                else{
                    ExitProgram();}
            }
        }
```

```
        private static string GetDateTimetoFilename(){//get time and modify format
            return ("(" + DateTime.Now + ")").Replace(" ", "_").Replace(".", "-
").Replace(":", "-");}

        private static string GetDateTime(){//get time
            return ("(" + DateTime.Now + ")");}

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {//change parameters if option is changed
            if (checkBox1.Checked){
                SerialNo.Text = _scannedSerialNumber;
                checkBox1.Text = Resources.Scanned_serial_number;}
            else{
                SerialNo.Text = _serialNumber;
                checkBox1.Text = Resources.Custom_serial_number + " (" +
Resources._change_if_needed + ")";}
        }

        private void SetNodeImage(TreeNode node, int imag)
        {
            try{//use invoke because action sent from background thread
                Invoke(new MethodInvoker(() => node.ImageIndex = imag)); //testing
node image to inprogress
                Invoke(new MethodInvoker(() => node.SelectedImageIndex = imag));}
            catch (Exception ex)
            {
                if (ex is ArgumentOutOfRangeException)
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message +
"): " + ex);
                else if (ex is IndexOutOfRangeException)
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message +
"): " + ex);
                else if (ex is InvalidOperationException){
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message +
"): " + ex);
                    Wri.ConWrite(Resources.Invalid_operation_exception + ". " +
Resources.Can_happen_at_program_exit + ".");}
                WriteDebugAfterEx();
            }
        }

        private void SetImage(int imag)
```

```
        {
            try{
                Invoke(new MethodInvoker(() => pictureBox1.Image =
imageList1.Images[imag]));}//set iamge showing testing state


            catch (Exception ex)
            {
                if (ex is ArgumentOutOfRangeException)
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message +
"): " + ex);
                else if (ex is IndexOutOfRangeException)
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message +
"): " + ex);
                else if (ex is InvalidOperationException){
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message +
"): " + ex);
                    Wri.ConWrite(Resources.Invalid_operation_exception + ". " +
Resources.Can_happen_at_program_exit + ".");}
                WriteDebugAfterEx();
            }
        }


        private static List<TreeNode> ReturnChildNodes(TreeNode parentNode)
        {//get all child nodes
            var nodes = new List<TreeNode> {parentNode};
            if (parentNode.GetNodeCount(false) <= 0) return nodes;//no more child
nodes
            for (int i = 0; i < parentNode.GetNodeCount(false); i++)
            {
                nodes.Add(parentNode.Nodes[i]);
                if (parentNode.Nodes[i].GetNodeCount(false) <= 0) continue;
                List<TreeNode> temp = ReturnChildNodes(parentNode.Nodes[i]);//get
child nodes of all nodes
                temp.ForEach(nodes.Add);
            }
            return nodes;//return all childnodes as list
        }


        private void SerialNo_ValueChanged(object sender, EventArgs e)
        {//checkbox changed
            if (checkBox1.Checked){
                _serialNumber = SerialNo.Text;
```

```
                SerialNo.Text = _scannedSerialNumber;}
            else{
                _scannedSerialNumber = SerialNo.Text;
                SerialNo.Text = _serialNumber;}
        }

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed)
            {
                if (msg.WParam.ToInt32() == (int) Keys.Enter){
                    Wri.ConWrite(Resources.Prevented_enter_key_from_firing);
                    return true;}
            }
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT

        private void button_setup_Click(object sender, EventArgs e)
        {
            buttonTemp.Enabled = false;
            if (OpenPopup()){ //opens startup popup window
                SetupTesting();}//start testing
        }

        private void button_cancel_Click(object sender, EventArgs e)
        {
            SendData("FUNC:TEST OFF");//test off command
            int a1 = CancelWorker1();
            int a2 = CancelWorker2();
            Wri.ConWrite("<worker_program> " + Resources._cancelled + ", " +
Resources._return + ": " + a1 + ", " + "<worker_test> " + Resources._cancelled +
", " + Resources._return + ": " + a2);
            if (a1 != 1 && a2 != 1) return;//both backgroundworkers are stopped
```

```
            Wri.ConWrite(Resources.Open + " <cancelling> " + Resources._window);
            _cancel = new Cancelling();//start cancelling window until
backgroundworkers are stopped
            _cancel.Show();
        }

        private void StopTest()
        {
            Wri.ConWrite(Resources.StopTest_started);
            if (SendDataAndReadEncased("FUNC:TEST OFF", "FUNC:TEST?", "TEST OFF"))
return;//if test is off then exit
            Wri.ConWrite(Resources.Test_on + ". " + "<TurnOff> " +
Resources._instance_starting);
            var turningOff = new TurnOff(this, Resources.Turning_off_the_test +
"\n" +
Resources.If_a_test_is_not_running_then_close_this_window_and_check_connection_to_
the_tester + "\n" +
Resources.If_the_test_is_running_then_stop_it_manually_and_close_this_window,
Resources.Turning_off_test);
            turningOff.ShowDialog();//show window until turning off the test
        }

        public void LogToOutput(string text) //logging to listboxes
        {//write to output listbox
            this.SynchronizedInvoke(() =>{
                OutputConsole.Items.Add(text);//add to listbox
                OutputConsole.SelectedIndex = OutputConsole.Items.Count -
1;//activate last index, scrolls down
                OutputConsole.SelectedIndex = -1;});
        }

        public void LogToError(string text)
        {//write to error listbox
            this.SynchronizedInvoke(() =>{
                ErrorConsole.Items.Add(text);
                ErrorConsole.SelectedIndex = ErrorConsole.Items.Count - 1;
                ErrorConsole.SelectedIndex = -1;});
        } //--logging to listboxes

        private void SetElements(bool state)
        {//set all elements to selected state
            this.SynchronizedInvoke(() =>{
```

```
                button_setup.Enabled = state;
                buttonTemp.Enabled = state;
                checkBox1.Enabled = state;
                SerialNo.Enabled = state;
                button_loadset.Enabled = state;
                button_saveset.Enabled = state;});//used to disable buttons and
other inouts while doing work
        }

        private void button_saveset_Click(object sender, EventArgs e)
        {
            MessageBoxOk mb;
            if (_currentTest.TestIsProgrammed == false){//if tester is not yet
programmed
                mb =new MessageBoxOk(Resources.No_program_settings_can_be_saved +
"\n" + Resources.Program_the_tester_before_saving, Resources.Nothing_to_save);
                mb.ShowDialog();
                return;}//exit
            _currentTest.StartedTest = _startedtest.ToArray();//save test
parameters
            _autoTestNoMin--;//Set program indexes to free places for next program
            _manualTestNoMin = _manualTestNo--;//Set program indexes to free
places for next program
            _listOfSavedTests.Add(_currentTest);//add to tests list
            Wri.ConWrite(Resources.Test_saved + ": " + _currentTest.AutoName + "
(" + _currentTest.AutoTestN + ") " + Resources.with_tests + ":", 1);
            string parameters = null;
            foreach (TestList param in _currentTest.TestParams)
            {//show all parameters
                parameters += "\n";
                parameters += param.TestParameters;
                Wri.ConWrite("\t" + Resources.Name + ": " + param.TestName + " ("
+ param.TestNo + "), " + Resources._test + ":" + param.TestParameters, 1);
            }
            mb =new MessageBoxOk(Resources.Test_is_saved + "\n" + Resources.Test +
" (" + _currentTest.AutoTestN + ") " + Resources._name + ": " +
_currentTest.AutoName + "\n" + Resources.Test_parameters + ": " + parameters,
Resources.Test_saved);
            mb.ShowDialog();
        }

        private void button_loadset_Click(object sender, EventArgs e)
        {
```

```
            var load = new LoadProgram(_listOfSavedTests);
            DialogResult dial = load.ShowDialog();//show window for loading saved
test
            if (dial != DialogResult.OK) return;//exit
            _currentTest = load.SelTest;//else load selected test
            Wri.ConWrite(Resources.Test_loaded + ": " + _currentTest.AutoName + "
(" + _currentTest.AutoTestN + ") " + Resources.with_tests + ":", 1);
            Wri.ConWrite(_currentTest.TestParams.Count() + Resources._list + ": "
+ _currentTest.TestParams);
            _autoTestNo = _currentTest.AutoTestN;
            _programList = _currentTest.TestParams;
            var modTree = new List<string>();//list to modify treeview
            foreach (TestList prog in _programList)
            {//all programs to list
                modTree.Add(prog.TestName);
                Wri.ConWrite("\t" + Resources.Name + ": " + prog.TestName + " (" +
prog.TestNo + "), " + Resources._test + ":" + prog.TestParameters, 1);
            }
            this.SynchronizedInvoke(() => treeView1.Nodes.Clear());//clear
treeview
            FillTreeView(modTree.ToArray());//list to treeview
            this.SynchronizedInvoke(() =>
            {
                treeView1.ExpandAll();//expand treeview
                treeView1.Nodes[0].Text = _currentTest.AutoName;
            });
            SetNodeImage(treeView1.Nodes[0].Nodes[0], 3);
            foreach (TreeNode nn in treeView1.Nodes[0].Nodes[0].Nodes){
                SetNodeImage(nn, 3);}
            _serialNumber = SerialNo.Text = "00000";
            foreach (string ll in _currentTest.StartedTest) WriteLogCsv(ll);
        }

        private void checkBox_advanced_CheckedChanged(object sender, EventArgs
e){//advanced option changed
            _pause = checkBox_advanced.Checked;}//change if pause is executed

        #region Programming Device

        private void worker_program_DoWork(object sender, DoWorkEventArgs e) //
PROGRAMMING DEVICE
        {
```

```
            this.SynchronizedInvoke(() => _serialNumber = SerialNo.Text =
"00000");
            _manualTestNo = _manualTestNoMin;
            _autoTestNo = _autoTestNoMin;
            _programList = null;
            _currentTest.TestIsProgrammed = false;
            SetElements(false);
            StartWorker(worker_program);
            this.SynchronizedInvoke(() => treeView1.Nodes.Clear());
            bool check = true;
            var sent = (string[]) e.Argument;
            bool state;
            string autoName = null;
            var toTest = new List<TestList>();
            var toTree = new List<string>();
            if (sent.Count() > 1)//if parameters not sent then skip testing
            {
                const int de = 2;
                if (!OpenComPort(sent[0])){//if com port can't be opened
                    Wri.ConWrite(Resources.Cannot_acces_COM_port_for_programming +
". " + Resources.Exiting + ".", 2);
                    CancelWorker1();//cancel backgroundworker
                    return;}
                string newPath = Path.GetDirectoryName(sent[1]);
                newPath += @"\temp\";//save new file to subdirectory
                if (!Directory.Exists(newPath)){//create dir if not existing
                    Directory.CreateDirectory(newPath);}
                newPath += Path.GetFileName(sent[1]);
                newPath = newPath.Replace(".csv",
"_removed_empty_lines.csv");//append to filename
                RemoveEmptyLines(_selectedProductPath, newPath);
                _selectedProductPath = newPath; //empty lines removed
                string readFile = ReadFile(_selectedProductPath);//whole file to
string
                string[] readFile2Split = readFile.Split('\n');//split to lines
                autoName = readFile2Split[0].Split(';')[0]; //get AUTO program
name
                int max = readFile2Split.Count();//count lines
                string tests = null;
                var testLines = new List<string> {autoName};
```

```
                foreach (string line in readFile2Split.Where(line =>
line.Split(';').Count() >= ParamsMain)){//add tests to list
                    toTree.Add(line.Split(';')[1]);//test name to add to treeview
                    testLines.Add(line);
                    tests += line;
                    tests += "\n";}
                FillTreeView(toTree.ToArray());
                try{
                    this.SynchronizedInvoke(() => treeView1.Nodes[0].Text =
autoName);}//change treeview first node name
                catch (Exception ex){
                    Wri.ConWrite(Resources.Changing_main_node_name + ". " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);
                    var tempor = new MessageBoxOk(Resources.Exeption_raised + " ("
+ Resources._changing_main_node_name + "): " + ex, Resources._0ERROR);
                    tempor.ShowDialog();}
                foreach (TreeNode thisNode in
ReturnChildNodes(treeView1.Nodes[0])){
                    SetNodeImage(thisNode, 0);}
                Invoke(new MethodInvoker(() => treeView1.ExpandAll()));
                var testsList = new MessageBoxAlt(Resources.List_of_tests,
Resources.List_of_tests_in_the_file_is + ":\n" + tests + "\n" +
Resources.Press__OK__if_this_is_correct + ".\n" +
Resources.Press__Cancel__to_exit_the_program_and_correct_mistakes, true, true);
                DialogResult dialogRes = testsList.ShowDialog();//asks if tests in
the list are ok
                if (dialogRes == DialogResult.No){//if not then exit
                    ExitProgram();}
                ZeroReference refer = _zerocheck ? CheckForGB(max, readFile2Split)
: new ZeroReference {Referen = false, Reference = "0"};//if zerocheck is set to
true then do it
                worker_program.ReportProgress(100/(max + de));//report progress
                Thread.Sleep(15);//wait
                SendData("FUNC:TEST OFF");//to tester: test off, just in case its
running
                if (!SendDataAndReadEncased("MAIN:FUNC AUTO", "MAIN:FUNC?", "AUTO
MODE")){//change mode to auto and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto mode", 2);}
                if (!SendDataAndReadEncased("AUTO:STEP " + _autoTestNo,
"AUTO:STEP?", IntTo3String(_autoTestNo))){//change auto test number and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto step", 2);}
                string delErr;
```

```
                do
                {//delete test list from automatic test
                    _busy1.WaitOne();
                    Thread.Sleep(50);
                    SendData("AUTO:PAGE:DEL 1");
                    delErr = GetError()[0];
                } while (delErr == "0");//do while no error deleting
                SendData("*CLS\n"); //deleting a test that is not there gives a
error
                if (!SendDataAndReadEncased("MAIN:FUNC MANU", "MAIN:FUNC?", "MANU
MODE")){//change mode to manual and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " manu mode", 2);}
                worker_program.ReportProgress(200/(max + de));
                Thread.Sleep(15);
                e.Cancel = CancelPending(worker_program);
                if (e.Cancel){//check if cancel request is active
                    state = false;
                    return;}
                int lineN = 0;
                SetNodeImage(treeView1.Nodes[0].Nodes[0], 2);//treeview
programming node to in progress
                foreach (string line in testLines)
                {
                    if (lineN == 0){
                        lineN++;
                        continue;}//skip first line
                    if (line.Trim() == "end"){//if end is before last line
                        Wri.ConWrite(Resources.End_of_file_detected);
                        break;}
                    if (line.Split(';').Count() < ParamsMain){//if not enough
parameters
                        Wri.ConWrite(Resources.Line_has_wrong_length + ": " +
line);
                        continue;}//skip line
                    _busy1.WaitOne();//pause request check
                    e.Cancel = CancelPending(worker_program);
                    if (e.Cancel){
                        state = false;
                        return;}
```

```
                    worker_program.ReportProgress((int) ((lineN + de -
0.5)*100/(max + de))); //reports a percentage between 0 and 100
                    Thread.Sleep(15);
                    try
                    {
                        WriteDebugAfterEx();
                        SetNodeImage(treeView1.Nodes[0].Nodes[0].Nodes[lineN - 1],
2); //set to inprogress (manual test)
                        if (!SendDataAndReadEncased("MANU:STEP " +
IntTo3String(_manualTestNo - lineN), "MANU:STEP?", IntTo3String(_manualTestNo -
lineN))){//change manual program number and check
                            check = false;
                            Wri.ConWrite(Resources.Error_setting + " manu step",
2);}
                        int testTime;
                        int success = ProgramManual(line, refer.Reference,
refer.Referen, out testTime);//program manual test according to 1 line
                        if (success == 0)
                        {
                            Wri.ConWrite(Resources._0ERROR + ". " +
Resources.Couldn_t_program_no_ + (_manualTestNo - lineN) + "::" + line + ". " +
Resources.Tring_again + ".");
                            switch (line.Split(';')[0])
                            {//program with default program
                                case "ACW":
                                    ProgramManual(Line1, refer.Reference, false,
out testTime);
                                    break;
                                case "DCW":
                                    ProgramManual(Line2, refer.Reference, false,
out testTime);
                                    break;
                                case "IR":
                                    ProgramManual(Line3, refer.Reference, false,
out testTime);
                                    break;
                                case "GB":
                                    ProgramManual(Line4, refer.Reference, false,
out testTime);
                                    break;
                            }//this sets parameters to low value so that maximum
ratings of parameter combinations will not be exeeded
                            success = ProgramManual(line, refer.Reference,
refer.Referen, out testTime);//try programming again
```

```
                        if (success == 0)
                        {//programming failed
                            Wri.ConWrite(Resources._0ERROR + ". " +
Resources.Couldn_t_program_no_ + (_manualTestNo - lineN) + "::" + line + ". " +
Resources._0FAILED + ".", 2);
                            var mess = new
MessageBoxAlt(Resources.Wrong_program, Resources.The_program_parameters_are_wrong
+ ".\n" + Resources.Program + ":" + line + "\n" +
Resources.Would_you_like_to_skip_this_test_and_only_do_other_tests + "?\n" +
Resources.Choosing_no_will_exit_the_program + ".", true, false);
                            DialogResult dialogResult =
mess.ShowDialog();//ask if line should be removed and not programmed
                            PauseWorker(worker_program);
                            if (dialogResult == DialogResult.Yes)
                            {//remove program line

SetNodeImage(treeView1.Nodes[0].Nodes[0].Nodes[lineN - 1], 1);
                                try{
                                    Invoke(new MethodInvoker( () =>
treeView1.Nodes[0].Nodes[1].Nodes[lineN - 1].Remove()));}//remove node from
testing
                                catch (ArgumentOutOfRangeException ex){
                                    Wri.ConWrite(Resources.Exeption_raised +
"(" + ex.Message + "): " + ex);
                                    WriteDebugAfterEx();}
                                lineN++;
                                StartWorker(worker_program);
                                continue;
                            }
                            if (dialogResult == DialogResult.No)
ExitProgram();
                        }
                    }
                    else if (success == -1){
                        break;}
                    int testNumber = _manualTestNo - lineN;
                    string testParams = GetDataEncased("MANU" + testNumber +
":EDIT:SHOW?");//get test pparameters from tester
                    _startedtest.Add("PARAMETERS:;" + autoName + " (PRODUCT
NAME);" + line.Split(';')[1] +
                                     " (SUBTEST NAME);SUBTEST" + (99 -
testNumber) + ";" + testParams);
                    string instructions = line.Split(';').Count() > ParamsMain
                        ? line.Split(';')[ParamsMain]
```

```
                            : Resources._do_as_instructed;//if last parameter is
not added then add default
                        toTest.Add(new TestList{
                            TestNo = (_manualTestNo - lineN),
                            TestingTime = testTime,
                            TestParameters = testParams,
                            TestDescription = instructions,
                            TestName = line.Split(';')[1]});//add parameters to
list
                        SetNodeImage(treeView1.Nodes[0].Nodes[0].Nodes[lineN - 1],
3);//set treeview node to completed
                        Wri.ConWrite(Resources.Added_to_test_list + ": " +
(_manualTestNo - lineN));
                    }
                    catch (IndexOutOfRangeException ex){
                        Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message
+ "): " + ex, 2);
                        WriteDebugAfterEx();}
                    worker_program.ReportProgress((lineN + de)*100/(max + de));
//reports a percentage between 0 and 100
                    Thread.Sleep(15);
                    lineN++;
                }
                if (!SendDataAndReadEncased("MAIN:FUNC AUTO", "MAIN:FUNC?", "AUTO
MODE")){//set tester mode to auto and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto mode", 2);}
                if (!SendDataAndReadEncased("AUTO:STEP " + _autoTestNo,
"AUTO:STEP?", IntTo3String(_autoTestNo))){//set tester auto test number and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto step", 2);}
                if (!SendDataAndReadEncased("AUTO:NAME " + autoName, "AUTO:NAME?",
autoName)){//set tester auto test name and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto name", 2);}
                foreach (TestList manu in toTest){//add manual tests to automatic
test
                    Wri.ConWrite(Resources.Add_manual_test_to_auto_test + ": " +
manu.TestNo);
                    SendData("AUTO:EDIT:ADD " + manu.TestNo);}
                _manualTestNo = _manualTestNo - lineN;
                if (!check){
```

```
                    Wri.ConWrite(Resources.Some_error_occured_while_programming,
2);
                    var tempor = new
MessageBoxOk(Resources.Some_error_occured_while_programming, Resources._0ERROR);
                    tempor.ShowDialog();}
                SetNodeImage(treeView1.Nodes[0].Nodes[0], !check ? 1 : 3);//set
node image to completed or failed
                foreach (string ll in _startedtest) WriteLogCsv(ll);
                worker_program.ReportProgress(100);
                Thread.Sleep(15);
                state = true;
            }
            else{
                Wri.ConWrite(Resources.Error_with_index_in_programming + "
(<worker_program>)");
                state = false;}
            if (e.Cancel){
                state = false;}
            try{
                ChangeLastHold(toTest.Last().TestNo);}
            catch (InvalidOperationException ex){//no tests in toTest list
                Wri.ConWrite(Resources.List_of_tests_is_empty + ". " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);}
            _programList = toTest.ToArray();
            if (autoName != null)//save if autoname parameter is there
                _currentTest = new SavedTest{
                    TestIsProgrammed = true,
                    AutoTestN = _autoTestNo,
                    AutoName = autoName,
                    TestParams = toTest.ToArray()};
            e.Result = state;
        }


        private void worker_program_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            SetElements(true);//set all elements to usable
            if (worker_program.IsBusy){
                worker_program.CancelAsync();
                _busy1.Set();} // Unblock worker so it can see that
            if (e.Error != null){ // First, handle the case where an exception was
thrown
```

```
                Wri.ConWrite(Resources.Programming + " (<worker_program>) " +
Resources._completed_with_error + ": " + e.Error.Message, 2);
                label2.Text = Resources.Programming_ended_with_an_error;
                this.SynchronizedInvoke(() => buttonTemp.Enabled = false);}
            else if (e.Cancelled){//backgroundworker was cancelled
                Wri.ConWrite(Resources.Programming + " (<worker_program>) " +
Resources._cancelled, 3);
                label2.Text = Resources.Programming_canceled;
                buttonTemp.Enabled = false;}
            else{ // Finally, handle the case where the operation succeeded
                Wri.ConWrite(Resources.Programming + " (<worker_program>) " +
Resources._finished, 1);
                label2.Text = Resources.Programming_successful;
                buttonTemp.Enabled = true;}
            try{
                _cancel.Dispose();}//dispose of backgroundworker after finishing
            catch (NullReferenceException ex){
                Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message + "): "
+ ex);
                WriteDebugAfterEx();}
            catch (Exception ex){
                Wri.ConWrite(Resources.Disposing_of + " <cancelling> " +
Resources._window + ". " + Resources.Exeption_raised + "(" + ex.Message + "): " +
ex, 2);
                WriteDebugAfterEx();
                var tempor = new MessageBoxOk(Resources.Exeption_raised + "(" +
Resources._close_cancel_form + "): " + ex, Resources._0ERROR);
                tempor.ShowDialog();}
        }


        private void worker_program_ProgressChanged(object sender,
ProgressChangedEventArgs e){
            progressBar1.Value = e.ProgressPercentage;} //change progresbar


        private int CancelWorker1()
        {
            Wri.ConWrite(Resources.Cancel_request + " (<worker_program>)");
            try{
                if (!worker_program.IsBusy) return 2;
                worker_program.CancelAsync();
                Wri.ConWrite(Resources.Cancel + " <worker_program>", 2);
                _busy1.Set(); // Unblock worker so it can see that
```

```
                return 1;}
            catch (NullReferenceException ex){
                Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message + "): "
+ ex);
                return -1;}
            catch (Exception ex){
                Wri.ConWrite(Resources.Cancelling + " <worker_program>: " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);
                var tempor = new MessageBoxOk(Resources.Exeption_raised + "(" +
Resources.Cancelling + " <worker_program>): " + ex, Resources._0ERROR);
                tempor.ShowDialog();
                return -2;}
        }

        private int CancelWorker2()
        {
            Wri.ConWrite(Resources.Cancel_request + " (<worker_test>)");
            try{
                if (!worker_test.IsBusy) return 2;
                worker_test.CancelAsync();
                Wri.ConWrite(Resources.Cancel + " <worker_test>", 2);
                _busy2.Set(); // Unblock worker so it can see that
                return 1;}
            catch (NullReferenceException ex){
                Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message + "): "
+ ex);
                return -1;}
            catch (Exception ex){
                Wri.ConWrite(Resources.Cancelling + " <worker_test>: " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);
                var tempor = new MessageBoxOk(Resources.Exeption_raised + "(" +
Resources.Cancelling + " <worker_test>): " + ex, Resources._0ERROR);
                tempor.ShowDialog();
                return -2;}
        }

        private string ReadFile(string filePath)
        {
            try{
                string text;
                using (var streamReader = new StreamReader(filePath,
Encoding.UTF8)) { text = streamReader.ReadToEnd(); }
```

```
                return text;}
            catch (Exception ex)
            {
                string str1 = null;
                string str2 = null;
                if (ex is FileNotFoundException){
                    str1 = Resources.The_tests_program_file_was_not_found + ". " +
Resources.Exiting_program + ".\n" + ex;
                    str2 = Resources.__Error + ", " + Resources._file_not_found;}
                else if (ex is IOException){
                    str1 = Resources.The_tests_program_file_could_not_be_accessed
+ ". " + Resources.Exiting_program + ".\n" + ex;
                    str2 = Resources.__Error + ", " +
Resources._cannot_access_file;}
                else if (ex is FileFormatException || ex is FileLoadException){
                    str1 = Resources.The_tests_program_file_cannot_be_opened + ".
" + Resources.Exiting_program + ".\n" + ex;
                    str2 = Resources.__Error + ", " +
Resources._cannot_open_file;}
                else{
                    str1 = Resources.The_tests_program_file_has_other_error + ". "
+ Resources.Exiting_program + ".\n" + ex;
                    str2 = Resources.__Error + ", " +
Resources._unexpected_error;}
                Wri.ConWrite(Resources.Reading_file + " (" + filePath + ")" +
Resources.Exception_raised + " (" + ex.Message + "): " + ex, 2);
                WriteDebugAfterEx();
                var tempor = new MessageBoxOk(str1, str2);
                tempor.ShowDialog();
                ExitProgram();
            }
            return "";
        }

        private int ProgramManual(string line, string reference, bool referen, out
int testTime) //this sets 1 MANU test
        {
            Wri.ConWrite(Resources.Program_manual_test + ": " + line);
            const int writeTimeout = 50;
            int success = 1;
            testTime = 0;
            string[] separateCommands = line.Split(';');
```

```
            if (separateCommands.Any())
            {
                if (separateCommands[0] == "end"){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Wrong_line_in_commands);
                    return -1;}
            }
            if (separateCommands.Count() < 12){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Wrong_line_in_commands);
                return 0;}
            string mode = separateCommands[0]; //mode
            string name = separateCommands[1]; //name
            string param1 = separateCommands[2].Replace(",", "."); //volt or curr
            string param2 = separateCommands[3]; //freq
            string param3 = separateCommands[4].Replace(",", "."); //LO set
            string param4 = separateCommands[5].Replace(",", "."); //HI set
            string param5;
            if (mode == "GB" && referen){
                param5 = reference;
                Wri.ConWrite(Resources.Using_reference_from_zerocheck + ": " +
reference);}
            else param5 = separateCommands[6].Replace(",", "."); //ref
            string param6 = separateCommands[7].Replace(",", "."); //test timer
            string param7 = separateCommands[8].Replace(",", "."); //rampup
            string param8 = separateCommands[9]; //ARC mode
            string param9 = separateCommands[10].Replace(",", "."); //ARC limit
            string param10 = separateCommands[11]; //groundmode
            try{
                testTime = (int) (decimal.Parse(param6,
CultureInfo.InvariantCulture) + decimal.Parse(param7,
CultureInfo.InvariantCulture) + 1);}
            catch (FormatException ex)
            {
                Wri.ConWrite(Resources.Exception_raised + " (" + ex.Message + "):
" + ex + "\n" + Resources.Error_converting + " testtime");
                decimal testTime1 = 0;
                decimal testTime2 = 0;
                try { testTime1 = decimal.Parse(param6,
CultureInfo.InvariantCulture); }
                catch (FormatException ex1){
```

```
                    Wri.ConWrite(Resources.Exception_raised + " (" + ex1.Message +
"): " + ex1 + "\n" + Resources.Error_converting + " testtime1");}
                try { testTime2 = decimal.Parse(param7,
CultureInfo.InvariantCulture); }
                catch (FormatException ex1){
                    Wri.ConWrite(Resources.Exception_raised + " (" + ex1.Message +
"): " + ex1 + "\n" + Resources.Error_converting + " testtime2");}
                testTime = (int) (testTime1 + testTime2 + 1);
                if (testTime <= 1) success = 0;
            }
            catch (Exception ex){
                Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message + "): "
+ ex);
                var tempor = new MessageBoxOk(Resources.Exeption_raised + "(" +
Resources._reading_testtime + "): " + ex, Resources._0ERROR);
                tempor.ShowDialog();
                success = 0;}
            if (!SendDataWithCheck("MANU:EDIT:MODE " + mode)){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._mode + " (" + mode + ") " +
Resources._is_wrong);
                success = 0;}
            Thread.Sleep(writeTimeout);
            SendData("MANU:NAME " + name); //send name. sends only max 10 charac
            Thread.Sleep(writeTimeout);
            if (mode == "GB") //send voltage or current
            {
                if (!SendDataWithCheck("MANU:" + mode + ":CURR " + param1)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._current + " (" + param1 + ") "
+ Resources._is_wrong, 2);
                    success = 0;}
            }
            else
            {
                if (!SendDataWithCheck("MANU:" + mode + ":VOLT " + param1)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._voltage + " (" + param1 + ") "
+ Resources._is_wrong + "", 2);
                    success = 0;}
            } //--send voltage or current
            Thread.Sleep(writeTimeout);
            if (mode == "ACW" || mode == "GB") //send freq
```

```
            {
                if (!SendDataWithCheck("MANU:" + mode + ":FREQ " + param2)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._frequency + " (" + param2 + ")
" + Resources._is_wrong, 2);
                    success = 0;}
            } //--send freq
            Thread.Sleep(writeTimeout);
            if (mode == "IR" || mode == "GB") //send HiSet
            {
                if (!SendDataWithCheck("MANU:" + mode + ":RHIS " + param4)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.HiSet + " (" + param4 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
            }
            else
            {
                if (!SendDataWithCheck("MANU:" + mode + ":CHIS " + param4)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.HiSet + " (" + param4 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
            } //--send HiSet
            Thread.Sleep(writeTimeout);
            if (mode == "IR" || mode == "GB") //send LoSet
            {
                if (!SendDataWithCheck("MANU:" + mode + ":RLOS " + param3)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.LoSet + " (" + param3 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
            }
            else
            {
                if (!SendDataWithCheck("MANU:" + mode + ":CLOS " + param3)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.LoSet + " (" + param3 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
            } //--send LoSet
            Thread.Sleep(writeTimeout);
            if (!SendDataWithCheck("MANU:" + mode + ":REF " + param5)){ //send ref
```

```
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._reference_value + " (" +
param5 + ") " + Resources._is_wrong, 2);
                success = 0;} //--send REF
            Thread.Sleep(writeTimeout);
            if (!SendDataWithCheck("MANU:" + mode + ":TTIM " + param6)){ //send
test time
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._test_time + " (" + param6 + ")
" + Resources._is_wrong, 2);
                success = 0;} //--send test time
            Thread.Sleep(writeTimeout);
            if (mode != "GB") //send ramp up time
            {
                if (!SendDataWithCheck("MANU:RTIM " + param7)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._ramp_up_time + " (" + param7 +
") " + Resources._is_wrong, 2);
                    success = 0;}
            } //--send ramp up time
            Thread.Sleep(writeTimeout);
            if (mode == "ACW" || mode == "DCW") //set arc detection mode
            {
                if (!SendDataWithCheck("MANU:UTIL:ARCM " + param8)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._arc_mode + " (" + param8 + ")
" + Resources._is_wrong, 2);
                    success = 0;}
                Thread.Sleep(writeTimeout);
                if (param8 == "ON") //set arc current
                {
                    if (!SendDataWithCheck("MANU:" + mode + ":ARCC " + param9)){
                        Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._arc_current + " (" + param9 +
") " + Resources._is_wrong, 2);
                        success = 0;}
                } //--set arc current
            } //--set arc detection mode
            Thread.Sleep(writeTimeout);
            if (mode == "ACW" || mode == "DCW") //set groundmode
            {
                if (!SendDataWithCheck("MANU:UTIL:GROUNDMODE " + param10)){
```

```
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._ground_mode + " (" + param10 +
") " + Resources._is_wrong, 2);
                    success = 0;}
            } //--set groundmode
            if (!SendDataAndReadEncased("MANU:UTIL:PASS ON", "MANU:UTIL:PASS?",
"ON")){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._pass_hold_cannot_be_set, 2);
                success = 0;}
            if (!SendDataAndReadEncased("MANU:UTIL:FAIL STOP", "MANU:UTIL:FAIL?",
"STOP")){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._fail_hold_cannot_be_set, 2);
                success = 0;}
            if (!SendDataAndReadEncased("MANU:UTIL:MAXH ON", "MANU:UTIL:MAXH?",
"ON")){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._fail_hold_cannot_be_set, 2);
                success = 0;}
            return success;
        }

        private ZeroReference CheckForGB(int max, string[] readFile2Split)
        {//zerocheck function
            var refer = new ZeroReference {Referen = false, Reference = "0"};
            for (int i = 1; i < max; i++) //check if zerocheck needed
            {
                try
                {
                    if (readFile2Split[i].Split(';').Count() <= 11) continue;
                    if (readFile2Split[i].Split(';')[0] != "GB") continue;//if
programs contain test in GB mode
                    Wri.ConWrite(Resources.GB_mode_detected);
                    var zeroCheck = new
MessageBoxAlt(Resources.There_is_a_GB_test, Resources.There_is_a_GB_test + ". " +
Resources.Would_you_like_to_do_a_zerocheck + "?\n" +
Resources.To_automatically_measure_reference_resistance + ",\n" +
Resources._put_the_leads_in_zeroing_position_and_press__Yes_ + ".\n" +
Resources.To_skip_this_press__No__, false, false);
                    DialogResult toDoZero = zeroCheck.ShowDialog();
                    if (toDoZero == DialogResult.Yes){ //do zerocheck
                        refer.Referen = true;
```

```
                        refer.Reference = ZeroCheck(_manualTestNo - i);//do
zerocheck
                        if (refer.Reference == "_fail")
                            refer.Referen = false;}//zerocheck failed
                    break;
                }
                catch (IndexOutOfRangeException ex){
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message +
"): " + ex);
                    WriteDebugAfterEx();}
            }
            return refer;
        }

        private string ZeroCheck(int n)
        {
            Wri.ConWrite(Resources.Zerocheck_started, 1);
            bool retry = false;
            string reference = "_fail";
            SendDataAndReadEncased("MAIN:FUNC MANU", "MAIN:FUNC?", "MANU
MODE");//set and check: tester to manu mode
            SendDataAndReadEncased("MANU:STEP " + IntTo3String(n), "MANU:STEP",
IntTo3String(n));//set and check: tester change test number
            SendDataAndReadEncased("MANU:EDIT:MODE GB", "MANU:EDIT:MODE?",
"GB");//set and check: tester to GB test
            do
            {
                SendDataWithCheck("MANU:GB:ZEROCHECK ON");//set zerocheck on
                SendDataWithCheck("FUNC:TEST ON");//test on
                while (!SendQuery("FUNC:TEST?", "TEST OFF")){//wait while test
ends
                    Thread.Sleep(500);}
                Thread.Sleep(500);
                string str = GetDataEncased("MANU:GB:REF?");//read reference data
                if (str.Split(',')[0] == "98")
                {//if result is error
                    var goAgain = new
MessageBoxAlt(Resources.Error_getting_reference_value,
Resources.There_was_a_problem_getting_reference_value + ".\n" +
Resources.Check_the_positions_of_the_leads_and_press__yes_to_try_again + ".\n" +
Resources.Press__no__to_use_reference_value_from_program_file_, false, false);
                    DialogResult errorGoAgain = goAgain.ShowDialog();
                    if (errorGoAgain == DialogResult.No) return "_fail";
```

```
                    if (errorGoAgain == DialogResult.Yes){
                        retry = true;
                        continue;}
                }
                reference = str.Split('m')[0];//only get number
                Wri.ConWrite("REF: " + reference);
                var refValue = new MessageBox3(Resources.Zerocheck_completed,
Resources.Zerocheck_completed + ".\n" + Resources.The_reference_value_is_ +
reference + " mOhm.\n " + Resources.If_this_value_is_good_then_press__yes_ + ".\n"
+
Resources.If_the_value_is_not_good_then_check_the_positions_of_the_leads_and_press
__no__to_try_again + ".\n" +
Resources.Press__cancel__to_skip_the_test_and_use_reference_value_from_the_program
_file, false);//ask if reference value is ok
                DialogResult referenceValue = refValue.ShowDialog();
                if (referenceValue == DialogResult.Cancel) return "_fail";
                if (referenceValue == DialogResult.No) retry = true;
                else if (referenceValue == DialogResult.Yes){
                    return reference;}
            } while (retry);
            return reference;
        } //--PROGRAMMING DEVICE

        #endregion Programming Device

        #region Testing

        private void worker_test_DoWork(object sender, DoWorkEventArgs e) //
TESTING
        {
            SetElements(false);//make all elements inactive
            this.SynchronizedInvoke(() => label2.Text = Resources.Testing);
            worker_test.ReportProgress(0);
            Thread.Sleep(15);
            foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes){//testing
node images to hourglass
                SetNodeImage(node, 0);}
            Wri.ConWrite(Resources.Testing_subroutine_start);
            SetNodeImage(treeView1.Nodes[0].Nodes[1], 2); //testing node to
inprogress
            SetImage(2);//set image to test in progress
            StartWorker(worker_test);
            int testR = TestWhile();
```

```
            if (testR == -1){//if test was cancelled
                Wri.ConWrite(Resources.Test_loop_canceled);
                SetImage(4);
                SetNodeImage(treeView1.Nodes[0].Nodes[1], 4);}
            SendData("FUNC:TEST OFF");
            Wri.ConWrite(Resources.Testing_stopped);
            CancelWorker2();//stop backgroundworker
        }


        private void worker_test_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            SetElements(true);//set elements to accesible
            Wri.ConWrite(Resources.Testing_ended);
            if (!SendDataAndReadEncased("FUNC:TEST OFF", "FUNC:TEST?", "TEST
OFF"))
                StopTest();//if test was running then start function to stop it
            if (e.Error != null){ // First, handle the case where an exception was
thrown
                Wri.ConWrite(Resources.Testing + "(<worker_program>) " +
Resources._completed_with_error + ": " + e.Error.Message, 2);
                label2.Text = Resources.Error_when_testing;}
            else if (e.Cancelled){//backgroundworker cancelled
                Wri.ConWrite(Resources.Testing + "(<worker_program>) " +
Resources._cancelled, 3);
                label2.Text = Resources.Testing_cancelled;}
            else{ // Finally, handle the case where the operation succeeded
                Wri.ConWrite(Resources.Testing + "(<worker_program>) " +
Resources._finished, 1);
                label2.Text = Resources.Testing_finished;}
            _busy2.Set();
            try{
                _cancel.Dispose();//dispose of backgroundworker
                Wri.ConWrite(Resources.Disposed_of + " <cancelling> " +
Resources._window);}
            catch (NullReferenceException ex){
                Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message + "): "
+ ex);
                WriteDebugAfterEx();}
            catch (Exception ex){
                Wri.ConWrite(Resources.Disposing_of + " <cancelling>." +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);
```

```
                WriteDebugAfterEx();
                var tempor = new MessageBoxOk(Resources.Exeption_raised + "(" +
Resources._close_cancel_form + "): " + ex, Resources._0ERROR);
                tempor.ShowDialog();}
            ToggleTestPic(false);
            PauseWorker(worker_test);
            WriteDebug();
        }

        private void worker_test_ProgressChanged(object sender,
ProgressChangedEventArgs e){
            progressBar1.Value = e.ProgressPercentage;} //change progresbar

        public void StartWorker(BackgroundWorker worker)
        {
            if (!worker.IsBusy){
                Wri.ConWrite(Resources.Will_start_worker + " (" +
Resources._not_busy + ") " + worker);
                worker.RunWorkerAsync();}
            Wri.ConWrite(Resources.Start_worker);
            if (worker == worker_program){
                _busy1.Set(); // Unblock the worker
                Wri.ConWrite(Resources.Start + " <worker_program>");}
            if (worker == worker_test){
                _busy2.Set(); // Unblock the worker
                Wri.ConWrite(Resources.Start + " <worker_test>");}
        }

        public void PauseWorker(BackgroundWorker worker)
        {
            Wri.ConWrite(Resources.Pause_worker);
            if (worker == worker_program){
                _busy1.Reset(); // Block the worker
                Wri.ConWrite(Resources.Pause + " <worker_program>");}
            if (worker == worker_test){
                _busy2.Reset(); // Block the worker
                Wri.ConWrite(Resources.Pause + " <worker_test>");}
        }

        public int TestWhile()
```

```
        {
            DialogResult dialRes;
            int imag = 2;
            string result = null;
            do
            {
                var logging = new List<string>();
                this.SynchronizedInvoke(() => label2.Text = Resources.Testing);
                worker_test.ReportProgress(0);
                Thread.Sleep(15);
                foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes){
                    SetNodeImage(node, 0);} //all test nodes to hourglass
                SetImage(0);
                SendData("MAIN:FUNC MANU"); //set tester mode to manu
                Thread.Sleep(15);
                SendData("FUNC:TEST OFF");//set tester test off, just in case
                SetNodeImage(treeView1.Nodes[0].Nodes[1], 2); //testing node01 to
in progress
                SetImage(2);
                if (CancelPending(worker_test)) return -1;
                _busy2.WaitOne();
                SendDataAndReadEncased("MAIN:FUNC AUTO", "MAIN:FUNC?", "AUTO
MODE");//set tester mode to auto and check
                SendDataAndReadEncased("AUTO:STEP " + IntTo3String(_autoTestNo),
"AUTO:STEP?", IntTo3String(_autoTestNo));//change test number and check
                int testW = TestEach(ref logging, true);
                if (testW < 0)
                {
                    if (testW > -3){//testing cancelled
                        Wri.ConWrite(Resources.Test_canceled, 1);
                        this.SynchronizedInvoke(() => label2.Text =
Resources.Testing_cancelled);
                        return -1;}
                    Wri.ConWrite(Resources.Testing_error + ". " +
Resources.Check_connection_to_the_tester);
                    this.SynchronizedInvoke(() => label2.Text =
Resources.Testing_error);
                    return -2;
                }
                if (testW == 1){//test passed
                    imag = 3;
```

```
                    Wri.ConWrite(Resources.Test_passed + " " +
_currentTest.AutoName + " (" + Resources._product_name + ") - " + SerialNo.Text +
" (" + Resources._serial_no + ")", 1);
                    logging.Add("PASS;" + _currentTest.AutoName + " (PRODUCT
NAME);" + SerialNo.Text + " (SERIAL NO)");
                    result = Resources.Test_passed;}
                else if (testW == 0){//test failed
                    imag = 1;
                    Wri.ConWrite(Resources.Test_failed + " " +
_currentTest.AutoName + " (" + Resources._product_name + ") - " + SerialNo.Text +
" (" + Resources._serial_no + ")", 1);
                    logging.Add("FAIL;" + _currentTest.AutoName + " (PRODUCT
NAME);" + SerialNo.Text + " (SERIAL NO)");
                    result = Resources.Test_failed;}
                foreach (string str in logging)
                    WriteLogCsv(str);//write all strings about tests to log
                if (SendQuery("FUNC:TEST?", "TEST ON")){
                    SendData("FUNC:TEST OFF");
                    Wri.ConWrite(Resources.Test_was_on + ". " +
Resources.Turning_off);}
                SetNodeImage(treeView1.Nodes[0].Nodes[1], imag); //test image to
result
                SetImage(imag);
                SendData("MAIN:FUNC MANU");
                worker_test.ReportProgress(100);
                Thread.Sleep(15);
                if (CancelPending(worker_test)) return -1;//cancel request
pending, exit
                var next = new MessageBoxAlt(Resources.Next_product, result + "\n"
+ Resources.Continue_testing_with_next_product + "?", false, false);
                dialRes = next.ShowDialog();
            } while (dialRes == DialogResult.Yes);//do while you choose 'yes'
            return 1;
        }

        public int TestEach(ref List<string> logging, bool lasttestcompleted)
        {
            int progN = 0;
            bool state = true;
            int report = _programList.Count();
            foreach (TestList test in _programList)
            {
                progN++;
```

```
                worker_test.ReportProgress((int) ((progN - 0.5)*100/report));
                Thread.Sleep(15);
                if (CancelPending(worker_test)) return -1;
                SetNodeImage(treeView1.Nodes[0].Nodes[1].Nodes[progN - 1], 2);
//testing node to in progress
                PauseWorker(worker_test);
                if (CancelPending(worker_test)) return -1;
                var resumeTest = new DialogResult();
                if (progN == 1)//is it first manual test
                {
                    var resume = new BarCode(this, test.TestDescription,
lasttestcompleted, true);
                    resumeTest = resume.ShowDialog();//dialogue to enter serial
number
                    this.SynchronizedInvoke(() => SerialNo.Text =
resume.textBox_barCode.Text);
                    if (String.IsNullOrEmpty(SerialNo.Text))
                        this.SynchronizedInvoke(() => SerialNo.Text = "0");
                    Wri.ConWrite(Resources.Tested_product + ": " + SerialNo.Text,
1);
                    logging.Add("TEST RESULTS;" + _currentTest.AutoName + "
(PRODUCT NAME);" + SerialNo.Text + " (SERIAL NO)");
                }
                else{
                    var resume = new BarCode(this, test.TestDescription,
lasttestcompleted, false);
                    resumeTest = resume.ShowDialog();}//dialogue without entering
barcode
                if (resumeTest == DialogResult.Yes){
                    StartWorker(worker_test);}
                else{
                    CancelWorker2();
                    SetNodeImage(treeView1.Nodes[0].Nodes[1].Nodes[progN - 1], 4);
                    return -2;}
                if (CancelPending(worker_test)) return -1;
                _busy2.WaitOne();
                Wri.ConWrite(Resources.Testing_time_is + ": " + test.TestingTime);
                _myTimer.Interval = 1000*test.TestingTime;//set timer interval to
testing time
                Wri.ConWrite(Resources.Test_timer + ": " + _myTimer.Interval);
                Wri.ConWrite(Resources.Start_the_tester);
                ToggleTestPic(true);
```

```
                SendData("FUNC:TEST ON");
                _elapsed = false;
                _myTimer.Stop();
                _myTimer.Start();//start timer
                while (!_elapsed)
                {//wait until timer runs down
                    Thread.Sleep(200);
                    if (CancelPending(worker_test)) return -1;
                }
                SendData("FUNC:TEST?");
                string received2 = GetIncomingData(returnPort());
                if (received2 != "TEST OFF")
                {
                    _myTimer.Stop();
                    _elapsed = false;
                    Wri.ConWrite(Resources.Test_still_not_finished);
                    _myTimer.Start();
                    do
                    {
                        if (CancelPending(worker_test)) return -1;
                        Thread.Sleep(1000);
                        SendData("FUNC:TEST?");//ask test status
                        received2 = GetIncomingData(returnPort());//get answer
                        if (!_elapsed) continue;
                        if (!SendQuery("FUNC:TEST?", "TEST ON")) return -3;
                        Wri.ConWrite(Resources.Test_cannot_be_turned_off + ". " +
Resources.Do_it_manually, 2);
                        var tempor = new
MessageBoxOk(Resources.It_seems_that_the_test_is_still_running + ".\n" +
Resources.Exiting_testing + ".\n" +
Resources.If_the_test_should_have_stopped_then_turn_off_the_tester,
Resources._0ERROR);
                        tempor.ShowDialog();//ask if everything is OK
                        StopTest();
                        ExitProgram();
                    } while (received2 != "TEST OFF");//do while tester's test is
turned off
                }

                ToggleTestPic(false);
                for (int i = 0; i < 3; i++)
```

```
                {
                    SendData("MEAS" + progN + "?");//ask for measurement results
                    received2 = GetIncomingData(returnPort());//get results
                    if (received2.Split(',')[0] != "98")
                    {
                        Wri.ConWrite(Resources.Test_result + ": " + received2);
                        int imag;
                        if (received2.Split(',')[1].Trim() == "PASS" ||
received2.Split(',')[1].Trim() == "HOLD"){//if OK, display results and add to log
                            imag = 3;
                            Wri.ConWrite(Resources.Subtest + " " + progN + " " +
Resources._passed + ": " + received2.Replace("HOLD", "PASS"), 1);
                            logging.Add(";PASS;SUBTEST" + progN + ";" +
received2.Replace("HOLD", "PASS"));}
                        else if (received2.Split(',')[1].Trim() == "FAIL"){//if
failed, display results and add to log
                            imag = 1;
                            state = false;
                            Wri.ConWrite(Resources.Subtest + " " + progN + " " +
Resources._failed + ": " + received2, 1);
                            logging.Add(";FAIL;SUBTEST" + progN + ";" +
received2);}
                        else{//failed on other reason
                            imag = 4;
                            state = false;
                            Wri.ConWrite(Resources.Subtest + " " + progN + " " +
Resources._failed + " (" + "error" + ")" + ": " + received2, 1);
                            logging.Add(";FAIL;SUBTEST" + progN + ";" +
received2);}
                        SetNodeImage(treeView1.Nodes[0].Nodes[1].Nodes[progN - 1],
imag);
                        break;
                    }
                    Wri.ConWrite(Resources.No_result_for_test);
                }
                worker_test.ReportProgress(progN*100/report);
                Thread.Sleep(15);
                if (!state){
                    Wri.ConWrite(Resources.Test_failed);
                    return 0;}
                _busy2.WaitOne();
                if (CancelPending(worker_test)) return -1;
```

```
            }
            return 1;
        }

        public bool ToggleTestPic(bool toggle)
        {
            if (toggle)
            {
                this.SynchronizedInvoke(() =>{
                    label_testing.Text = Resources._0TESTING;
                    label_testing2.Text = Resources._0BE_0CAREFUL;
                    pictureBox_testing.Show();});//warnings to testing in progress
state

                return true;
            }
            this.SynchronizedInvoke(() =>{//if test not running, then warnings off
                label_testing.Text = " ";
                label_testing2.Text = " ";
                pictureBox_testing.Hide();});
            return false;
        }

        public int GetTestTime(int step)
        {
            SendDataAndReadEncased("MAIN:FUNC MANU", "MAIN:FUNC?", "MANU
MODE");//tester change and test: to manu mode
            SendDataAndReadEncased("MANU:STEP " + IntTo3String(step),
"MANU:STEP?", IntTo3String(step));//tester change and test: change test
            string inC = GetDataEncased("MANU:EDIT:MODE?");//get test mode
            inC = inC.Split(' ')[0];
            if (inC == "ACW" || inC == "DCW" || inC == "IR" || inC == "GB")
            {
                string mode = inC;
                int a = TestTime(mode);
                int b = RampTime(mode);
                if (a >= 0 && b >= 0){
                    Wri.ConWrite(Resources.Total_test_time_is + ": " + (a + b));
                    return (a + b);}
            }
```

```
            Wri.ConWrite(Resources.Error_getting_test_time);
            return -1;
        }

        private int TestTime(string mode){
            string inC = GetDataEncased("MANU:" + mode + ":TTIM?");//get testtime
from tester
            int testT = int.Parse(inC.Split('.')[0]) + 1;//convert to decimal
            Wri.ConWrite(Resources.Test_time + ": " + inC + " (" + testT + ")");
            return testT;}

        private int RampTime(string mode){
            if (mode == "GB") return 0;
            string inC = GetDataEncased("MANU:" + mode + ":TTIM?");//get ramptime
from tester
            int testT = int.Parse(inC.Split('.')[0]) + 1;//convert to decimal
            Wri.ConWrite(Resources.Ramp_up_time + ": " + inC + " (" + testT +
")");
            return testT;}

        private void ChangeLastHold(int prog)
        {
            SendDataAndReadEncased("MAIN:FUNC MANU", "MAIN:FUNC?", "MANU
MODE");//change tester to manu mode and check
            SendDataAndReadEncased("MANU:STEP " + IntTo3String(prog),
"MANU:STEP?", IntTo3String(prog));//cahnge tester test
            SendDataAndReadEncased("MANU:UTIL:PASS OFF", "MANU:UTIL:PASS?",
"OFF");//set 'pass' parameter off
        } //--TESTING

        #endregion Testing

        #region Building Treeview

        private void FillTreeView(string[] names) //BUILDING TREENODE
        {
            var items = new List<ItemInfo>{//add nodes
                new ItemInfo {ID = 1, ParentID = 0, Name = Resources.Testing},
                new ItemInfo {ID = 2, ParentID = 1, Name = Resources.Programming},
                new ItemInfo {ID = 3, ParentID = 1, Name = Resources.Testing},
                new ItemInfo {ID = 4, ParentID = 1, Name =
Resources.Writing_log},};
```

```
            int id = 5;
            var childs = new List<int>();
            foreach (string str in names){//add childnodes for each programming
and test
                items.Add(new ItemInfo {ID = id, ParentID = 2, Name = str});//test
childnode for programming
                items.Add(new ItemInfo {ID = id + 1, ParentID = 3, Name =
str});//test childnode for testing
                childs.Add(id + 1);
                id += 2;}
            FillNode(items, null);
        }

        private void FillNode(List<ItemInfo> items, TreeNode node)
        {
            int parentID = node != null
                ? (int) node.Tag
                : 0;//init node
            TreeNodeCollection nodesCollection = node != null
                ? node.Nodes
                : treeView1.Nodes;//init nodecollection
            foreach (ItemInfo item in items.Where(i => i.ParentID == parentID))
            {
                if (InvokeRequired)
                {
                    Invoke((MethodInvoker) delegate{
                        TreeNode newNode = nodesCollection.Add(item.Name,
item.Name);//create new node according to parameters
                        newNode.Tag = item.ID;
                        FillNode(items, newNode);});
                }
                else{
                    TreeNode newNode = nodesCollection.Add(item.Name, item.Name);
                    newNode.Tag = item.ID;
                    FillNode(items, newNode);}
            }
        } //--BUILDING TREENODE

        #endregion Building Treeview

        #region Sending Data
```

```
        public void SendData(string toSend, bool newline = true) //SENDING DATA
        {//send data to Sefelec tester
            if (serialPort1.IsOpen)//serial port is open
            {
                Thread.Sleep(50);
                try{
                    serialPort1.Write(toSend);//send data
                    if (newline) serialPort1.Write("\n\r");} //newline
                catch (IOException ex){
                    Wri.ConWrite(Resources.Problem_with_connection_to_tester + ".
" + Resources._0IO_Exeption_raised + "(" + ex.Message + "): " + ex, 2);}
                Wri.ConWrite(Resources._data_send_success + ": " + toSend);
            }
            else{//serial port is not open
                Wri.ConWrite(Resources.Cannot_send_data + ", " +
Resources._not_connected_to_serial_port, 2);}
        }

        private bool SendDataWithCheck(string toSend)
        {//send data to Sefelec tester and check errors
            if (serialPort1.IsOpen)
            {
                SendData("*CLS\n");
                Thread.Sleep(50);
                SendData(toSend);
                Thread.Sleep(100);
                string[] error = GetError();//read latest error from tester
                if (Char.IsNumber(error[0][0]))//result is number
                    if (int.Parse(error[0]) != 0){//result is not 0, 0 is no error
                        return false;}
            }
            else{
                Wri.ConWrite(Resources.Cannot_send_data + " (SendDataWithCheck)" +
", " + Resources._not_connected_to_serial_port, 2);
                return false;}
            return true;
        }

        public bool SendQuery(string query, string answer)
```

```
        {//send data request to Sefelec tester
            if (serialPort1.IsOpen)
            {
                //string received = null;
                for (int i = 0; i < 3; i++){//try max 3 times
                    SendData("*CLS");
                    Thread.Sleep(100);
                    SendData(query);//send question
                    Thread.Sleep(100);
                    string received = GetIncomingData(returnPort());//get answer
                    Wri.ConWrite("SendQuery: " + received);
                    if (received.Trim() == answer) return true;//if as expected,
send OK
                    Thread.Sleep(200);}
                return false;
            }
            Wri.ConWrite(Resources.Cannot_send_query + ": " + query + "
(SendQuery), " + Resources._not_connected_to_serial_port, 2);
            return false;
        }

        private string[] GetError()
        {//get last error message from Sefelec tester
            EmptyIncomingBuffer(returnPort());//empty buffer
            SendData("SYST:ERR?");//ask for latest error
            Thread.Sleep(100);
            string error = GetIncomingData(returnPort());//get incoming data
            Wri.ConWrite("GetError: " + error);
            string[] errorSp = error.Split(',');//split error to number and
description
            if (!Char.IsNumber(errorSp[0][0]))
            {//if not a number
                Thread.Sleep(50);
                error = GetIncomingData(returnPort());//try reading again
                Wri.ConWrite("GetError (" + Resources._try + "2): " + error);
                errorSp = error.Split(',');
                if (!Char.IsNumber(errorSp[0][0])){//if not a number
                    string[] t = {"99",
Resources._custom_error__problem_getting_the_rigth_answer_from_device};
                    return t;}//can't read error
            }
```

```
            if (int.Parse(errorSp[0]) == 0) return errorSp; //returns error
            if (DebugWin.InvokeRequired){
                DebugWin.Invoke(new MethodInvoker(() =>
Wri.ConWrite(Resources.Error_from_device + "(Sefelec)" + ": " + error)));}
            else{
                Wri.ConWrite(Resources.Error_from_device + "(Sefelec)" + ": " +
error);}
            return errorSp; //returns error
        }

        private void EmptyIncomingBuffer(SerialPort
serial){serial.DiscardInBuffer();}

        private string GetIncomingData(SerialPort serial)
        {//get data from serial port buffer
            string str;
            int i = 0;
            do
            {
                i++;
                Thread.Sleep(100);
                try{
                    str = serial.ReadExisting();}//read data from buffer
                catch (Exception ex){
                    str = Resources._connection_error;
                    Wri.ConWrite(Resources.Problem_with_connection_to_tester + ".
" + Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);}
                Wri.ConWrite("ReadExixting: " + str.Replace(Environment.NewLine,
""));
            } while (str == String.Empty && i < 3);
            if (str == String.Empty)//nothing in buffer
                return "98" + ", " + Resources._error_no_answer;
            return str.Replace(Environment.NewLine, "");
        }

        public bool SendDataAndRead(string data, string question, string answer)
        {//send data, question and read answer to Sefelec tester
            string received = null;
            if (serialPort1.IsOpen)
            {
                for (int i = 0; i < 3; i++){
```

```
                    SendData("*CLS");
                    Thread.Sleep(100);
                    SendData(data);//send data
                    Thread.Sleep(100);
                    SendData(question);//send query
                    Thread.Sleep(100);
                    received = GetIncomingData(returnPort());//get answer
                    Wri.ConWrite("SendandRead: " + received);
                    if (received.Trim() == answer) return true;//check if answer
is what was expected
                    Thread.Sleep(200);}
                Wri.ConWrite(Resources.Serious_error + ". " +
Resources.Did_not_get_the_right_answer + ": " + Resources._received + ":" +
received + " ," + Resources._expected + ":" + answer);
                return false;
            }
            Wri.ConWrite(Resources.Cannot_send_data + " (SendDataAndRead)" + ", "
+ Resources._not_connected_to_serial_port, 2);
            return false;
        }

        public bool SendDataAndReadEncased(string data, string question, string
answer)
        {//send data, question and read answer to Sefelec tester, encased in try,
safer
            while (true)
            {
                string received = null;
                if (serialPort1.IsOpen)
                {
                    for (int i = 0; i < 3; i++)
                    {
                        try{
                            SendData("*CLS");}
                        catch (IOException ex){

Wri.ConWrite(Resources.Problem_with_connection_to_tester + ". " +
Resources._0IO_Exeption_raised + "(" + ex.Message + "): " + ex, 2);}
                        Thread.Sleep(100);
                        SendData(data);//send data
                        Thread.Sleep(100);
                        SendData(question);//send query
```

```
                        Thread.Sleep(100);
                        received = GetIncomingData(returnPort());//get answer
                        Wri.ConWrite("SendandReadEncased: " + received);
                        if (received.Trim() == answer) return true;//check if
answer is what was expected
                        Thread.Sleep(200);
                    }
                    Wri.ConWrite(Resources.Serious_error + ". " +
Resources.Did_not_get_the_right_answer + ": " + Resources._received + ":" +
received + " ," + Resources._expected + ":" + answer);
                }
                else
                {
                    Wri.ConWrite(Resources.Cannot_send_data + "
(SendDataAndReadEncased)" + ", " + Resources._not_connected_to_serial_port, 2);
                    received = Resources._0SERIAL_0PORT_0IS_0NOT_0OPEN;
                }
                var retry = new MessageBoxAlt("Retry" + "?",
Resources.Error_sending_the_command + ":\n" + data + "\n" + Resources._received +
":\n" + received + "\n" + Resources.Unplug_and_replug_the_tester_if_nessesery +
"\n" + Resources.Press__yes__to_try_again + ".\n" +
Resources.Press__no__to_exit_the_program, true, false);
                DialogResult dailr = retry.ShowDialog();//ask to reset connection
and try again
                if (dailr == DialogResult.No) ExitProgram();
                else
                {
                    try{
                        OpenComPort(_serialCom);}
                    catch (Exception ex){
                        Wri.ConWrite(Resources.Error_reopening_0COM_port + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                        var tempor = new
MessageBoxOk(Resources.Error_reopening_0COM_port + ". " +
Resources.Exception_raised + "\n" + ex, Resources.Error_with_0COM_port);
                        tempor.ShowDialog();}
                }
            }
        }

        public string GetDataEncased(string data)
        {//send request for data to Sefelec tester and read answer, safer
            while (true)
```

```
            {
                string received = null;
                if (serialPort1.IsOpen)
                {
                    for (int i = 0; i < 3; i++)
                    {
                        EmptyIncomingBuffer(returnPort());
                        try{
                            SendData("*CLS");}
                        catch (IOException ex){

Wri.ConWrite(Resources.Problem_with_connection_to_tester + ". " +
Resources._0IO_Exeption_raised + "(" + ex.Message + "): " + ex, 2);}
                        Thread.Sleep(100);
                        SendData(data);//send data
                        Thread.Sleep(100);
                        received = GetIncomingData(returnPort());
                        Wri.ConWrite("GetDataEncased: " + received);//get answer
                        if (received.Trim() != "98" + ", " +
Resources._error_no_answer) return received;//check if answer is what was expected
                        Thread.Sleep(200);
                    }
                    Wri.ConWrite(Resources.Serious_error + ". " +
Resources.Did_not_get_an_answer_to + ": " + data + ", " + Resources._received +
":" + received);
                }
                else{
                    Wri.ConWrite(Resources.Cannot_send_data + "
(SendDataAndReadEncased)" + ", " + Resources._not_connected_to_serial_port, 2);
                    received = Resources._0SERIAL_0PORT_0IS_0NOT_0OPEN;}
                var retry = new MessageBoxAlt("Retry" + "?",
Resources.Error_sending_the_command + ":\n" + data + "\n" + Resources._received +
":\n" + received + "\n" + Resources.Press__yes__to_try_again + ".\n" +
Resources.Press__no__to_exit_the_program, true, false);
                DialogResult dailr = retry.ShowDialog();
                if (dailr == DialogResult.No) ExitProgram();
                else
                {//try reopening the serial port
                    try{
                        OpenComPort(_serialCom);}
                    catch (Exception ex){
                        Wri.ConWrite(Resources.Error_reopening_0COM_port + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
```

```
                        var tempor = new
MessageBoxOk(Resources.Error_reopening_0COM_port + ". " +
Resources.Exception_raised + "\n" + ex, Resources.Error_with_0COM_port);
                        tempor.ShowDialog();}
                }
            }
        }

        private SerialPort returnPort()
        {
            return serialPort1;
        }//--SENDING DATA

        #endregion Sending Data

        private class ItemInfo{ //struct for treeView
            public int ID;
            public string Name;
            public int ParentID;}

        public struct Listboxes{
            public ListBox Box;
            public string File;}

        public struct SavedTest
        {
            public string AutoName;
            public int AutoTestN;
            public string[] StartedTest;
            public bool TestIsProgrammed;
            public TestList[] TestParams;
            public override string ToString() { return AutoName; }//override if
ToString method called
        }

        public struct TestList{
            public string Relays;
            public string TestDescription;
            public string TestName;
            public int TestNo;
```

```
            public string TestParameters;
            public int TestingTime;}

        public struct ZeroReference{
            public bool Referen;
            public string Reference;}
    }
}
```

Programmi fail 3: BarCode.cs

```
using SefelecTester.Properties;
using System;
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public partial class BarCode : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT,
declare timer
        private bool _keyElapsed = true; //KEYPRESS EVENT

        public BarCode(MainWin main, string instructions, bool increment, bool
firstsub)
        {
            InitializeComponent();
            if (!IsHandleCreated)CreateHandle();//if not created, then force it
            KeyPreview = true; //KEYPRESS EVENT, receive keypress events
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT, follow timer
elapsed event
            textBox_barCode.LostFocus += ControlLostFocus;//focus lost from
barcode entry event
            textBox_barCode.Leave += ControlLostFocus;//focus lost from barcode
entry event
            textBox_barCode.Text = main.SerialNo.Text;
            label2.Text = Resources.Instructions + ": " + instructions;
```

```
            label_test.Text += "\n" + Resources.Pressing__OK__will_start_the_test
+ "\n" + Resources._place_the_leads_in_the_testing_position + "\n" +
Resources._0BEFORE_pressing__OK__;
            if (firstsub)//first time calling
            {
                textBox_barCode.Focus();
                if (main.checkBox1.Checked){
                    label1.Text = Resources.Scan_the_products_barcode;
                    textBox_barCode.Focus();
                    textBox_barCode.Focus();//double because sometimes 1 doesn't
work
                    textBox_barCode.Text = "";}
                else{
                    label1.Text = Resources.Change_the_barcode_manually + " (" +
Resources._if_needed + ")";
                    textBox_barCode.Text = increment ?
IncrementOne(main.SerialNo.Text) : main.SerialNo.Text;}
                textBox_barCode.Focus();
            }
            else{//already called before, no need to enter barcode
                textBox_barCode.Hide();
                label_barCode.Hide();
                label1.Hide();}
        }

        protected override sealed void CreateHandle(){
            base.CreateHandle();}

        private void ControlLostFocus(object sender, EventArgs e){
            textBox_barCode.Focus();}//refocus on barcode entry

        private void button_cancel_Click(object sender, EventArgs e){
            var canc = new MessageBoxAlt(Resources.Cancel_test + "?",
Resources.Are_you_sure + "?" + "\n" + Resources.This_will_cancel_the_test, false,
false);//conformation?
            DialogResult dail = canc.ShowDialog();
            if (dail == DialogResult.Yes) DialogResult = DialogResult.Cancel;
            else if (dail == DialogResult.No) DialogResult = DialogResult.None;}

        private static unsafe string IncrementOne(string str)
        {//increments last number in serial number (it might contain letters)
            bool added = false;
```

```
            fixed (char* pt = str)
            {
                for (int i = str.Length - 1; i >= 0; i--)
                {
                    int val = pt[i] - '0';
                    if (val < 0 || val > 9) // Current char isn't a digit
                    {
                        if (added){ // Digits have been found and processed
earlier
                            // Add 1 before the digits, because if the code
reaches this, it means it was something like 999, which should become 1000
                            str = str.Insert(i + 1, "1");
                            break;}
                        continue;
                    }
                    added = true;
                    if (val < 9){ // Digit isn't 9
                        pt[i] = (char)(val + 1 + '0'); // Set it to be itself + 1,
and break
                        break;}
                    pt[i] = '0'; // Digit is 9. Set it to be 0 and continue to
previous characters
                    if (i == 0) str = str.Insert(0, "1"); // Reached beginning of
string and should add 1 before digits
                }
            }
            return str;
        }


        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed)
            {
                if (msg.WParam.ToInt32() == (int)Keys.Enter){//enter pressed
                    return true;}//do nothing
            }
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();//restart timer
            return base.ProcessCmdKey(ref msg, keyData);//else let button press
action to be executed
```

```
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 4: InvokeC.cs

```
using System;
using System.ComponentModel;

namespace SefelecTester //InvokeC.SynchronizedInvoke(this, () => );
{
    internal static class InvokeC//static class, no need to create instance
    {//used to execue actions from background threads to UI
        public static void SynchronizedInvoke(this ISynchronizeInvoke sync, Action
action)//use invoke if nessesery
        {
            if (!sync.InvokeRequired) // If the invoke is not required, then
action is done here and exit
            {
                action(); // Execute action
                return;
            }
            sync.Invoke(action, new object[] { }); // Marshal to the required
context
        }

        public static void CertainInvoke(this ISynchronizeInvoke sync, Action
action)//always use invoke
        {
            sync.Invoke(action, new object[] { });
        }
    }
}
```

Programmi fail 5: ListBoxWriter.cs

```
using SefelecTester.Properties;
using System;
using System.IO;
```

```
using System.Text;
using System.Windows.Forms;

namespace SefelecTester
{
    public class ListBoxWriter : TextWriter //this class redirects
Console.Writeline() to debug listbox
    {
        private readonly ListBox _list;
        private StringBuilder _content = new StringBuilder();//for building string
        public ListBoxWriter(ListBox list){
            _list = list;}
        public override Encoding Encoding{//set encoding
            get { return Encoding.UTF8; }}

        public override void Write(char value)
        {
            base.Write(value);
            _content.Append(value);
            if (value != '\n') return; // InvokeRequired required compares the
thread ID of the
            // calling thread to the thread ID of the creating thread.
            // If these threads are different, it returns true.
            if (_list.InvokeRequired)
            {
                //SetTextCallback d = new SetTextCallback(SetText);
                try{
                    _list.Invoke(new MethodInvoker(() =>
_list.Items.Add(_content.ToString())));//add to listbox
                    _list.Invoke(new MethodInvoker(() => _list.SelectedIndex =
_list.Items.Count - 1));//scroll to bottom
                    _list.Invoke(new MethodInvoker(() => _list.SelectedIndex = -
1));}//scroll to bottom
                catch (ObjectDisposedException ex){
                    Console.WriteLine(Resources.Exception_raised + " (" +
ex.Message + "): " + ex);}
            }
            else{//do same without invoke
                _list.Items.Add(_content.ToString());
                _list.SelectedIndex = _list.Items.Count - 1;
                _list.SelectedIndex = -1;}
```

```
            _content = new StringBuilder();
        }
    }
}
```

Programmi fail 6: LoadProgram.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;

namespace SefelecTester
{
    public partial class LoadProgram : Form//displays saved tests and lets you
choose one
    {
        public MainWin.SavedTest SelTest;

        public LoadProgram(List<MainWin.SavedTest> tests)
        {
            InitializeComponent();
            var objects = new BindingList<MainWin.SavedTest>();
            foreach (MainWin.SavedTest test in tests){
                objects.Add(test);}//add all saved tests to bindinglist
            comboBox_listload.ValueMember = null;//make combobox
            comboBox_listload.DisplayMember = "autoName";
            comboBox_listload.DataSource = objects;
        }

        private void button_ok_Click(object sender, EventArgs e){
            SelTest = (MainWin.SavedTest)comboBox_listload.SelectedValue;}//load
active value from combobox to SelTest

        private void comboBox_listload_SelectedIndexChanged(object sender,
EventArgs e){
            SelTest = (MainWin.SavedTest)comboBox_listload.SelectedValue;}//load
active value from combobox to SelTest
    }
}
```

Programmi fail 7: MessageBox3.cs

```
using SefelecTester.Properties;
using System;
using System.Drawing;
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public sealed partial class MessageBox3 : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        //look to BarCode.cs for more information
        private readonly bool _exit;
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private bool _keyElapsed = true; //KEYPRESS EVENT

        public MessageBox3(string title, string text, bool exit)//messagebox,
exit=true > conformation window before exiting
        {
            InitializeComponent();
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            _exit = exit;//if conformation dialogue should be shown
            Font = SystemFonts.MessageBoxFont;//select fonts
            label1.Font = SystemFonts.MessageBoxFont;
            Text = title;
            label1.Text = text;
            button_no.Location = new Point(button_no.Location.X, label1.Location.Y
+ label1.Size.Height + 10);//buttons after text
            button_yes.Location = new Point(button_yes.Location.X,
label1.Location.Y + label1.Size.Height + 10);//position is relative, depends on
length of the text
        }

        private void button_cancel_Click(object sender, EventArgs e)
        {
            if (!_exit) return;
            var exiting = new MessageBoxAlt(Resources.Exit + "?",
Resources.Are_you_sure_you_want_to_exit + "?", false, false);
```

```
            DialogResult exit = exiting.ShowDialog();
            DialogResult = exit == DialogResult.Yes ? DialogResult.No :
DialogResult.None;//if 'No' chosen then cancel exiting
        }

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed){
                if (msg.WParam.ToInt32() == (int)Keys.Enter) return true;}
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 8: MessageBoxAlt.cs

```
using SefelecTester.Properties;
using System;
using System.Drawing;
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public sealed partial class MessageBoxAlt : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        //look to BarCode.cs for more information
        private readonly bool _exit;
        private readonly string _altText;
        private readonly string _altTitle;
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private bool _keyElapsed = true; //KEYPRESS EVENT
```

```
        public MessageBoxAlt(string title, string text, bool exit, bool okcancel,
string altTitle = "", string altText = "")//messagebox, exit=true > conformation
dialogue before exit, okcancel=true > ok&cancel buttons instead of yes&no,
altTitle&altText > alternative text for exit conformation
        {
            InitializeComponent();
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            _exit = exit;
            Font = SystemFonts.MessageBoxFont;//select fonts
            label1.Font = SystemFonts.MessageBoxFont;
            Text = title;
            label1.Text = text;
            button_no.Location = new Point(button_no.Location.X, label1.Location.Y
+ label1.Size.Height + 10);
            button_yes.Location = new Point(button_yes.Location.X,
label1.Location.Y + label1.Size.Height + 10);
            if (okcancel){
                button_yes.Text = Resources.OK;
                button_no.Text = Resources.Cancel;}
            _altTitle = altTitle;
            _altText = altText;
        }


        private void button2_Click(object sender, EventArgs e)
        {
            if (!_exit) return;//no exit dialogue
            string title = Resources.Exit + @"?";
            string text = Resources.Are_you_sure_you_want_to_exit_the_program +
"?";
            if (_altTitle != String.Empty) title = _altTitle;//if exists then use
alternative
            if (_altText != String.Empty) text = _altText;
            var exiting = new MessageBoxAlt(title, text, false, false);
            DialogResult exit = exiting.ShowDialog();
            DialogResult = exit == DialogResult.Yes ? DialogResult.No :
DialogResult.None;//if 'No' chosen then cancel exiting
        }


        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
```

```
        {
            if (!_keyElapsed){
                if (msg.WParam.ToInt32() == (int)Keys.Enter) return true;}
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 9: MessageBoxOk.cs

```
using System.Drawing;
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public sealed partial class MessageBoxOk : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        //look to BarCode.cs for more information
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private bool _keyElapsed = true; //KEYPRESS EVENT

        public MessageBoxOk(string text, string title)
        {
            InitializeComponent();
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            Font = SystemFonts.MessageBoxFont;//select fonts
            label1.Font = SystemFonts.MessageBoxFont;
            Text = title;
            label1.Text = text;
```

```
            button_ok.Location = new Point(button_ok.Location.X, label1.Location.Y
+ label1.Size.Height + 10);//button location set relative to the text
        }

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed){
                if (msg.WParam.ToInt32() == (int)Keys.Enter) return true;}
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 10: OtherOptions.cs

```
using SefelecTester.Properties;
using System;
using System.IO;
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public partial class OtherOptions : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        //look to BarCode.cs for more information
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private bool _keyElapsed = true; //KEYPRESS EVENT

        public OtherOptions()
        {
            InitializeComponent();
```

```
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
        }

        private void button1_Click(object sender, EventArgs e)//delete debug logs
        {
            string path = Directory.GetCurrentDirectory();
            path += @"\debug_info";//debug directory relative to executive
directory
            if (Directory.Exists(@path))
            {//directory exists
                try{
                    Directory.Delete(@path, true);//delete directory
                    Wri.ConWrite(Resources.Deleted_debug_info_folder + path + ")",
1);
                    var tempor = new
MessageBoxOk(Resources.Deleted_debug_info_folder + path + ")", Resources.OK);
                    tempor.ShowDialog();}
                catch (Exception){
                    Wri.ConWrite(Resources.Problem_deleting_debug_info_folder +
path + ")", 3);
                    var tempor = new
MessageBoxOk(Resources.Problem_deleting_debug_info_folder, Resources.__Error);
                    tempor.ShowDialog();}
            }
            else{//no such directory
                Wri.ConWrite(Resources.No_debug_info_folder_found + path + ")",
2);
                var tempor = new MessageBoxOk(Resources.No_debug_info_folder_found
+ path + ")", Resources.No_folder);
                tempor.ShowDialog();}
        }

        private void button2_Click(object sender, EventArgs e)//delete temporary
files
        {//directory exists
            string path = Directory.GetCurrentDirectory();
            path += @"\Products\temp";//temp file directory relative to executive
directory
            if (Directory.Exists(@path))
            {
                try{
```

```
                    Directory.Delete(@path, true);//delete directory
                    Wri.ConWrite(Resources.Deleted_products_temp_folder + path +
")", 1);
                    var tempor = new
MessageBoxOk(Resources.Deleted_products_temp_folder + path + ")", Resources.OK);
                    tempor.ShowDialog();}
                catch (Exception){
                    Wri.ConWrite(Resources.Problem_deleting_products_temp_folder +
path + ")", 3);
                    var tempor = new
MessageBoxOk(Resources.Problem_deleting_products_temp_folder, Resources.__Error);
                    tempor.ShowDialog();}
            }
            else{//no such directory
                Wri.ConWrite(Resources.No_debug_products_temp_folder_found + path
+ ")", 2);
                var tempor = new
MessageBoxOk(Resources.No_debug_products_temp_folder_found + path + ")",
Resources.No_folder);
                tempor.ShowDialog();}
        }

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed){
                if (msg.WParam.ToInt32() == (int)Keys.Enter) return true;}
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 11: PleaseWait.cs

```
using SefelecTester.Properties;
using System;
using System.ComponentModel;
```

```
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public partial class PleaseWait : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        //look to BarCode.cs for more information
        private readonly MainWin _main;
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private readonly BackgroundWorker _worker;
        private bool _keyElapsed = true; //KEYPRESS EVENT

        public PleaseWait(MainWin main, BackgroundWorker worker, string title)
        {
            InitializeComponent();
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            _main = main;
            _worker = worker;
            label2.Text = title;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            _main.PauseWorker(_worker);//send pause request to backgroundworker
            var exiting = new MessageBoxAlt(Resources.Exit + "?",
Resources.Are_you_sure_you_want_to_exit_the_program + @"?", false, false);
            DialogResult exit = exiting.ShowDialog();//show confirm to exit
dialogue
            if (exit == DialogResult.Yes){
                DialogResult = DialogResult.OK;}
            else{
                DialogResult = DialogResult.None;
                _main.StartWorker(_worker);}//if not exiting then send request to
resume backgroundworker
        }
```

```
        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed){
                if (msg.WParam.ToInt32() == (int)Keys.Enter) return true;}
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 12: RequestListener.cs

```
using System;

namespace SefelecTester
{ //classes to send data with event
    public delegate void RequestHandler(object o, RequestArgs e);

    public struct Combine
    {
        public int Select;//indicates to which listbox to add
        public string Text;//what to add
    }

    public class RequestArgs : EventArgs
    {
        public int Select;
        public string Text;

        public RequestArgs(Combine temp)
        {
            Text = temp.Text;
            Select = temp.Select;
        }
```

```
    }

    public class RequestListener
    {
        private readonly MainWin _main;//create instance of main window too use
its functions

        public RequestListener(MainWin main){_main = main;}

        public void ProcessRequest(object o, RequestArgs e)
        {
            switch (e.Select)//redirect to selected output
            {
                case 1:
                    _main.LogToOutput(e.Text);//execute Mainwin class method
                    break;
                case 2:
                    _main.LogToError(e.Text);//execute Mainwin class method
                    break;
                case 3:
                    _main.LogToOutput(e.Text);
                    _main.LogToError(e.Text);
                    break;
            }
        }
    } //--classes to send data with event
}
```

Programmi fail 13: StartupSettings.cs

```
using SefelecTester.Properties;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Management;
using System.Timers;
using System.Windows.Forms;
```

```
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public partial class StartupSettings : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        //look to BarCode.cs for more information
        private readonly List<string> _description = new List<string>();
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        public int ControllerNumber = 999;
        private const string SefelecVID = "VID_10C4";
        private bool _keyElapsed = true; //KEYPRESS EVENT
        public string SelectedCom;
        public string SelectedProductPath;
        private SerialPort _serialPort;
        public bool Zerocheck;

        public StartupSettings()
        {
            InitializeComponent();
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            GetAvailable();
            DropDownListMod();
            GetProductList();
        }

        private void DropDownListMod()
        {
            buttonOK.Enabled = SelectSerial.SelectedItem != null;//OK button
enabled if atleast 1 item in list
            int n = 0;
            foreach (string str in _description)
            {//automatically select item with Sefelec VID
                if (str.Contains(SefelecVID)){
                    SelectSerial.SelectedIndex = n;
                    break;}
                n++;
            }
```

```
        }

        private void Startup_Settings_Load(object sender, EventArgs e)
        {
            if (SelectSerial.Items.Count == 0 || SelectProduct.Items.Count == 0)
            {//if no COM devices connected or no parameter files present
                string noItems = "\n" + Resources.Please + ":";
                if (SelectSerial.Items.Count == 0)
                    noItems += "\n" + Resources.Connect_a_COM_device;
                if (SelectProduct.Items.Count == 0)
                    noItems += "\n" +
Resources.Add_a_testing_program_to_the_rigth_folder;
                noItems += "\n\n" + Resources.Then_press_reload;
                var noItem = new MessageBoxOk(Resources.There_is_no_item_to_select
+ noItems, Resources._0ERROR);
                noItem.ShowDialog();//ask to add missing items
            }
            else{//select first tiems
                SelectProduct.SelectedIndex = 0;
                SelectSerial.SelectedIndex = 0;}
        }

        private void GetAvailable() //Get available choices
        {
            SelectSerial.Items.Clear();//clear list
            try
            {
                var searcher = new ManagementObjectSearcher("root\\WMI", "SELECT *
FROM MSSerial_PortName");//use WMI function
                foreach (var queryObj in
searcher.Get().Cast<ManagementObject>())//get all serial ports
                {
                    _serialPort = new SerialPort(queryObj["PortName"].ToString(),
115200, Parity.None, 8, StopBits.One);//try port
                    Wri.ConWrite("InstanceName: " + queryObj["InstanceName"]);
                    Wri.ConWrite("PortName: " + queryObj["PortName"]);
                    try
                    {
                        _serialPort.Open();
                        if
(queryObj["InstanceName"].ToString().Contains(SefelecVID)){//add to name if
Sefelec product
```

```
                            SelectSerial.Items.Add(queryObj["PortName"] + " <=");
                            _description.Add(queryObj["InstanceName"] + " <=" +
Resources.Sefelec_product);}
                        else{
                            SelectSerial.Items.Add(queryObj["PortName"]);

_description.Add(queryObj["InstanceName"].ToString());}
                        _serialPort.Close();
                    }
                    catch (Exception ex)
                    {
                        Wri.ConWrite(Resources.Listing_ports + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                        var tempor = new MessageBoxOk(Resources.Serial_port + " ("
+ queryObj["PortName"] + ") " + Resources._is_in_use + ". " +
Resources.You_cannot_select_this + ". " + Resources._0ERROR + ": " + ex,
Resources._0ERROR + " (" + Resources.with_COM_port + ")");
                        tempor.ShowDialog();
                    }
                }
            }
            catch (ManagementException ex)
            {
                Wri.ConWrite(Resources.Querying_for_WMI_data + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                var tempor = new
MessageBoxOk(Resources.An_error_occurred_while_querying_for_WMI_data + ": " +
ex.Message, Resources.Is_a_device_connected + "?");
                tempor.ShowDialog();
            }
        }


        private void GetProductList()
        {
            SelectProduct.Items.Clear();//clear list
            string folder =
Path.GetDirectoryName(Process.GetCurrentProcess().MainModule.FileName) +
@"\Products\";
            const string filter = "*.csv";
            string[] files = Directory.GetFiles(folder, filter);//get all .csv
files in folder
            foreach (string path in files){
                SelectProduct.Items.Add(Path.GetFileNameWithoutExtension(path) + "
@" + path);}//show file name first
```

```
        } //--Get Available Choices


        private void SelectSerial_SelectedIndexChanged(object sender, EventArgs e)
//  Get Changes
        {
            SelectedCom = SelectSerial.Text.Split(' ')[0];
            SerialPortName.Text = _description[SelectSerial.SelectedIndex];
            DropDownListMod();
        }


        private void tooteValik_SelectedIndexChanged(object sender, EventArgs e)
        {
            string[] words = SelectProduct.Text.Split('@');//select new item if
active index changed
            SelectedProductPath = words[words.Count() - 1];
            Wri.ConWrite(Resources.Selected_product + ": " + SelectedProductPath);
        } //--Get Changes


        private void Startup_Settings_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13){//enter key selects 'OK'
                buttonOK.Select();
                buttonOK.PerformClick();}
            else if (e.KeyChar == (char)27){//escape key selects 'Cancel'
                buttonCancel.Select();
                buttonCancel.PerformClick();}
        }


        private void numericUpDown1_ValueChanged(object sender, EventArgs
e){//controller number changed
            ControllerNumber = (int)numericUpDown1.Value;}


        private void buttonCancel_Click(object sender, EventArgs e){
            var exiting = new MessageBoxAlt(Resources.Exit + "?",
Resources.Are_you_sure_you_want_to_close_the_window + "?", false, false);
            DialogResult exit = exiting.ShowDialog();
            DialogResult = exit == DialogResult.Yes ? DialogResult.Cancel :
DialogResult.None;}//if 'No' chosen then cancel exiting


        private void checkBox_zero_CheckedChanged(object sender, EventArgs
e)//checkbox to choose Zerocheck changed
```

```
        {
            if (checkBox_zero.Checked){
                checkBox_zero.Text = Resources.Zerocheck_will_be_performed;
                Zerocheck = true;}
            else{
                checkBox_zero.Text = Resources.Zerocheck_is_off;
                Zerocheck = false;}
        }

        private void button_reload_Click(object sender, EventArgs e)
        {//get lists again
            GetAvailable();
            DropDownListMod();
            GetProductList();
            SelectProduct.SelectedIndex = 0;
            SelectSerial.SelectedIndex = 0;
        }

        private void button_other_Click(object sender, EventArgs e){//open window
with other options
            var other = new OtherOptions();
            other.Show();}

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed){
                if (msg.WParam.ToInt32() == (int)Keys.Enter) return true;}
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 14: TurnOff.cs

```
using SefelecTester.Properties;
using System;
using System.Drawing;
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester
{
    public sealed partial class TurnOff : Form
    {//to turn off the test in the tester
        private readonly MainWin _main;
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private readonly System.Windows.Forms.Timer _offTimer = new
System.Windows.Forms.Timer(); //KEYPRESS EVENT
        private bool _keyElapsed = true;

        public TurnOff(MainWin main, string text, string title)
        {
            //KEYPRESS EVENT commands are used to ignore enter key from barcode
reader
            //look to BarCode.cs for more information
            InitializeComponent();
            Shown += TurnOff_Shown;
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            _offTimer.Enabled = true;
            _offTimer.Interval = 500;
            _offTimer.Tick += offTimer_Tick;
            Font = SystemFonts.MessageBoxFont;//select fonts
            label1.Font = SystemFonts.MessageBoxFont;
            _main = main;
            Text = title;
            label1.Text = text;
            button_ok.Location = new Point(button_ok.Location.X, label1.Location.Y
+ label1.Size.Height + 10);
        }

        private void TurnOff_Shown(object sender, EventArgs e){//window opens
```

```
            _offTimer.Enabled = true;}//timer starts

        private void offTimer_Tick(object sender, EventArgs e)
        {
            if (!_main.SendDataAndReadEncased(@"FUNC:TEST OFF", @"FUNC:TEST?",
@"TEST OFF")) return;//if test not ended
            Wri.ConWrite(Resources.Test_successfully_turned_off);//otherwise
            _offTimer.Dispose();//dispose of timer
            DialogResult = DialogResult.OK;
            Dispose();//dispose of window
        }

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//KEYPRESS EVENT
        {
            if (!_keyElapsed){
                if (msg.WParam.ToInt32() == (int)Keys.Enter) return true;}
            _keyElapsed = false;
            _keyTimer.Stop();
            _keyTimer.Start();
            return base.ProcessCmdKey(ref msg, keyData);
        }

        private void keyTimer_Elapsed(object sender, ElapsedEventArgs e){
            _keyElapsed = true;} //--KEYPRESS EVENT
    }
}
```

Programmi fail 15: WriteConsole.cs

```
using System;

namespace SefelecTester
{
    public class Wri//class to create events that are sent functions in main form
    {//used to send strings to listboxes from all classes
        public static event RequestHandler RequestEvent;

        public static void Wri_m(MainWin main)
        {//writing to selected output will be using this method
```

```
            var dbsl = new RequestListener(main);//make RequestListener class
instance using constructor
            RequestEvent += dbsl.ProcessRequest;//add method to event, now
requesting event executes the method
        }

        public static void OnRequest(RequestArgs e)
        {
            if (RequestEvent != null)//there is a method to execute
                RequestEvent(new object(), e);//send request, other class is
listening to it
        }

        public static void ConWrite(string text, int select = 0)
        {
            Console.WriteLine(text);//write text to Debug output
            var send = new Combine { Text = text, Select = select };
            var e1 = new RequestArgs(send);//parameters to struct
            OnRequest(e1);//start request
        }
    }
}
```

L7.2. Programm testri juhtimiseks rakisega (vaid erinevad osad)

Selles programmis on puudu LoadProgram.cs (Programmi fail 6: LoadProgram.cs, lk 149). Lisatud on SafeSerialPort.cs (Programmi fail 18: SafeSerialPort.cs, lk 197).

Erinevusi on klassides:

- BarCode.cs (Programmi fail 17: BarCode.cs, lk 196).
- StartupSettings.cs (Programmi fail 19: StartupSettings.cs, lk 199).
- MainWin.cs, millest välja jäetud osad, mis on samad (Programmi fail 16: MainWin.cs, lk 168).

Kõigil samasugustel osadel on kaks erinevust, mis tingitud nimeruumi („namespace“) erinevusest.

Tabel 7.1. Programmide "namespace" erinevused.

SefelecTester	**SefelecTester_DDC**
namespace SefelecTester	namespace SefelecTester_DDC
using SefelecTester.Properties	using SefelecTester_DDC.Properties

Programmi fail 16: MainWin.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Globalization;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Text;
using System.Threading;
using System.Timers;
using System.Windows.Forms;
using SefelecTester_DDC.Properties;
using Timer = System.Timers.Timer;

namespace SefelecTester_DDC
{
    public partial class MainWin : Form
    {
        private const int AutoTestNoMin = 99;
        private const string Line1 =
"ACW;blank;0.1;50;0;42;0;1;1;OFF;0;ON;debug;0000";//parameter strings to default
the program
        private const string Line2 =
"DCW;blank;0.1;#;0;11;0;1;1;OFF;0;ON;debug;0000";
        private const string Line3 =
"IR;blank;0.1;#;1;9999;0;1;1;OFF;#;OFF;debug;0000";
        private const string Line4 =
"GB;blank;3;50;0;650;0;1;#;OFF;0;OFF;debug;0000";
        private const int ManualTestNoMin = 99;
        private const int ParamsMain = 13;
        private readonly char[] _avrState = {'0', '0', '0', '0', '0', '0', '0',
'0'};
        private readonly ManualResetEvent _busy1 = new
ManualResetEvent(false);//events to reset backgroundworkers
```

```
        private readonly ManualResetEvent _busy2 = new ManualResetEvent(false);
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private readonly Timer _myTimer = new Timer();//timer for test
        private readonly string[] _ports = {"pr0", "hv1", "hv2", "hv3", "hv4",
"lr5", "lg6", "ly7", "a3"};//AVR port names
        public string StartTime = null;
        private int _autoTestNo = 100;
        private Cancelling _cancel;
        private int _controllerNo;
        private SavedTest _currentTest;
        private bool _elapsed;
        private bool _keyElapsed = true; //--KEYPRESS EVENT
        private int _manualTestNo = 100;
        private bool _pause;
        private TestList[] _programList;
        private string _scannedSerialNumber = "00000";
        private string _selectedProductPath;
        private string _serialCom;//Sefelec tester COM port
        private string _serialComAvr;//AVR controller COM port
        private string _serialNumber = "00000";
        private List<string> _startedtest;
        private int _testNo;
        private TextWriter _writer; //for TextBoxWriter

        public MainWin()
        {
            InitializeComponent();
            Wri.Wri_m(this);
            FormClosing += Form1_FormClosing;
        }

        private void MainWin_Load(object sender, EventArgs e) //load to form
        {
            //KEYPRESS EVENT commands are used to ignore enter key from barcode
reader
            //look to BarCode.cs for more information
            Show(); //show main window
            _writer = new ListBoxWriter(DebugWin); // Instantiate the writer
            Console.SetOut(_writer); // Redirect the out Console stream
            KeyPreview = true; //KEYPRESS EVENT
```

```
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            worker_program.ProgressChanged += worker_program_ProgressChanged;//to
report backgroundworker progress
            worker_program.RunWorkerCompleted +=
worker_program_RunWorkerCompleted;//runs when backgroundworker completed
            worker_test.ProgressChanged += worker_test_ProgressChanged;
            worker_test.RunWorkerCompleted += worker_test_RunWorkerCompleted;
            OutputTabs.SelectedIndex = 0;//choose first output tab
            if (Debugger.IsAttached){//advanced options when debugging
                OutputTabs.SelectedIndex = 2;//show debug window
                checkBox_advanced.Enabled = true;
                checkBox_advanced.Show();//show advanced options
                labelavr.Show();}//show advanced options for AVR
            Wri.ConWrite(Resources.Program_is_starting, 1);
            StartTime = GetDateTimetoFilename();
            button_setup_Click(null, null);
        }

        private void Form1_FormClosing(Object sender, FormClosingEventArgs e)
//##same as SefelecTester/MainWin.cs

        private void ExitProgram()
        {
            TreeNode[] node = treeView1.Nodes.Find(Resources.Write_log,
true);//find nodes with 'Write Log' name
            if (node.Any()) {
                SetNodeImage(treeView1.Nodes[0].Nodes[2], 1);} //log node to in
progress
            CancelWorker1();//cancel both backgroundworkers
            CancelWorker2();
            try{
                worker_program.Dispose();//dispose of both backgroundworkers
                worker_test.Dispose();}
            catch (Exception ex){
                Wri.ConWrite(Resources.Exception_raised + " (" + ex.Message + "):
" + ex);
                WriteDebugAfterEx();}
            try{
                if (serialPort1.IsOpen) serialPort1.Close();//close serial ports
                if (serialPort_avr.IsOpen) serialPort_avr.Close();}
            catch (IOException) {}
```

```
            CloseForms();
            Wri.ConWrite(Resources.Writing_logs);
            WriteLogCsv("TESTING FINISHED");
            Wri.ConWrite(Resources.Program_Exit, 1);
            WriteDebug();
            if (node.Any()) {
                SetNodeImage(treeView1.Nodes[0].Nodes[2], 3);}//log node to
completed
            this.SynchronizedInvoke(Dispose);//dispose of window
            Application.Exit();//exit application
        }

        private void CloseForms() //##same as SefelecTester/MainWin.cs

        private bool OpenPopup()
        {
            while (worker_program.IsBusy) CancelWorker1();
            if (serialPort1.IsOpen) serialPort1.Close();//close serial port
            if (serialPort_avr.IsOpen) serialPort_avr.Close();
            var popup = new StartupSettings(); //start StartupSettings form
            DialogResult dialogresult = popup.ShowDialog();
            if (dialogresult == DialogResult.OK) //get dialogue result from popup
form
            {
                label2.Text = Resources.Programming_started;
                _controllerNo = popup.ControllerNumber;//get controller number
form StartupSettings window
                Wri.ConWrite(Resources.Controller_number_is + ": " +
_controllerNo, 1);
                _serialCom = popup.SelectedCom;//get serial port number form
StartupSettings window
                _serialComAvr = popup.SelectedComAvr;//get parameters file path
form StartupSettings window
                _selectedProductPath = popup.SelectedProductPath;//get parameters
file path form StartupSettings window
                int controller = popup.ControllerNumber;
                if (!OpenComPort_avr(popup.SelectedComAvr, true)) ExitProgram();
                _startedtest = new List<string> {"STARTED TESTING;TESTER ID:;" +
controller};
                string[] toPass = { popup.SelectedCom, popup.SelectedProductPath,
popup.SelectedComAvr };//string to send to backgroundworker
                try {
```

```
                    worker_program.RunWorkerAsync(toPass); } //this will run all
Transmitting protocol coding at background thread
                catch (Exception ex)
                {
                    Wri.ConWrite(Resources.Problem_starting + " <worker_program>"
+ ". " + Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                    WriteDebugAfterEx();
                    var tempor = new MessageBoxOk(Resources.Problem_starting + "
<worker_program>", Resources._0ERROR);
                    tempor.ShowDialog();
                }
                popup.Dispose(); //close popup form
            }
            else if (dialogresult == DialogResult.Cancel) {
                return false; }
            return true;
        }

        private void SetupTesting() //##same as SefelecTester/MainWin.cs

        private void StartTesting() //##same as SefelecTester/MainWin.cs

        public bool OpenComPort(string com) //##same as SefelecTester/MainWin.cs

        public bool OpenComPort_avr(string com, bool zero = false)
        {//open serial port for AVR controller
            if (!serialPort_avr.IsOpen)
            {//serial port not open already
                try{
                    serialPort_avr = new SafeSerialPort(com, 9600, Parity.None, 8,
StopBits.One){//SafeSerialPort- otherwise AVR has issues if serial port not closed
                        PortName = com,
                        BaudRate = 9600};
                    serialPort_avr.Open();}//open serial port
                catch (Exception ex){
                    Wri.ConWrite(Resources.Problem_opening_COM_avr_port + ". " +
Resources.Exeption_raised + " (" + ex.Message + "): " + ex, 2);
                    WriteDebugAfterEx();
                    var tempor = new
MessageBoxOk(Resources._0APPLICATION_0WILL_0EXIT + ". " + Resources._0ERROR + ": "
+ ex, Resources._0APPLICATION_0WILL_0EXIT + ". " + Resources._0ERROR + " (" +
Resources._with_COM_port + ")");
```

```
                    tempor.ShowDialog();
                    ExitProgram();}
            }
            else
            { Wri.ConWrite(Resources.Serial_avr_port_already_open); }
            if (serialPort_avr.IsOpen)//serial port already open
            {
                if (!zero) return true;//AVR default port states not requested
                Wri.ConWrite(Resources.Serial_avr_port + " " + com + " " +
Resources._connected, 1);
                if (!SetAvrPorts("00000000")){//try to set AVR ports to low
                    Wri.ConWrite(Resources.Cannot_zero_avr_ports, 2);
                    return false;}
                Wri.ConWrite(Resources._avr_ports_zeroed);
                return true;
            }
            Wri.ConWrite(Resources.Cannot_connect_to_serial_avr_port + ": " + com,
2);
            return false;
        }

        private bool SetAvrPorts(string states)
        {
            if (states.Length != 8 && states.Length != 4){//check if the string
stating states has wrong length
                Wri.ConWrite(Resources.Command_to_set_AVR_ports_has_wrong_length +
": " + states, 2);
                return false;}
            int i = 0;
            if (states.Length == 4) i = 1;//sets only ports 2-5 (HV relays)
            foreach (char a in states)
            {
                if (a == 'x') continue;//char to skip port
                bool result = true;
                if (a == '0') { if (!SetOnePort(i, false)) result = false; }
                else if (a == '1') { if (!SetOnePort(i, true)) result = false; }
                else{

Wri.ConWrite(Resources.Command_to_set_AVR_ports_has_wrong_character, 2);
                    return false;}
                if (!result){
```

```
Wri.ConWrite(Resources.Cannot_send_one_port_data_to_serial_avr_port, 2);
                    return false;}
                i++;
            }
            Wri.ConWrite(Resources._avr_ports_set_to + ": " + states);
            return true;
        }

        private bool SetOnePort(int port, bool state)
        { //ss - set port, gg - get port state
            if (port > 8 || port < 0) return false;//check if port number is in
the right range
            DialogResult dial;
            var retry = new MessageBoxAlt(Resources.Problem_writing_AVR_port,
Resources.Cannot_set_AVR_port + "\n" + Resources.Would_you_like_to_try_again +
"?\n" + Resources.Press__no__to_exit_the_program, true, false);//set up messagebox
to be used later if nessecary
            do
            {
                try
                {
                    for (int i = 0; i < 3; i++)//try up to 3 times
                    {
                        if (serialPort_avr.IsOpen)
                        {
                            EmptyIncomingBuffer(serialPort_avr);
                            string toSend = " ss " + _ports[port];//set port
command + port name
                            if (state) toSend += " 1 ";
                            else toSend += " 0 ";//+ add state
                            serialPort_avr.Write(toSend + "\r\n");//send set port
command
                            string incoming =
GetIncomingData(serialPort_avr);//read answer
                            try {
                                if (incoming.Split(',')[0] == "98") { //no data
received error code

Wri.ConWrite(Resources.No_data_received_when_setting_avr_port, 2); } }
                            catch (Exception ex) {
```

```
Wri.ConWrite(Resources.Data_received_when_setting_avr_port_was_not_normal + ". " +
Resources.Exception_raised + ": " + ex.Message);}
                            string expected = toSend.Replace("ss", "new");//answer
returns in: "new portname state"
                            if (incoming.Trim() == expected.Trim()){
                                Wri.ConWrite(Resources._avr_port + " " + port + "
" + Resources._set_successfully_to + " " + state);
                                if (state) _avrState[port] = '1';
                                else _avrState[port] = '0';
                                Wri.ConWrite("Avr state: " + string.Join(",",
_avrState));
                                this.SynchronizedInvoke(() => labelavr.Text = new
string(_avrState));
                                return true;}
                            Wri.ConWrite(Resources.Setting_avr_ports_failed + ". "
+ Resources.Expected + ":" + expected + " " + Resources.Received + ":" +
incoming); //'na' - no port with that name. 'unk' - wrong on/off param
                            continue;
                        }
                        Wri.ConWrite(Resources.Cannot_send_data + ", " +
Resources._not_connected_to_serial_avr_port, 2);
                    }
                }
                catch (Exception ex){
                    if (ex is IOException || ex is InvalidOperationException)
Wri.ConWrite(Resources.Error_writing_to_avr, 2);
                    else Wri.ConWrite(Resources.Unknown_error_with_avr, 2);
                    Wri.ConWrite(ex.Message + ": " + ex);
                    ResetAvr();
                    dial = DialogResult.Yes;
                    continue;}
                //if no exit then sth failed
                dial = retry.ShowDialog();//ask if to try again
                if (dial == DialogResult.No) ExitProgram();
                else{
                    ResetAvr(); }
            }
            while (dial == DialogResult.Yes);
            return false;
        }
```

```
        private void ResetAvr()//sometimes VirtualSerial in AVR becomes
inaccessible
        {//asks to reset connection to AVR controller
            var temp = new string(_avrState);
            try{//try ro close port
                serialPort_avr.Close();

Wri.ConWrite(Resources.MainWin_ResetAvr_AVR_device_successfully_closed);}
            catch (Exception) {}
            try{//try to dispose of port
                serialPort_avr.Dispose();

Wri.ConWrite(Resources.MainWin_ResetAvr_AVR_device_successfully_disposed);}
            catch (Exception) {}
            var ok = new MessageBoxAlt(Resources.Unplug_and_replug_AVR,
Resources.Unplug_AVR_device + ", " +
Resources._then_plug_it_back_in_and_press__OK_ + "\n\n" +
Resources.To_exit_the_program_press__Cancel_, true, true);
            DialogResult res = ok.ShowDialog();//ask to unplug and plug in AVR
controller
            if (res == DialogResult.No) ExitProgram();
            try{
                OpenComPort_avr(_serialComAvr);//try to open port
                SetAvrPorts(new string(_avrState));//try to set all AVR ports low
                Wri.ConWrite(Resources.MainWin_ResetAvr_AVR_device_reset_success +
". " + Resources.MainWin_ResetAvr_Ports_restored_to + ": " + new
string(_avrState));}
            catch (Exception e){
                Wri.ConWrite(e.Message + ": " + e);
                var exiting = new MessageBoxOk(Resources.Cannot_send_data_to_AVR +
"\n" + Resources.It_seems_to_be_disconnected + "\n" +
Resources.The_program_will_now_exit + "\n" +
Resources.Plug_out_and_in_the_AVR_cable_before_starting_the_program_again,
Resources._0AVR_not_connected);
                exiting.ShowDialog();
                ExitProgram();}
        }


        private void buttonTemp_Click(object sender, EventArgs e) //##same as
SefelecTester/MainWin.cs


        private void RemoveEmptyLines(string path1, string path2) //##same as
SefelecTester/MainWin.cs
```

```
        public void WriteDebug() //##same as SefelecTester/MainWin.cs

        public void WriteDebugAfterEx() //##same as SefelecTester/MainWin.cs

        private void OnTimer(Object source, ElapsedEventArgs e) //##same as
SefelecTester/MainWin.cs

        private static string IntTo3String(int no) //##same as
SefelecTester/MainWin.cs

        private static bool CancelPending(BackgroundWorker worker) //##same as
SefelecTester/MainWin.cs

        private void WriteLog(string text) //##same as SefelecTester/MainWin.cs

        private void WriteLogCsv(string text) //##same as SefelecTester/MainWin.cs

        private static string GetDateTimetoFilename() //##same as
SefelecTester/MainWin.cs

        private static string GetDateTime() //##same as SefelecTester/MainWin.cs

        private void checkBox1_CheckedChanged(object sender, EventArgs e) //##same
as SefelecTester/MainWin.cs

        private void SetNodeImage(TreeNode node, int imag) //##same as
SefelecTester/MainWin.cs

        private void SetImage(int imag) //##same as SefelecTester/MainWin.cs

        private static List<TreeNode> ReturnChildNodes(TreeNode parentNode)
//##same as SefelecTester/MainWin.cs

        private void SerialNo_ValueChanged(object sender, EventArgs e) //##same as
SefelecTester/MainWin.cs

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//##same as SefelecTester/MainWin.cs //KEYPRESS EVENT

        private void button_setup_Click(object sender, EventArgs e) //##same as
SefelecTester/MainWin.cs
```

```
        private void button_cancel_Click(object sender, EventArgs e) //##same as
SefelecTester/MainWin.cs


        private void StopTest() //##same as SefelecTester/MainWin.cs


        public void LogToOutput(string text) //##same as SefelecTester/MainWin.cs


        public void LogToError(string text) //##same as SefelecTester/MainWin.cs


        private void SetElements(bool state)
        {//set all elements to selected state
            this.SynchronizedInvoke(() =>{
                button_setup.Enabled = state;
                buttonTemp.Enabled = state;
                checkBox1.Enabled = state;
                SerialNo.Enabled = state;});//used to disable buttons and other
inouts while doing work
        }


        private void checkBox_advanced_CheckedChanged(object sender, EventArgs e)
//##same as SefelecTester/MainWin.cs


        #region Programming Device


        private void worker_program_DoWork(object sender, DoWorkEventArgs e) //
PROGRAMMING DEVICE
        {
            _manualTestNo = ManualTestNoMin;
            _autoTestNo = AutoTestNoMin;
            _programList = null;
            _currentTest.TestIsProgrammed = false;
            SetElements(false);
            StartWorker(worker_program);
            this.SynchronizedInvoke(() => treeView1.Nodes.Clear());
            bool check = true;
            var sent = (string[]) e.Argument;
            bool state;
            string autoName = null;
            var toTest = new List<TestList>();
            var toTree = new List<string>();
            if (sent.Count() > 1)//if parameters not sent then skip testing
```

```
            {
                const int de = 2;
                if (!OpenComPort(sent[0])){//if com port can't be opened
                    Wri.ConWrite(Resources.Cannot_acces_COM_port_for_programming +
". " + Resources.Exiting + ".", 2);
                    CancelWorker1();//cancel backgroundworker
                    return;}
                string newPath = Path.GetDirectoryName(sent[1]);
                newPath += @"\temp\";//save new file to subdirectory
                if (!Directory.Exists(newPath)){//create dir if not existing
                    Directory.CreateDirectory(newPath);}
                newPath += Path.GetFileName(sent[1]);
                newPath = newPath.Replace(".csv",
"_removed_empty_lines.csv");//append to filename
                RemoveEmptyLines(_selectedProductPath, newPath);
                _selectedProductPath = newPath; //empty lines removed
                string readFile = ReadFile(_selectedProductPath);//whole file to
string
                string[] readFile2Split = readFile.Split('\n');//split to lines
                autoName = readFile2Split[0].Split(';')[0]; //get AUTO program
name
                int max = readFile2Split.Count();//count lines
                string tests = null;
                var testLines = new List<string> {autoName};
                foreach (string line in readFile2Split.Where(line =>
line.Split(';').Count() >= ParamsMain)){//add tests to list
                    toTree.Add(line.Split(';')[1]);//test name to add to treeview
                    testLines.Add(line);
                    tests += line;
                    tests += "\n";}
                FillTreeView(toTree.ToArray());
                try{
                    this.SynchronizedInvoke(() => treeView1.Nodes[0].Text =
autoName);}//change treeview first node name
                catch (Exception ex){
                    Wri.ConWrite(Resources.Changing_main_node_name + ". " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);
                    var tempor = new MessageBoxOk(Resources.Exeption_raised + " ("
+ Resources._changing_main_node_name + "): " + ex, Resources._0ERROR);
                    tempor.ShowDialog();}
                foreach (TreeNode thisNode in
ReturnChildNodes(treeView1.Nodes[0])){
```

```
                    SetNodeImage(thisNode, 0);}
                Invoke(new MethodInvoker(() => treeView1.ExpandAll()));
                var testsList = new MessageBoxAlt(Resources.List_of_tests,
Resources.List_of_tests_in_the_file_is + ":\n" + tests + "\n" +
Resources.Press__OK__if_this_is_correct + ".\n" +
Resources.Press__Cancel__to_exit_the_program_and_correct_mistakes, true, true);
                DialogResult dialogRes = testsList.ShowDialog();//asks if tests in
the list are ok
                if (dialogRes == DialogResult.No){//if not then exit
                    ExitProgram();}
                worker_program.ReportProgress(100/(max + de));//report progress
                Thread.Sleep(15);//wait
                SendData("FUNC:TEST OFF");//to tester: test off, just in case its
running
                if (!SendDataAndReadEncased("MAIN:FUNC AUTO", "MAIN:FUNC?", "AUTO
MODE")){//change mode to auto and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto mode", 2);}
                if (!SendDataAndReadEncased("AUTO:STEP " + _autoTestNo,
"AUTO:STEP?", IntTo3String(_autoTestNo))){//change auto test number and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto step", 2);}
                string delErr;
                do
                {//delete test list from automatic test
                    _busy1.WaitOne();
                    Thread.Sleep(50);
                    SendData("AUTO:PAGE:DEL 1");
                    delErr = GetError()[0];
                } while (delErr == "0");//do while no error deleting
                SendData("*CLS\n"); //deleting a test that is not there gives a
error
                if (!SendDataAndReadEncased("MAIN:FUNC MANU", "MAIN:FUNC?", "MANU
MODE")){//change mode to manual and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " manu mode", 2);}
                worker_program.ReportProgress(200/(max + de));
                Thread.Sleep(15);
                e.Cancel = CancelPending(worker_program);
                if (e.Cancel){//check if cancel request is active
                    state = false;
                    return;}
```

```
                int lineN = 0;
                SetNodeImage(treeView1.Nodes[0].Nodes[0], 2);//treeview
programming node to in progress
                foreach (string line in testLines)
                {
                    if (lineN == 0){
                        lineN++;
                        continue;}//skip first line
                    if (line.Trim() == "end"){//if end is before last line
                        Wri.ConWrite(Resources.End_of_file_detected);
                        break;}
                    if (line.Split(';').Count() < ParamsMain){//if not enough
parameters
                        Wri.ConWrite(Resources.Line_has_wrong_length + ": " +
line);
                        continue;}//skip line
                    _busy1.WaitOne();//pause request check
                    e.Cancel = CancelPending(worker_program);
                    if (e.Cancel){
                        state = false;
                        return;}
                    worker_program.ReportProgress((int) ((lineN + de -
0.5)*100/(max + de))); //reports a percentage between 0 and 100
                    Thread.Sleep(15);
                    try
                    {
                        WriteDebugAfterEx();
                        SetNodeImage(treeView1.Nodes[0].Nodes[0].Nodes[lineN - 1],
2); //set to inprogress (manual test)
                        if (!SendDataAndReadEncased("MANU:STEP " +
IntTo3String(_manualTestNo - lineN), "MANU:STEP?", IntTo3String(_manualTestNo -
lineN))){//change manual program number and check
                            check = false;
                            Wri.ConWrite(Resources.Error_setting + " manu step",
2);}
                        int testTime;
                        int success = ProgramManual(line, "0", false, out
testTime);//program manual test according to 1 line
                        if (success == 0)
                        {
                            Wri.ConWrite(Resources._0ERROR + ". " +
Resources.Couldn_t_program_no_ + (_manualTestNo - lineN) + "::" + line + ". " +
Resources.Tring_again + ".");
```

```
                            switch (line.Split(';')[0])
                            {//program with default program
                                case "ACW":
                                    ProgramManual(Line1, "0", false, out
testTime);
                                    break;
                                case "DCW":
                                    ProgramManual(Line2, "0", false, out
testTime);
                                    break;
                                case "IR":
                                    ProgramManual(Line3, "0", false, out
testTime);
                                    break;
                                case "GB":
                                    ProgramManual(Line4, "0", false, out
testTime);
                                    break;
                            }//this sets parameters to low value so that maximum
ratings of parameter combinations will not be exeeded
                            success = ProgramManual(line, "0", false, out
testTime);//try programming again
                            if (success == 0)
                            {//programming failed
                                Wri.ConWrite(Resources._0ERROR + ". " +
Resources.Couldn_t_program_no_ + (_manualTestNo - lineN) + "::" + line + ". " +
Resources._0FAILED + ".", 2);
                                var mess = new
MessageBoxAlt(Resources.Wrong_program, Resources.The_program_parameters_are_wrong
+ ".\n" + Resources.Program + ":" + line + "\n" +
Resources.Would_you_like_to_skip_this_test_and_only_do_other_tests + "?\n" +
Resources.Choosing_no_will_exit_the_program + ".", true, false);
                                DialogResult dialogResult =
mess.ShowDialog();//ask if line should be removed and not programmed
                                PauseWorker(worker_program);
                                if (dialogResult == DialogResult.Yes)
                                {//remove program line

SetNodeImage(treeView1.Nodes[0].Nodes[0].Nodes[lineN - 1], 1);
                                    try{
                                        Invoke(new MethodInvoker( () =>
treeView1.Nodes[0].Nodes[1].Nodes[lineN - 1].Remove()));}//remove node from
testing
                                    catch (ArgumentOutOfRangeException ex){
```

```
                                    Wri.ConWrite(Resources.Exeption_raised +
"(" + ex.Message + "): " + ex);
                                    WriteDebugAfterEx();}
                                lineN++;
                                StartWorker(worker_program);
                                continue;
                            }
                            if (dialogResult == DialogResult.No)
ExitProgram();
                        }
                    }
                    else if (success == -1){
                        break;}
                    int testNumber = _manualTestNo - lineN;
                    string testParams = GetDataEncased("MANU" + testNumber +
":EDIT:SHOW?");//get test pparameters from tester
                    _startedtest.Add("PARAMETERS:;" + autoName + " (PRODUCT
NAME);" + line.Split(';')[1] +
                                     " (SUBTEST NAME);SUBTEST" + (99 -
testNumber) + ";" + testParams);
                    string instructions = line.Split(';').Count() > ParamsMain
                        ? line.Split(';')[ParamsMain]
                        : Resources._do_as_instructed;//if last parameter is
not added then add default
                    toTest.Add(new TestList{
                        TestNo = (_manualTestNo - lineN),
                        TestingTime = testTime,
                        TestParameters = testParams,
                        TestDescription = instructions,
                        TestName = line.Split(';')[1]});//add parameters to
list
                    SetNodeImage(treeView1.Nodes[0].Nodes[0].Nodes[lineN - 1],
3);//set treeview node to completed
                    Wri.ConWrite(Resources.Added_to_test_list + ": " +
(_manualTestNo - lineN));
                }
                catch (IndexOutOfRangeException ex){
                    Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message
+ "): " + ex, 2);
                    WriteDebugAfterEx();}
                worker_program.ReportProgress((lineN + de)*100/(max + de));
//reports a percentage between 0 and 100
                Thread.Sleep(15);
```

```
                    lineN++;
                }
                if (!SendDataAndReadEncased("MAIN:FUNC AUTO", "MAIN:FUNC?", "AUTO
MODE")){//set tester mode to auto and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto mode", 2);}
                if (!SendDataAndReadEncased("AUTO:STEP " + _autoTestNo,
"AUTO:STEP?", IntTo3String(_autoTestNo))){//set tester auto test number and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto step", 2);}
                if (!SendDataAndReadEncased("AUTO:NAME " + autoName, "AUTO:NAME?",
autoName)){//set tester auto test name and check
                    check = false;
                    Wri.ConWrite(Resources.Error_setting + " auto name", 2);}
                foreach (TestList manu in toTest){//add manual tests to automatic
test
                    Wri.ConWrite(Resources.Add_manual_test_to_auto_test + ": " +
manu.TestNo);
                    SendData("AUTO:EDIT:ADD " + manu.TestNo);}
                _manualTestNo = _manualTestNo - lineN;
                if (!check){
                    Wri.ConWrite(Resources.Some_error_occured_while_programming,
2);
                    var tempor = new
MessageBoxOk(Resources.Some_error_occured_while_programming, Resources._0ERROR);
                    tempor.ShowDialog();}
                SetNodeImage(treeView1.Nodes[0].Nodes[0], !check ? 1 : 3);//set
node image to completed or failed
                foreach (string ll in _startedtest) WriteLogCsv(ll);
                worker_program.ReportProgress(100);
                Thread.Sleep(15);
                state = true;
            }
            else{
                Wri.ConWrite(Resources.Error_with_index_in_programming + "
(<worker_program>)");
                state = false;}
            if (e.Cancel){
                state = false;}
            try{
                ChangeLastHold(toTest.Last().TestNo);}
            catch (InvalidOperationException ex){//no tests in toTest list
```

```
                Wri.ConWrite(Resources.List_of_tests_is_empty + ". " +
Resources.Exeption_raised + "(" + ex.Message + "): " + ex, 2);}
            _programList = toTest.ToArray();
            if (autoName != null)//save if autoname parameter is there
                _currentTest = new SavedTest{
                    TestIsProgrammed = true,
                    AutoTestN = _autoTestNo,
                    AutoName = autoName,
                    TestParams = toTest.ToArray()};
            e.Result = state;
        }

        private void worker_program_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e) //##same as SefelecTester/MainWin.cs

        private void worker_program_ProgressChanged(object sender,
ProgressChangedEventArgs e) //##same as SefelecTester/MainWin.cs

        private int CancelWorker1() //##same as SefelecTester/MainWin.cs

        private int CancelWorker2() //##same as SefelecTester/MainWin.cs

        private string ReadFile(string filePath) //##same as
SefelecTester/MainWin.cs

        private int ProgramManual(string line, string reference, bool referen, out
int testTime) //this sets 1 MANU test
        {
            Wri.ConWrite(Resources.Program_manual_test + ": " + line);
            const int writeTimeout = 50;
            int success = 1;
            testTime = 0;
            string[] separateCommands = line.Split(';');
            if (separateCommands.Any())
            {
                if (separateCommands[0] == "end"){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Wrong_line_in_commands);
                    return -1;}
            }
            if (separateCommands.Count() < 12){
```

```
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Wrong_line_in_commands);
                return 0;}
            string mode = separateCommands[0]; //mode
            string name = separateCommands[1]; //name
            string param1 = separateCommands[2].Replace(",", "."); //volt or curr
            string param2 = separateCommands[3]; //freq
            string param3 = separateCommands[4].Replace(",", "."); //LO set
            string param4 = separateCommands[5].Replace(",", "."); //HI set
            string param5 = separateCommands[6].Replace(",", "."); //ref
            string param6 = separateCommands[7].Replace(",", "."); //test timer
            string param7 = separateCommands[8].Replace(",", "."); //rampup
            string param8 = separateCommands[9]; //ARC mode
            string param9 = separateCommands[10].Replace(",", "."); //ARC limit
            string param10 = separateCommands[11]; //groundmode
            try{
                testTime = (int) (decimal.Parse(param6,
CultureInfo.InvariantCulture) + decimal.Parse(param7,
CultureInfo.InvariantCulture) + 1);}
            catch (FormatException ex)
            {
                Wri.ConWrite(Resources.Exception_raised + " (" + ex.Message + "):
" + ex + "\n" + Resources.Error_converting + " testtime");
                decimal testTime1 = 0;
                decimal testTime2 = 0;
                try { testTime1 = decimal.Parse(param6,
CultureInfo.InvariantCulture); }
                catch (FormatException ex1){
                    Wri.ConWrite(Resources.Exception_raised + " (" + ex1.Message +
"): " + ex1 + "\n" + Resources.Error_converting + " testtime1");}
                try { testTime2 = decimal.Parse(param7,
CultureInfo.InvariantCulture); }
                catch (FormatException ex1){
                    Wri.ConWrite(Resources.Exception_raised + " (" + ex1.Message +
"): " + ex1 + "\n" + Resources.Error_converting + " testtime2");}
                testTime = (int) (testTime1 + testTime2 + 1);
                if (testTime <= 1) success = 0;
            }
            catch (Exception ex){
                Wri.ConWrite(Resources.Exeption_raised + "(" + ex.Message + "): "
+ ex);
                var tempor = new MessageBoxOk(Resources.Exeption_raised + "(" +
Resources._reading_testtime + "): " + ex, Resources._0ERROR);
```

```
                tempor.ShowDialog();
                success = 0;}
            if (!SendDataWithCheck("MANU:EDIT:MODE " + mode)){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._mode + " (" + mode + ") " +
Resources._is_wrong);
                success = 0;}
            Thread.Sleep(writeTimeout);
            SendData("MANU:NAME " + name); //send name. sends only max 10 charac
            Thread.Sleep(writeTimeout);
            if (mode == "GB") //send voltage or current
            {
                if (!SendDataWithCheck("MANU:" + mode + ":CURR " + param1)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._current + " (" + param1 + ") "
+ Resources._is_wrong, 2);
                    success = 0;}
            }
            else
            {
                if (!SendDataWithCheck("MANU:" + mode + ":VOLT " + param1)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._voltage + " (" + param1 + ") "
+ Resources._is_wrong + "", 2);
                    success = 0;}
            } //--send voltage or current
            Thread.Sleep(writeTimeout);
            if (mode == "ACW" || mode == "GB") //send freq
            {
                if (!SendDataWithCheck("MANU:" + mode + ":FREQ " + param2)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._frequency + " (" + param2 + ")
" + Resources._is_wrong, 2);
                    success = 0;}
            } //--send freq
            Thread.Sleep(writeTimeout);
            if (mode == "IR" || mode == "GB") //send HiSet
            {
                if (!SendDataWithCheck("MANU:" + mode + ":RHIS " + param4)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.HiSet + " (" + param4 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
```

```
            }
            else
            {
                if (!SendDataWithCheck("MANU:" + mode + ":CHIS " + param4)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.HiSet + " (" + param4 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
            } //--send HiSet
            Thread.Sleep(writeTimeout);
            if (mode == "IR" || mode == "GB") //send LoSet
            {
                if (!SendDataWithCheck("MANU:" + mode + ":RLOS " + param3)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.LoSet + " (" + param3 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
            }
            else
            {
                if (!SendDataWithCheck("MANU:" + mode + ":CLOS " + param3)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources.LoSet + " (" + param3 + ") " +
Resources._is_wrong, 2);
                    success = 0;}
            } //--send LoSet
            Thread.Sleep(writeTimeout);
            if (!SendDataWithCheck("MANU:" + mode + ":REF " + param5)){ //send ref
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._reference_value + " (" +
param5 + ") " + Resources._is_wrong, 2);
                success = 0;} //--send REF
            Thread.Sleep(writeTimeout);
            if (!SendDataWithCheck("MANU:" + mode + ":TTIM " + param6)){ //send
test time
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._test_time + " (" + param6 + ")
" + Resources._is_wrong, 2);
                success = 0;} //--send test time
            Thread.Sleep(writeTimeout);
            if (mode != "GB") //send ramp up time
            {
                if (!SendDataWithCheck("MANU:RTIM " + param7)){
```

```
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._ramp_up_time + " (" + param7 +
") " + Resources._is_wrong, 2);
                    success = 0;}
            } //--send ramp up time
            Thread.Sleep(writeTimeout);
            if (mode == "ACW" || mode == "DCW") //set arc detection mode
            {
                if (!SendDataWithCheck("MANU:UTIL:ARCM " + param8)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._arc_mode + " (" + param8 + ")
" + Resources._is_wrong, 2);
                    success = 0;}
                Thread.Sleep(writeTimeout);
                if (param8 == "ON") //set arc current
                {
                    if (!SendDataWithCheck("MANU:" + mode + ":ARCC " + param9)){
                        Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._arc_current + " (" + param9 +
") " + Resources._is_wrong, 2);
                        success = 0;}
                } //--set arc current
            } //--set arc detection mode
            Thread.Sleep(writeTimeout);
            if (mode == "ACW" || mode == "DCW") //set groundmode
            {
                if (!SendDataWithCheck("MANU:UTIL:GROUNDMODE " + param10)){
                    Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._ground_mode + " (" + param10 +
") " + Resources._is_wrong, 2);
                    success = 0;}
            } //--set groundmode
            if (!SendDataAndReadEncased("MANU:UTIL:PASS ON", "MANU:UTIL:PASS?",
"ON")){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._pass_hold_cannot_be_set, 2);
                success = 0;}
            if (!SendDataAndReadEncased("MANU:UTIL:FAIL STOP", "MANU:UTIL:FAIL?",
"STOP")){
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._fail_hold_cannot_be_set, 2);
                success = 0;}
            if (!SendDataAndReadEncased("MANU:UTIL:MAXH ON", "MANU:UTIL:MAXH?",
"ON")){
```

```
                Wri.ConWrite(Resources.Error_programming + ". " +
Resources.Program + " (" + name + ") " + Resources._fail_hold_cannot_be_set, 2);
                success = 0;}
            return success;
        }

        //--PROGRAMMING DEVICE--PROGRAMMING DEVICE--PROGRAMMING DEVICE--
PROGRAMMING DEVICE--
        #endregion Programming Device

        #region Testing

        private void worker_test_DoWork(object sender, DoWorkEventArgs e) //##same
as SefelecTester/MainWin.cs

        private void worker_test_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e) //##same as SefelecTester/MainWin.cs

        private void worker_test_ProgressChanged(object sender,
ProgressChangedEventArgs e) //##same as SefelecTester/MainWin.cs

        public void StartWorker(BackgroundWorker worker) //##same as
SefelecTester/MainWin.cs

        public void PauseWorker(BackgroundWorker worker) //##same as
SefelecTester/MainWin.cs

        public int TestWhile() //##same as SefelecTester/MainWin.cs

        public int TestEach(ref List<string> logging, bool lasttestcompleted)
        {
            int progN = 0;
            bool state = true;
            int report = _programList.Count();
            PauseWorker(worker_test);
            var barcode = new BarCode(this, Resources.Do_NOT_press__OK__yet + "\n"
+ Resources.Scan_change_the_barcode_if_needed + "\n" +
Resources.Put_the_product_in_the_testing_box + "\n" +
Resources.Only_then_press__OK_ + "\n\n" +
Resources.Test_will_start_after_pressing__OK_ + "\n" +
Resources.Make_sure_the_product_is_correctly_in_the_box_before_continuing);
            DialogResult barcodeforTest = barcode.ShowDialog();//dialogue for
entering barcode
```

```
            this.SynchronizedInvoke(() => SerialNo.Text =
barcode.textBox_barCode.Text);
            if (String.IsNullOrEmpty(SerialNo.Text)) this.SynchronizedInvoke(() =>
SerialNo.Text = "0");
            if (barcodeforTest == DialogResult.Yes)
            {
                SetOnePort(0, true); //cyllinder to connected position
                SetNodeImage(treeView1.Nodes[0].Nodes[1], 2); //testing node to in
progress
                SetImage(0);
                var connected = new MessageBoxAlt(Resources.Cyllinder_connected +
"?", Resources.Press__OK__after_the_cyllinder_has_connected_the_connector + "\n" +
Resources.Press__Cancel__if_there_is_a_problem, false, true);
                DialogResult answ = connected.ShowDialog();//asks if cylinder is
in place
                if (answ == DialogResult.No){//if not then cancels
                    CancelWorker2();
                    return 5;}
                Wri.ConWrite(Resources.Tested_product + ": " + SerialNo.Text, 1);
                logging.Add("TEST RESULTS;" + _currentTest.AutoName + " (PRODUCT
NAME);" + SerialNo.Text + " (SERIAL NO)");
                StartWorker(worker_test);
            }
            else{//cancel pressed, cancels
                CancelWorker2();
                return 5;}
            foreach (TreeNode node in treeView1.Nodes[0].Nodes[1].Nodes){
                SetNodeImage(node, 0);} //all test nodes to hourglass
            SetOnePort(5, false); //red led off
            SetOnePort(6, false); //green led off
            foreach (TestList test in _programList)
            {//do for all tests
                while (!SetAvrPorts(test.Relays)){//do while seting relay suceeds
                    var retry = new MessageBoxAlt(Resources.Retry + "?",
Resources.Error_setting_relays_to_state + ":\n" + test.Relays, true, false);
                    DialogResult dailr = retry.ShowDialog();
                    if (dailr == DialogResult.No) ExitProgram();}
                progN++;
                worker_test.ReportProgress((int) ((progN - 0.5)*100/report));
                Thread.Sleep(200);
                if (CancelPending(worker_test)) return -1;
```

```
                SetNodeImage(treeView1.Nodes[0].Nodes[1].Nodes[progN - 1], 2);
//programming node01n1 to in progress
                SetImage(2);
                PauseWorker(worker_test);
                if (_pause) MessageBox.Show("pause");
                StartWorker(worker_test);
                if (CancelPending(worker_test)) return -1;
                if (CancelPending(worker_test)) return -1;
                _busy2.WaitOne();
                Wri.ConWrite(Resources.Testing_time_is + ": " + test.TestingTime);
                _myTimer.Interval = 1000*test.TestingTime;
                Wri.ConWrite(Resources.Test_timer + ": " + _myTimer.Interval);
                Wri.ConWrite(Resources.Start_the_tester);
                ToggleTestPic(true); //picture showing test is ongoing
                SendData("FUNC:TEST ON"); //turn test on in the tester
                _elapsed = false;
                _myTimer.Stop();
                _myTimer.Start();//start timer
                while (!_elapsed){//wait until timer elapses
                    Thread.Sleep(200);
                    if (CancelPending(worker_test)) return -1;}
                SendData("FUNC:TEST?");
                string received2 = GetIncomingData(serialPort1);
                if (received2 != "TEST OFF")
                {
                    _myTimer.Stop();
                    _elapsed = false;
                    Wri.ConWrite(Resources.Test_still_not_finished);
                    _myTimer.Start();
                    do
                    {
                        if (CancelPending(worker_test)) return -1;
                        Thread.Sleep(1000);
                        SendData("FUNC:TEST?");//ask test status
                        received2 = GetIncomingData(serialPort1);//get answer
                        if (!_elapsed) continue;
                        if (!SendQuery("FUNC:TEST?", "TEST ON")) return -3;
                        Wri.ConWrite(Resources.Test_cannot_be_turned_off + ". " +
Resources.Do_it_manually, 2);
```

```
                        var tempor = new
MessageBoxOk(Resources.It_seems_that_the_test_is_still_running + ".\n " +
Resources.Exiting_testing + ".\n " +
Resources.If_the_test_should_have_stopped_then_turn_off_the_tester,
Resources._0ERROR);
                        tempor.ShowDialog();//ask if the test really is on
                        StopTest();
                        SetOnePort(7, false); //yellow led blink off
                        ExitProgram();
                    }
                    while (received2 != "TEST OFF");//while test is still on
                }
                ToggleTestPic(false);
                for (int i = 0; i < 3; i++)
                {
                    SendData("MEAS" + progN + "?");//ask for measurement results
                    received2 = GetIncomingData(serialPort1);//get results
                    if (received2.Split(',')[0] != "98")
                    {
                        Wri.ConWrite(Resources.Test_result + ": " + received2);
                        int imag;
                        if (received2.Split(',')[1].Trim() == "PASS" ||
received2.Split(',')[1].Trim() == "HOLD"){//if OK, display results and add to log
                            imag = 3;
                            Wri.ConWrite(Resources.Subtest + " " + progN + " " +
Resources._passed + ": " + received2.Replace("HOLD", "PASS"), 1);
                            logging.Add(";PASS;SUBTEST" + progN + ";" +
received2.Replace("HOLD", "PASS"));}
                        else if (received2.Split(',')[1].Trim() == "FAIL"){//if
failed, display results and add to log
                            imag = 1;
                            state = false;
                            Wri.ConWrite(Resources.Subtest + " " + progN + " " +
Resources._failed + ": " + received2, 1);
                            logging.Add(";FAIL;SUBTEST" + progN + ";" +
received2);}
                        else{//failed on other reason
                            imag = 4;
                            state = false;
                            Wri.ConWrite(Resources.Subtest + " " + progN + " " +
Resources._failed + " (" + Resources.__Error + ")" + ": " + received2, 1);
                            logging.Add(";FAIL;SUBTEST" + progN + ";" +
received2);}
```

```
                        SetNodeImage(treeView1.Nodes[0].Nodes[1].Nodes[progN - 1],
imag);
                        break;
                    }
                    Wri.ConWrite(Resources.No_result_for_test);
                }
                worker_test.ReportProgress(progN*100/report);
                Thread.Sleep(15);
                if (!state){
                    Wri.ConWrite(Resources.Test_failed);
                    return 0;}
                _busy2.WaitOne();
                if (CancelPending(worker_test)) return -1;
            }
            return 1;
        }

        public bool ToggleTestPic(bool toggle)
        {
            if (toggle)
            {
                SetOnePort(7, true); //yellow led blinking to test status
                SetOnePort(5, false);
                SetOnePort(6, false); //red and green led off
                this.SynchronizedInvoke(() =>{
                    label_testing.Text = Resources._0TESTING;
                    label_testing2.Text = Resources._0BE_0CAREFUL;
                    pictureBox_testing.Show();});//warnings to testing in progress
state
                return true;
            }
            SetOnePort(7, false);//yellow led off
            this.SynchronizedInvoke(() =>{//if test not running, then warnings off
                label_testing.Text = " ";
                label_testing2.Text = " ";
                pictureBox_testing.Hide();});
            return false;
        }

        private void ChangeLastHold(int prog) //##same as SefelecTester/MainWin.cs
```

```
//--TESTING

        #endregion Testing

        #region Building Treeview

        private void FillTreeView(string[] names) //##same as
SefelecTester/MainWin.cs

        private void FillNode(List<ItemInfo> items, TreeNode node) //##same as
SefelecTester/MainWin.cs

        #endregion Building Treeview

        #region Sending Data
         //SENDING DATA
        public void SendData(string toSend, bool newline = true) //##same as
SefelecTester/MainWin.cs

        private bool SendDataWithCheck(string toSend) //##same as
SefelecTester/MainWin.cs

        public bool SendQuery(string query, string answer) //##same as
SefelecTester/MainWin.cs

        private string[] GetError()//##same as SefelecTester/MainWin.cs

        private static void EmptyIncomingBuffer(SerialPort serial) //##same as
SefelecTester/MainWin.cs

        private string GetIncomingData(SerialPort serial) //##same as
SefelecTester/MainWin.cs

        public bool SendDataAndRead(string data, string question, string answer)
//##same as SefelecTester/MainWin.cs

        public bool SendDataAndReadEncased(string data, string question, string
answer) //##same as SefelecTester/MainWin.cs

        public string GetDataEncased(string data) //##same as
SefelecTester/MainWin.cs
//--SENDING DATA
```

```
        #endregion Sending Data

        private class ItemInfo //##same as SefelecTester/MainWin.cs

        public struct Listboxes //##same as SefelecTester/MainWin.cs

        public struct SavedTest //##same as SefelecTester/MainWin.cs

        public struct TestList //##same as SefelecTester/MainWin.cs
    }
}
```

Programmi fail 17: BarCode.cs

```
using System;
using System.Drawing;
using System.Timers;
using System.Windows.Forms;
using SefelecTester_DDC.Properties;
using Timer = System.Timers.Timer;

namespace SefelecTester_DDC
{
    public partial class BarCode : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        private bool _keyElapsed = true; //KEYPRESS EVENT

        public BarCode(MainWin main, string instructions)
        {
            InitializeComponent();
            if (!IsHandleCreated) CreateHandle();//if not created, then force it
            KeyPreview = true; //KEYPRESS EVENT
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            textBox_barCode.LostFocus += ControlLostFocus;//focus lost from
barcode entry event
            textBox_barCode.Leave += ControlLostFocus;//focus lost from barcode
entry event
            textBox_barCode.Text = main.SerialNo.Text;
            label2.Text = Resources.Instructions + ": " + instructions + "\n";
```

```
            textBox_barCode.Focus();
            if (main.checkBox1.Checked){
                label1.Text = Resources.Scan_the_products_barcode;
                textBox_barCode.Focus();
                textBox_barCode.Focus();
                textBox_barCode.Text = "";}
            else{
                label1.Text = Resources.Change_the_barcode_manually + " (" +
Resources._if_needed + ")";
                textBox_barCode.Text = IncrementOne(main.SerialNo.Text);}
            button_cancel.Location = new Point(button_cancel.Location.X,
label2.Location.Y + label2.Size.Height + 10);
            button_ok.Location = new Point(button_ok.Location.X, label2.Location.Y
+ label2.Size.Height + 10);
            textBox_barCode.Focus();
        }

        protected override sealed void CreateHandle(){
            base.CreateHandle(); }

        private void ControlLostFocus(object sender, EventArgs e){
            textBox_barCode.Focus(); }//refocus on barcode entry

        private void button_cancel_Click(object sender, EventArgs e) //##same as
SefelecTester/BarCode.cs

        private static unsafe string IncrementOne(string str) //##same as
SefelecTester/BarCode.cs


//KEYPRESS EVENT
        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//##same as SefelecTester/BarCode.cs
//--KEYPRESS EVENT
    }
}
```

Programmi fail 18: SafeSerialPort.cs

```
using System;
using System.IO;
using System.IO.Ports;
```

```
namespace SefelecTester_DDC
{
    public class SafeSerialPort : SerialPort
    {
        private Stream _theBaseStream;

        public SafeSerialPort(string portName, int baudRate, Parity parity, int
dataBits, StopBits stopBits)
            : base(portName, baudRate, parity, dataBits, stopBits) { }

        public new void Open()
        {
            try
            {
                base.Open();
                _theBaseStream = BaseStream;
                GC.SuppressFinalize(BaseStream);
            }
            catch {
            }
        }

        public new void Dispose() { Dispose(true); }

        protected override void Dispose(bool disposing)
        {
            if (disposing && (base.Container != null)) { base.Container.Dispose();
}
            try
            {
                if (_theBaseStream.CanRead)
                {
                    _theBaseStream.Close();
                    GC.ReRegisterForFinalize(_theBaseStream);
                }
            }
            catch
            { // ignore exception - bug with USB - serial adapters.
            }
```

```
            base.Dispose(disposing);
        }
    }
}
```

Programmi fail 19: StartupSettings.cs

```
using SefelecTester_DDC.Properties;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Management;
using System.Timers;
using System.Windows.Forms;
using Timer = System.Timers.Timer;

namespace SefelecTester_DDC
{
    public partial class StartupSettings : Form
    {
        //KEYPRESS EVENT commands are used to ignore enter key from barcode reader
        //look to BarCode.cs for more information
        private const string SefelecVID = "VID_10C4";
        private const string avrVID = "VID_03EB";
        private readonly List<string> _description = new List<string>();
        private readonly Timer _keyTimer = new Timer(200); //KEYPRESS EVENT
        public int ControllerNumber = 999;
        public string SelectedCom;
        public string SelectedComAvr;
        public string SelectedProductPath;
        private bool _keyElapsed = true; //KEYPRESS EVENT
        private SerialPort _serialPort;

        public StartupSettings()
        {
            InitializeComponent();
            KeyPreview = true; //KEYPRESS EVENT
```

```
            _keyTimer.Elapsed += keyTimer_Elapsed; //KEYPRESS EVENT
            GetAvailable();
            DropDownListMod();
            GetProductList();
        }

        private void DropDownListMod()
        {
            if (SelectSerial.SelectedItem == null || SelectSerialAVR.SelectedItem
== null || SelectProduct.SelectedItem == null)
                buttonOK.Enabled = false;
            else buttonOK.Enabled = true;
        }

        private void Startup_Settings_Load(object sender, EventArgs e)
        {
            if (SelectSerial.Items.Count == 0 || SelectProduct.Items.Count == 0 ||
SelectSerialAVR.Items.Count == 0)
            {//if no COM devices connected or no parameter files present
                string noItems = "\n" + Resources.Please + ":";
                if (SelectSerial.Items.Count == 0)
                    noItems += "\n" + Resources.Connect_a_COM_device + " (" +
Resources.Sefelec_tester + ")";
                if (SelectProduct.Items.Count == 0)
                    noItems += "\n" +
Resources.Add_a_testing_program_to_the_rigth_folder;
                if (SelectSerialAVR.Items.Count == 0)
                    noItems += "\n" + Resources.Connect_a_COM_device + " (" +
Resources._testing_box + ")";
                noItems += "\n\n" + Resources.Then_press_reload;
                var noItem = new MessageBoxOk(Resources.There_is_no_item_to_select
+ noItems, Resources._0ERROR);
                noItem.ShowDialog();//ask to add missing items
            }
            else{//select first tiems
                SelectProduct.SelectedIndex = 0;
                SelectProduct.SelectedItem = null;}
        }

        private void GetAvailable() //Get available choices
        {
```

```
            SelectSerial.Items.Clear();//clear list
            SelectSerialAVR.Items.Clear();//clear list
            try
            {
                var searcher = new ManagementObjectSearcher("root\\WMI", "SELECT *
FROM MSSerial_PortName");//use WMI function
                foreach (ManagementObject queryObj in
searcher.Get().Cast<ManagementObject>())
                {//get all serial ports
                    _serialPort = new SerialPort(queryObj["PortName"].ToString(),
115200, Parity.None, 8, StopBits.One);
                    Wri.ConWrite("InstanceName: " + queryObj["InstanceName"]);
                    Wri.ConWrite("PortName: " + queryObj["PortName"]);
                    try
                    {
                        string added = null;
                        _serialPort.Open();
                        if
(queryObj["InstanceName"].ToString().Contains(SefelecVID)){//add to name if
Sefelec product
                            SelectSerial.Items.Add(queryObj["PortName"] + " <=");
                            added = " <=" + Resources.Sefelec_tester;}
                        else SelectSerial.Items.Add(queryObj["PortName"]);
                        if
(queryObj["InstanceName"].ToString().Contains(avrVID)){//add to name if AVR
product
                            SelectSerialAVR.Items.Add(queryObj["PortName"] + "
<=");
                            added = " <=" + Resources._0AVR_controller;}
                        else SelectSerialAVR.Items.Add(queryObj["PortName"]);
                        _description.Add(queryObj["InstanceName"] + added);
                        _serialPort.Close();
                    }
                    catch (Exception ex)
                    {
                        Wri.ConWrite(Resources.Listing_ports + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                        var tempor = new MessageBoxOk(Resources.Serial_port + " ("
+ queryObj["PortName"] + ") " + Resources._is_in_use + ". " +
Resources.You_cannot_select_this + ". " + Resources._0ERROR + ": " + ex,
Resources._0ERROR + " (" + Resources.with_COM_port + ")");
                        tempor.ShowDialog();
                    }
```

```
                }
                int n = 0;
                SelectSerial.SelectedItem = null;
                SelectSerialAVR.SelectedItem = null;//deselect all
                foreach (string str in _description){
                    if (str.Contains(SefelecVID)) SelectSerial.SelectedIndex =
n;//find Sefelec product and select the first one
                    else if (str.Contains(avrVID)) SelectSerialAVR.SelectedIndex =
n;//find AVR product and select the first one
                    n++;}
            }
            catch (ManagementException ex){
                Wri.ConWrite(Resources.Querying_for_WMI_data + ". " +
Resources.Exception_raised + "(" + ex.Message + "): " + ex, 2);
                var tempor = new
MessageBoxOk(Resources.An_error_occurred_while_querying_for_WMI_data + ": " +
ex.Message, Resources.Is_a_device_connected + "?");
                tempor.ShowDialog();}
        }

        private void GetProductList()
        {
            SelectProduct.Items.Clear();
            string folder =
Path.GetDirectoryName(Process.GetCurrentProcess().MainModule.FileName) +
@"\Products\";
            const string filter = "*.csv";
            string[] files = Directory.GetFiles(folder, filter);//get all .csv
files in folder
            foreach (string path in files){
                SelectProduct.Items.Add(Path.GetFileNameWithoutExtension(path) + "
@" + path); }
        } //--Get Available Choices

        private void SelectSerial_SelectedIndexChanged(object sender, EventArgs e)
//  Get Changes
        {
            SelectedCom = SelectSerial.Text.Split(' ')[0];
            SerialPortName.Text = _description[SelectSerial.SelectedIndex];
            DropDownListMod();
        }

        private void tooteValik_SelectedIndexChanged(object sender, EventArgs e)
```

```
        {
            string[] words = SelectProduct.Text.Split('@');
            SelectedProductPath = words[words.Count() - 1];
            Wri.ConWrite(Resources.Selected_product + ": " + SelectedProductPath);
            DropDownListMod();
        } //--Get Changes

        private void Startup_Settings_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13){//enter key selects 'OK'
                buttonOK.Select();
                buttonOK.PerformClick();}
            else if (e.KeyChar == (char)27){//escape key selects 'Cancel'
                buttonCancel.Select();
                buttonCancel.PerformClick();}
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs
e){//controller number changed
            ControllerNumber = (int)numericUpDown1.Value;}

        private void buttonCancel_Click(object sender, EventArgs e){
            var exiting = new MessageBoxAlt(Resources.Exit + "?",
Resources.Are_you_sure_you_want_to_close_the_window + "?", false, false);
            DialogResult exit = exiting.ShowDialog();
            DialogResult = exit == DialogResult.Yes ? DialogResult.Cancel :
DialogResult.None;}//if 'No' chosen then cancel exiting

        private void button_reload_Click(object sender, EventArgs e)
        {//get lists again
            GetAvailable();
            DropDownListMod();
            GetProductList();
            SelectProduct.SelectedIndex = 0;
            SelectSerial.SelectedIndex = 0;
            SelectSerialAVR.SelectedIndex = 0;
        }

        private void button_other_Click(object sender, EventArgs e) //##same as
SefelecTester/StartupSettings.cs
```

```
//KEYPRESS EVENT
        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
//##same as SefelecTester/StartupSettings.cs
//--KEYPRESS EVENT

        private void SelectSerialAVR_SelectedIndexChanged(object sender, EventArgs
e)
        {
            SelectedComAvr = SelectSerialAVR.Text.Split(' ')[0];
            SerialPortAVRName.Text = _description[SelectSerialAVR.SelectedIndex];
            DropDownListMod();
        }
    }
}
```

L7.3. Programm mikrokontrolleri juhtimiseks

Programm on loodud LUFA [79] Virtual Serial Device projekti. Välja toon failid, mida on muudetud:

- VirtualSerial.h (Programmi fail 21: VirtualSerial.h, lk 213).
- VirtualSerial.c (Programmi fail 20: VirtualSerial.c, lk 204).
- Avr035.h – lisasin, et lihtsamini (kasutades makrosid) kontrolleri pinnide olekuid seada ja lugeda (Programmi fail 22: avr035.h, lk 216).

Programmi fail 20: VirtualSerial.c

```
/*
             LUFA Library
     Copyright (C) Dean Camera, 2014.

  dean [at] fourwalledcubicle [dot] com
           www.lufa-lib.org
*/

/*
  Copyright 2014  Dean Camera (dean [at] fourwalledcubicle [dot] com)

  Permission to use, copy, modify, distribute, and sell this
  software and its documentation for any purpose is hereby granted
```

```
  without fee, provided that the above copyright notice appear in
  all copies and that both that the copyright notice and this
  permission notice and warranty disclaimer appear in supporting
  documentation, and that the name of the author not be used in
  advertising or publicity pertaining to distribution of the
  software without specific, written prior permission.

  The author disclaims all warranties with regard to this
  software, including all implied warranties of merchantability
  and fitness.  In no event shall the author be liable for any
  special, indirect or consequential damages or any damages
  whatsoever resulting from loss of use, data or profits, whether
  in an action of contract, negligence or other tortious action,
  arising out of or in connection with the use or performance of
  this software.
*/

/** \file
 *
 *  Main source file for the VirtualSerial demo. This file contains the main tasks
of
 *  the demo and is responsible for the initial application hardware
configuration.
 */

#include "VirtualSerial.h"

/** LUFA CDC Class driver interface configuration and state information. This
structure is
 *  passed to all CDC Class driver functions, so that multiple instances of the
same class
 *  within a device can be differentiated from one another.
 */
USB_ClassInfo_CDC_Device_t VirtualSerial_CDC_Interface =
      {
            .Config =
                  {
                        .ControlInterfaceNumber   = INTERFACE_ID_CDC_CCI,
                        .DataINEndpoint           =
                              {
                                    .Address          = CDC_TX_EPADDR,
```

```
                                        .Size             = CDC_TXRX_EPSIZE,
                                        .Banks            = 1,
                                    },
                                .DataOUTEndpoint =
                                    {
                                        .Address          = CDC_RX_EPADDR,
                                        .Size             = CDC_TXRX_EPSIZE,
                                        .Banks            = 1,
                                    },
                                .NotificationEndpoint =
                                    {
                                        .Address          =
CDC_NOTIFICATION_EPADDR,
                                        .Size             =
CDC_NOTIFICATION_EPSIZE,
                                        .Banks            = 1,
                                    },
                            },
        };

/** Standard file stream for the CDC interface when set up, so that the virtual
CDC COM port can be
 *  used like any regular character stream in the C APIs.
 */
static FILE USBSerialStream;

int yellowBlink=0;

/** Main program entry point. This routine contains the overall program flow,
including initial
 *  setup of all components and the main program loop.
 */
int main(void)
{
        SetupHardware();
        /* Create a regular character stream for the interface so that it can be
used with the stdio.h functions */
        CDC_Device_CreateStream(&VirtualSerial_CDC_Interface, &USBSerialStream);
        LEDs_SetAllLEDs(LEDMASK_USB_NOTREADY);
        GlobalInterruptEnable();
        SetupIOPorts();
```

```
	TCCR1B |= (1 << WGM12); // Configure timer 1 for CTC mode
	TIMSK1 |= (1 << OCIE1A); // Enable CTC interrupt
	sei(); //  Enable global interrupts
	OCR1A   = 25000; // Set CTC compare value, with a prescaler of 64
	TCCR1B |= ((1 << CS10) | (1 << CS11)); // Start timer at Fcpu/64

	for (;;)
	{
		GetData();
		/* Must throw away unused bytes from the host, or it will lock up
while waiting for the device */
		CDC_Device_ReceiveByte(&VirtualSerial_CDC_Interface);
		CDC_Device_USBTask(&VirtualSerial_CDC_Interface);
		USB_USBTask();
	}
}
ISR(TIMER1_COMPA_vect)
{
	if (yellowBlink == 1){//do on timer completion
		C_FLIPBIT(ly7);} // Toggle yellow LED
}

/** Configures the board hardware and chip peripherals for the demo's
functionality. */
void SetupHardware(void)
{
#if (ARCH == ARCH_AVR8)
	/* Disable watchdog if enabled by bootloader/fuses */
	MCUSR &= ~(1 << WDRF);
	wdt_disable();

	/* Disable clock division */
	clock_prescale_set(clock_div_1);
#elif (ARCH == ARCH_XMEGA)
	/* Start the PLL to multiply the 2MHz RC oscillator to 32MHz and switch the
CPU core to run from it */
	XMEGACLK_StartPLL(CLOCK_SRC_INT_RC2MHZ, 2000000, F_CPU);
	XMEGACLK_SetCPUClockSource(CLOCK_SRC_PLL);

	/* Start the 32MHz internal RC oscillator and start the DFLL to increase it
to 48MHz using the USB SOF as a reference */
```

```
	XMEGACLK_StartInternalOscillator(CLOCK_SRC_INT_RC32MHZ);
	XMEGACLK_StartDFLL(CLOCK_SRC_INT_RC32MHZ, DFLL_REF_INT_USBSOF, F_USB);

	PMIC.CTRL = PMIC_LOLVLEN_bm | PMIC_MEDLVLEN_bm | PMIC_HILVLEN_bm;
#endif

	/* Hardware Initialization */
	LEDs_Init();
	USB_Init();
}

void SetupIOPorts(void)
{
	DDRA=0xFF;//set port A pins as outputs
	PORTA=0x00;//port A pins will be set to 0000000
	DDRF=0xFF;//set port F pins as outputs
	PORTF=0x00;//port F pins will be set to 0000000
	DDRC=0xFF;//set port C pins as outputs
	PORTC=0x00;//port C pins will be set to 0000000
}

void SendFeedback(char* switched_)
{
	char* answerString=NULL;//string with port state
	/*checks incoming string for each port name*/
	if (strncmp(switched_, "pr0", strlen("pr0")) == 0){
		if (C_CHECKBIT(pr0)) answerString = " st pr0 1 \r\n";//if port bit
set high
		else answerString = " st pr0 0 \r\n";}//if port bit set low
	else if(strncmp(switched_, "hv1", strlen("hv1"))==0){
		if(C_CHECKBIT(hv1)) answerString=" st hv1 1 \r\n";
		else answerString=" st hv1 0 \r\n";}
	else if(strncmp(switched_, "hv2", strlen("hv2"))==0){
		if(C_CHECKBIT(hv2)) answerString=" st hv2 1 \r\n";
		else answerString=" st hv2 0 \r\n";}
	else if(strncmp(switched_, "hv3", strlen("hv3"))==0){
		if(C_CHECKBIT(hv3)) answerString=" st hv3 1 \r\n";
		else answerString=" st hv3 0 \r\n";}
	else if(strncmp(switched_, "hv4", strlen("hv4"))==0){
		if(C_CHECKBIT(hv4)) answerString=" st hv4 1 \r\n";
```

```
            else answerString=" st hv4 0 \r\n";}
      else if(strncmp(switched_, "lr5", strlen("lr5"))==0){
            if(C_CHECKBIT(lr5)) answerString=" st lr5 1 \r\n";
            else answerString=" st lr5 0 \r\n";}
      else if(strncmp(switched_, "lg6", strlen("lg6"))==0){
            if(C_CHECKBIT(lg6)) answerString=" st lg6 1 \r\n";
            else answerString=" st lg6 0 \r\n";}
      else if(strncmp(switched_, "ly7", strlen("ly7"))==0){
            if(yellowBlink==1) answerString=" st ly7 1 \r\n";
            else answerString=" st ly7 0 \r\n";}
      else answerString=" err st na \r\n";
      fputs(answerString, &USBSerialStream);//send answer over stream
}
void SetPort(char* port, char* state)
{
      char* answerString=NULL;//answer to send afet setting
      int i=-1;//init at unused value
      if(strncmp(state, "1", strlen("1"))==0) i=1;//set i according to state
      else if(strncmp(state, "0", strlen("0"))==0) i=0;
      if (i>=0)
      {
            if(strncmp(port, "a3", strlen("a3"))==0){//command to set all used
ports
                  if(i==1) {ChangeAllPorts(true); answerString=" new all 1
\r\n";}
                  else {ChangeAllPorts(false); answerString=" new all 0 \r\n";}}
            /*set port bit according to port name and given state*/
            else if(strncmp(port, "pr0", strlen("pr0"))==0){
                  if(i==1) {C_SETBIT(pr0); answerString=" new pr0 1 \r\n";}
                  else {C_CLEARBIT(pr0); answerString=" new pr0 0 \r\n";}}
            else if(strncmp(port, "hv1", strlen("hv1"))==0){
                  if(i==1) {C_SETBIT(hv1); answerString=" new hv1 1 \r\n";}
                  else {C_CLEARBIT(hv1); answerString=" new hv1 0 \r\n";}}
            else if(strncmp(port, "hv2", strlen("hv2"))==0){
                  if(i==1) {C_SETBIT(hv2); answerString=" new hv2 1 \r\n";}
                  else {C_CLEARBIT(hv2); answerString=" new hv2 0 \r\n";}}
            else if(strncmp(port, "hv3", strlen("hv3"))==0){
                  if(i==1) {C_SETBIT(hv3); answerString=" new hv3 1 \r\n";}
                  else {C_CLEARBIT(hv3); answerString=" new hv3 0 \r\n";}}
            else if(strncmp(port, "hv4", strlen("hv4"))==0){
```

```
                  if(i==1) {C_SETBIT(hv4); answerString=" new hv4 1 \r\n";}
                  else {C_CLEARBIT(hv4); answerString=" new hv4 0 \r\n";}}
            else if(strncmp(port, "lr5", strlen("lr5"))==0){
                  if(i==1) {C_SETBIT(lr5); answerString=" new lr5 1 \r\n";}
                  else {C_CLEARBIT(lr5); answerString=" new lr5 0 \r\n";}}
            else if(strncmp(port, "lg6", strlen("lg6"))==0){
                  if(i==1) {C_SETBIT(lg6); answerString=" new lg6 1 \r\n";}
                  else {C_CLEARBIT(lg6); answerString=" new lg6 0 \r\n";}}
            else if(strncmp(port, "ly7", strlen("ly7"))==0){
                  if(i==1) {yellowBlink=1; C_SETBIT(ly7); answerString=" new ly7
1 \r\n";}
                  else {yellowBlink=0; C_CLEARBIT(ly7); answerString=" new ly7 0
\r\n";}}
            else answerString=" err new na \r\n";//command not correct, send
error
      }
      else{//state value wriong
            answerString=" err new unk \r\n";}
      fputs(answerString, &USBSerialStream);//send answer over stream
}
bool GetData(void)
{
      char getString[CDC_TXRX_EPSIZE];
      char sendString[CDC_TXRX_EPSIZE];
      if(fgets(getString, CDC_TXRX_EPSIZE, &USBSerialStream)!=NULL)
      {//data has command, param1, (param2)
            C_FLIPBIT(led1);//AVR board LED, blinks show data received
            sprintf(sendString, " mir %s\r\n", getString);
            fputs(sendString, &USBSerialStream);//mirror command back
            char** tokens=str_split(getString, ' ');//split command by delimiter
            if(strncmp(*(tokens), "gg", strlen("gg"))==0){//command is get state
                  SendFeedback(*(tokens+1));}
            else if(strncmp(*(tokens), "ss", strlen("ss"))==0){//command is send
state
                  SetPort(*(tokens+1), *(tokens+2));}
            return true;
      }
      else//no data recieved
            return false;
}
```

```
void ChangeAllPorts(bool state)
{//sets all used ports according to given state
	if (state){
		C_SETBIT(pr0);//port to high
		C_SETBIT(hv1);
		C_SETBIT(hv2);
		C_SETBIT(hv3);
		C_SETBIT(hv4);
		C_SETBIT(lr5);
		C_SETBIT(lg6);
		C_SETBIT(ly7);}
	else{
		C_CLEARBIT(pr0);//port to low
		C_CLEARBIT(hv1);
		C_CLEARBIT(hv2);
		C_CLEARBIT(hv3);
		C_CLEARBIT(hv4);
		C_CLEARBIT(lr5);
		C_CLEARBIT(lg6);
		C_CLEARBIT(ly7);}
}

/** Event handler for the library USB Connection event. */
void EVENT_USB_Device_Connect(void){
	LEDs_SetAllLEDs(LEDMASK_USB_ENUMERATING);}

/** Event handler for the library USB Disconnection event. */
void EVENT_USB_Device_Disconnect(void){
	LEDs_SetAllLEDs(LEDMASK_USB_NOTREADY);}

/** Event handler for the library USB Configuration Changed event. */
void EVENT_USB_Device_ConfigurationChanged(void){
	bool ConfigSuccess = true;
	ConfigSuccess &=
CDC_Device_ConfigureEndpoints(&VirtualSerial_CDC_Interface);
	LEDs_SetAllLEDs(ConfigSuccess ? LEDMASK_USB_READY : LEDMASK_USB_ERROR);}

/** Event handler for the library USB Control Request reception event. */
void EVENT_USB_Device_ControlRequest(void){
	CDC_Device_ProcessControlRequest(&VirtualSerial_CDC_Interface);}
```

```
char** str_split(char* a_str, const char a_delim)
{//split string by given delimiter
	char** result = 0;
	size_t count = 0;
	char* tmp = a_str;//input string
	char* last_comma = 0;
	char delim[2];
	delim[0] = a_delim;//delimiter
	delim[1] = 0;
	char * temp = tmp;
	int i;
	while (*temp == ' ') {//if character is space, done until first other
character
		temp++;}//don't count it
	i = 0;
	while (*temp != NULL) {//char has content
		tmp[i++] = *temp++;}//copy to tmp
	tmp[i] = '\0';//add end
	removeDoubles(tmp, a_delim);//remove double delimiters
	/* Count how many elements will be extracted. */
	while (*tmp)
	{
		if (a_delim == *tmp){
			count++;//counts delimiters
			last_comma = tmp;}
		tmp++;
	}
	/* Add space for trailing token. */
	count += last_comma < (a_str + strlen(a_str) - 1);
	/* Add space for terminating null string so caller
	knows where the list of returned strings ends. */
	count++;
	result = malloc(sizeof(char*)* count);//allocate memory
	if (result)
	{
		size_t idx = 0;
		char* token = strtok(a_str, delim);//split string by delimiter, takes
first part before delimiter
		while (token){
```

```
                    assert(idx < count);//check value, should never happen
                    *(result + idx++) = strdup(token);//copy from token to [first
block+menmer] address
                    token = strtok(0, delim);}//split remaining string by delimiter
             assert(idx == count - 1);
             *(result + idx) = 0;//add 0 as last
      }
      return result;
}

/*thank to sizablegrin*/
/*http://forums.devshed.com/programming-42/function-takes-string-removes-double-
letters-487011.html */
void copyForward(char *str)
{
      char *src = str + 1;
      char *dest = str;
      if (*dest == '\0') return;//end character reached
      dest = str;
      src = str + 1;
      while (*dest) *dest++ = *src++;
      return;
}

void removeDoubles(char *str, const char* a_delim)
{
      char *current = str + 1;
      char *last = str;
      while (*current)
      {
             while (*current == *last && *current==a_delim) copyForward(current);
             current++;
             last++;
      }
      return;
}
/*thank to sizablegrin*/
```

Programmi fail 21: VirtualSerial.h

```
/*
```

```
             LUFA Library
     Copyright (C) Dean Camera, 2014.

  dean [at] fourwalledcubicle [dot] com
           www.lufa-lib.org
*/

/*
  Copyright 2014  Dean Camera (dean [at] fourwalledcubicle [dot] com)

  Permission to use, copy, modify, distribute, and sell this
  software and its documentation for any purpose is hereby granted
  without fee, provided that the above copyright notice appear in
  all copies and that both that the copyright notice and this
  permission notice and warranty disclaimer appear in supporting
  documentation, and that the name of the author not be used in
  advertising or publicity pertaining to distribution of the
  software without specific, written prior permission.

  The author disclaims all warranties with regard to this
  software, including all implied warranties of merchantability
  and fitness.  In no event shall the author be liable for any
  special, indirect or consequential damages or any damages
  whatsoever resulting from loss of use, data or profits, whether
  in an action of contract, negligence or other tortious action,
  arising out of or in connection with the use or performance of
  this software.
*/

/** \file
 *
 *  Header file for VirtualSerial.c.
 */

#ifndef _VIRTUALSERIAL_H_
#define _VIRTUALSERIAL_H_

      /* Includes: */
            #include <avr/io.h>
```

```
#include <avr/wdt.h>
#include <avr/power.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include <util/delay.h>
#include "avr035.h"

#include "Descriptors.h"

#include <LUFA/Drivers/Board/LEDs.h>
#include <LUFA/Drivers/Board/Joystick.h>
#include <LUFA/Drivers/USB/USB.h>
#include <LUFA/Platform/Platform.h>

      /* Macros: */
            /** LED mask for the library LED driver, to indicate that the USB
interface is not ready. */
            #define LEDMASK_USB_NOTREADY      LEDS_LED1

            /** LED mask for the library LED driver, to indicate that the USB
interface is enumerating. */
            #define LEDMASK_USB_ENUMERATING  (LEDS_LED2 | LEDS_LED3)

            /** LED mask for the library LED driver, to indicate that the USB
interface is ready. */
            #define LEDMASK_USB_READY        (LEDS_LED2 | LEDS_LED4)

            /** LED mask for the library LED driver, to indicate that an error
has occurred in the USB interface. */
            #define LEDMASK_USB_ERROR        (LEDS_LED1 | LEDS_LED3)

            /*define names for AVR board LEDs*/
            #define led1 PORTD, 4
            #define led2 PORTD, 5
            #define led3 PORTD, 6
            #define led4 PORTD, 7

            /*define names for AVR ports*/
```

```
#define pr0  PORTC, 0//power relay
#define hv1  PORTC, 3//hv relay
#define hv2  PORTC, 4//hv relay
#define hv3  PORTC, 7//hv relay
#define hv4  PORTA, 0//hv relay
#define lr5  PORTA, 3//red LED
#define lg6  PORTA, 4//green LED
#define ly7  PORTA, 7//yellow LED

/* Function Prototypes: */
	void SetupHardware(void);
	void CheckJoystickMovement(void);
	void SetupIOPorts(void);
	bool GetData(void);
	void SendFeedback(char*);
	void SetPort(char*, char*);
	char** str_split(char*, const char);
	void removeDoubles(char*, const char*);
	void copyForward(char*);
	void ChangeAllPorts(bool);

	void EVENT_USB_Device_Connect(void);
	void EVENT_USB_Device_Disconnect(void);
	void EVENT_USB_Device_ConfigurationChanged(void);
	void EVENT_USB_Device_ControlRequest(void);


#endif
```

Programmi fail 22: avr035.h

```
#ifndef _AVR035_H_
#define _AVR035_H_

// from AVR035: Efficient C Coding for AVR

/*set and clear bits in IO registers*/
#define SETBIT(ADDRESS,BIT) (ADDRESS |= (1<<BIT))//set port pin bit high
#define CLEARBIT(ADDRESS,BIT) (ADDRESS &= ~(1<<BIT))//set port pin bit low
```

```
#define FLIPBIT(ADDRESS,BIT) (ADDRESS ^= (1<<BIT))//change port pin bit to
opposite
#define CHECKBIT(ADDRESS,BIT) (ADDRESS & (1<<BIT))//check port pin bit state

/*bit macros for setting and clearing bits*/
#define SETBITMASK(x,y) (x |= (y))
#define CLEARBITMASK(x,y) (x &= (~y))
#define FLIPBITMASK(x,y) (x ^= (y))
#define CHECKBITMASK(x,y) (x & (y))

/*macros to set and clear bits by only the bit name*/
#define VARFROMCOMB(x, y) x
#define BITFROMCOMB(x, y) y

#define C_SETBIT(comb) SETBIT(VARFROMCOMB(comb), BITFROMCOMB(comb))
#define C_CLEARBIT(comb) CLEARBIT(VARFROMCOMB(comb), BITFROMCOMB(comb))
#define C_FLIPBIT(comb) FLIPBIT(VARFROMCOMB(comb), BITFROMCOMB(comb))
#define C_CHECKBIT(comb) CHECKBIT(VARFROMCOMB(comb), BITFROMCOMB(comb))
/*macros to set and clear bits by only the bit name*/

#endif
```

L8. Juhendid

Kahjuks on aruandesse kopeeritud juhendite vormistus natuke valeks muutunud.

L8.1. Juhend rakiseta programmile

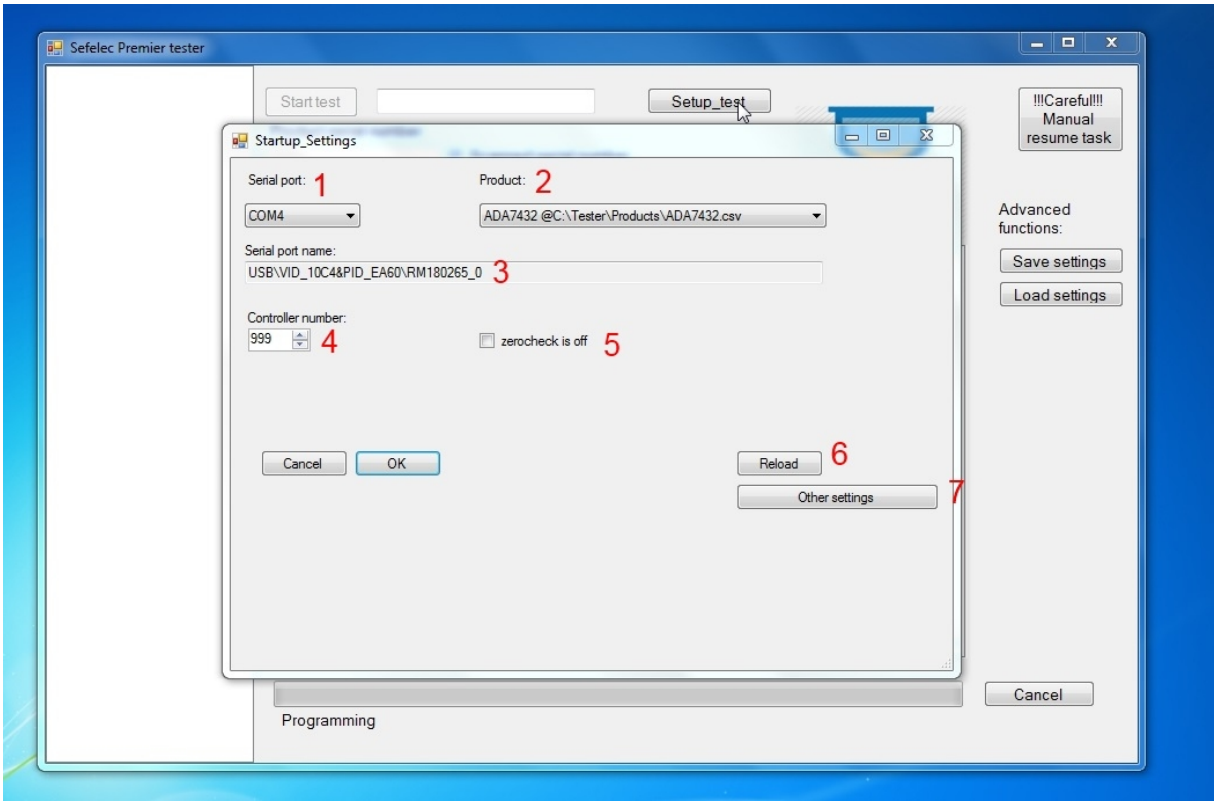
SEFELEC PREMIER 2804 TESTRI KASUTAMINE

1. Testri programmi valimine

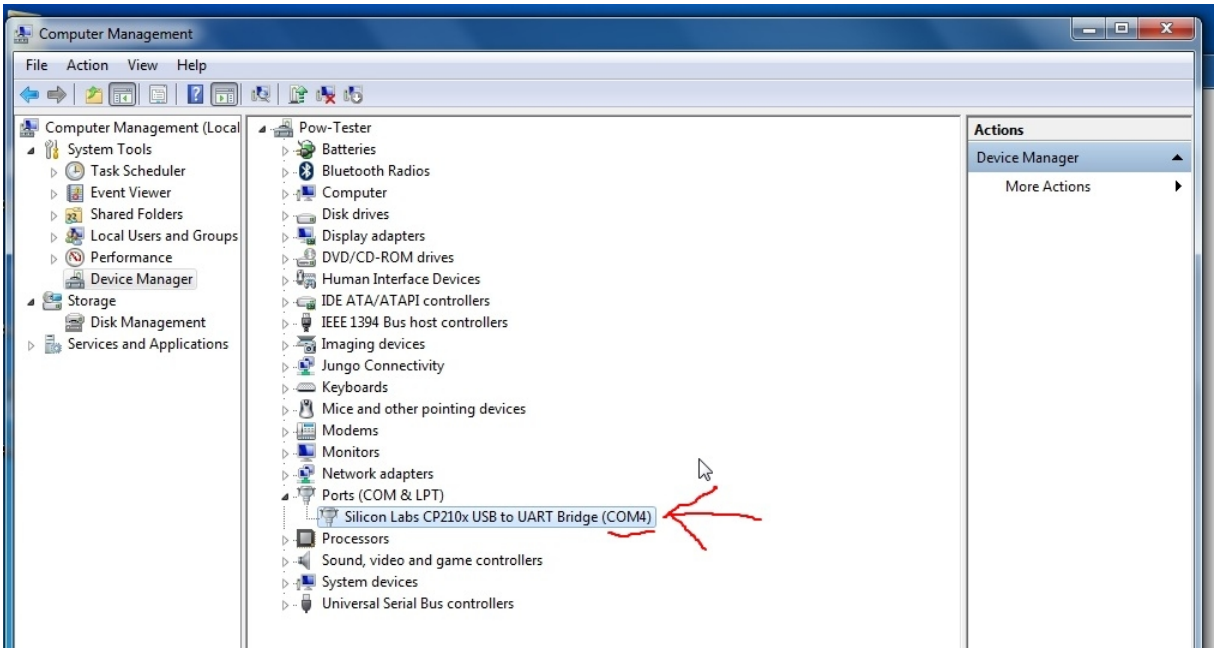
1. Lülita sisse tester ja arvuti. Vaata, et nad oleksid ühendatud USB kaabliga.

Soovitavalt peaks tester soojenema 30min enne testidega alustamist, et tagada parim tulemus. Kui toode ei läbi testi ja sisse lülitamisest on möödas alla 30 minuti, siis oota kuni aeg täis saab ja proovi uuesti.

2. Käivita arvutis programm ’Sefelec tester’. Avaneb selline pilt:



3. <1>- vali COM port; <2>- vali kasutatav test; <3>- COM pordi andmed; <4>- vali oma id number; <5>- vali, kas kasutad ’zerocheck’ funktsiooni (NB! tavaliselt ei kasutata); <6>- lae uuesti praegune aken; <7>- lisafunktsioonid.
4. Vali kasutatav COM port <1>. Kui sa ei tea, millist valida, siis vaata järgi ’Device managerist’.

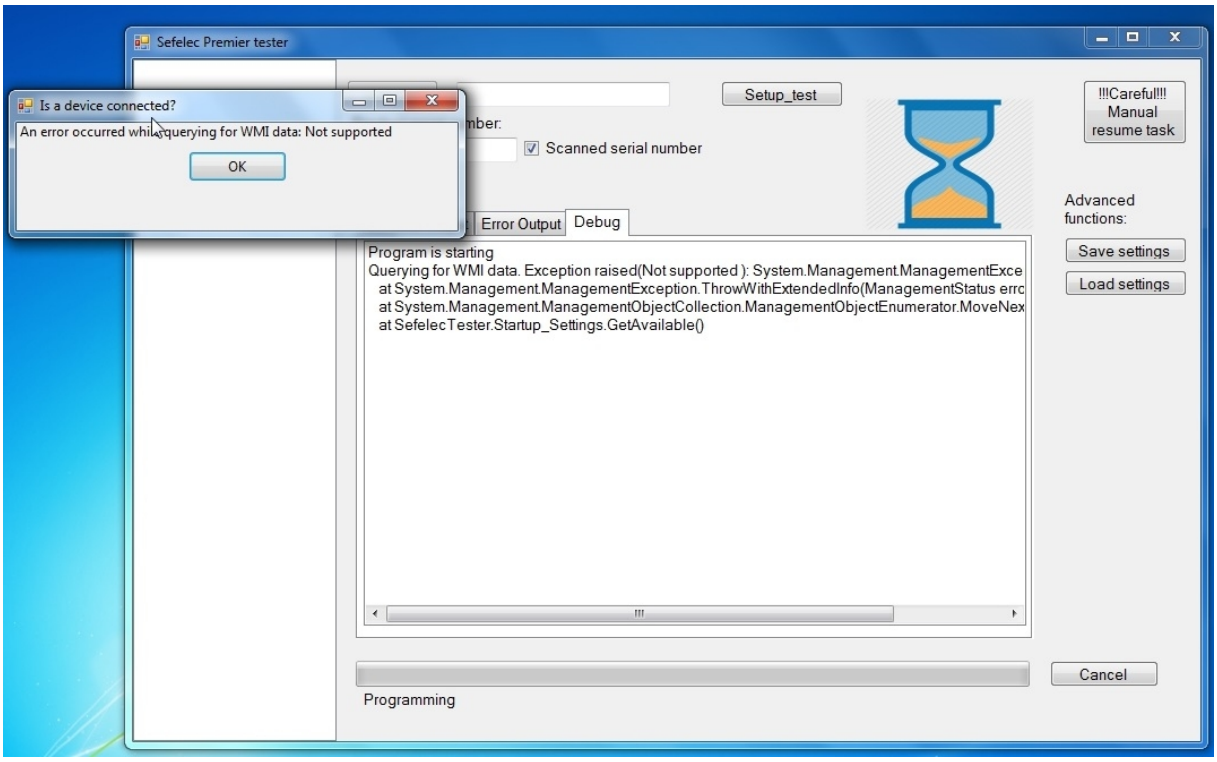


5. Vali testitav toode <2>.
6. Sisesta oma id number <4>.

7. Kui vaja teostada ’zerocheck’ siis vali vastav kast <5>.
8. Vajuta ’OK’ või klaviatuurilt ’Enter’. Kui kõik läheb hästi, jätka sammuga ’Testri programmeerimine’.

Kui nimekirjas ei ole toote testi, siis tuleb see lisada (räägi inseneriga). Kui nimekirjas ei ole õiget COM porti, siis kontrolli, et juhtmed oleksid arvutiga korralikult ühendatud ja vajuta ’Reload’ <6> ning tee uuesti sammud 3 kuni 8.

1.1 Võimalikud vead

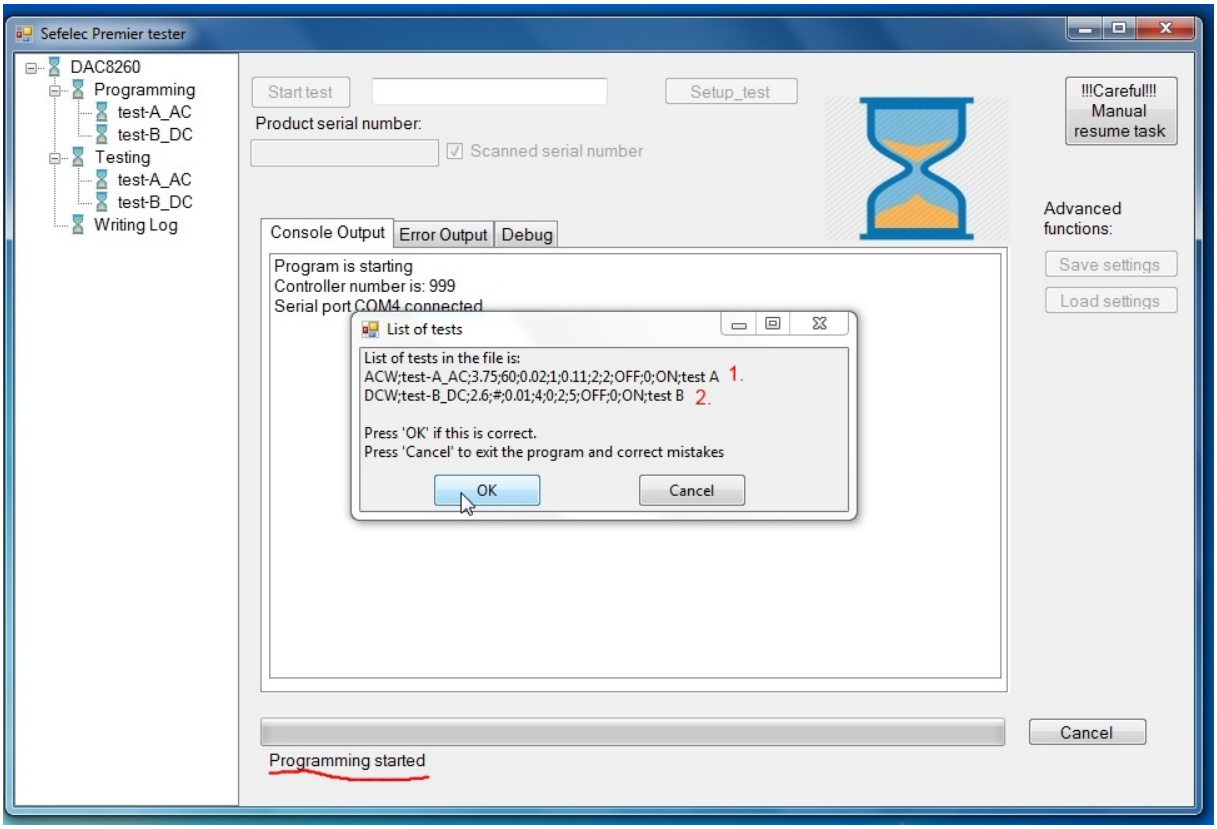


Tester ei ole sisse lülitatud või ei ole arvutiga ühendatud. Kontrolli üle ja paranda vead. Seejärel vajuta ’OK’ ja jälgi järgmise akna juhiseid.

2. Testri programmeerimine

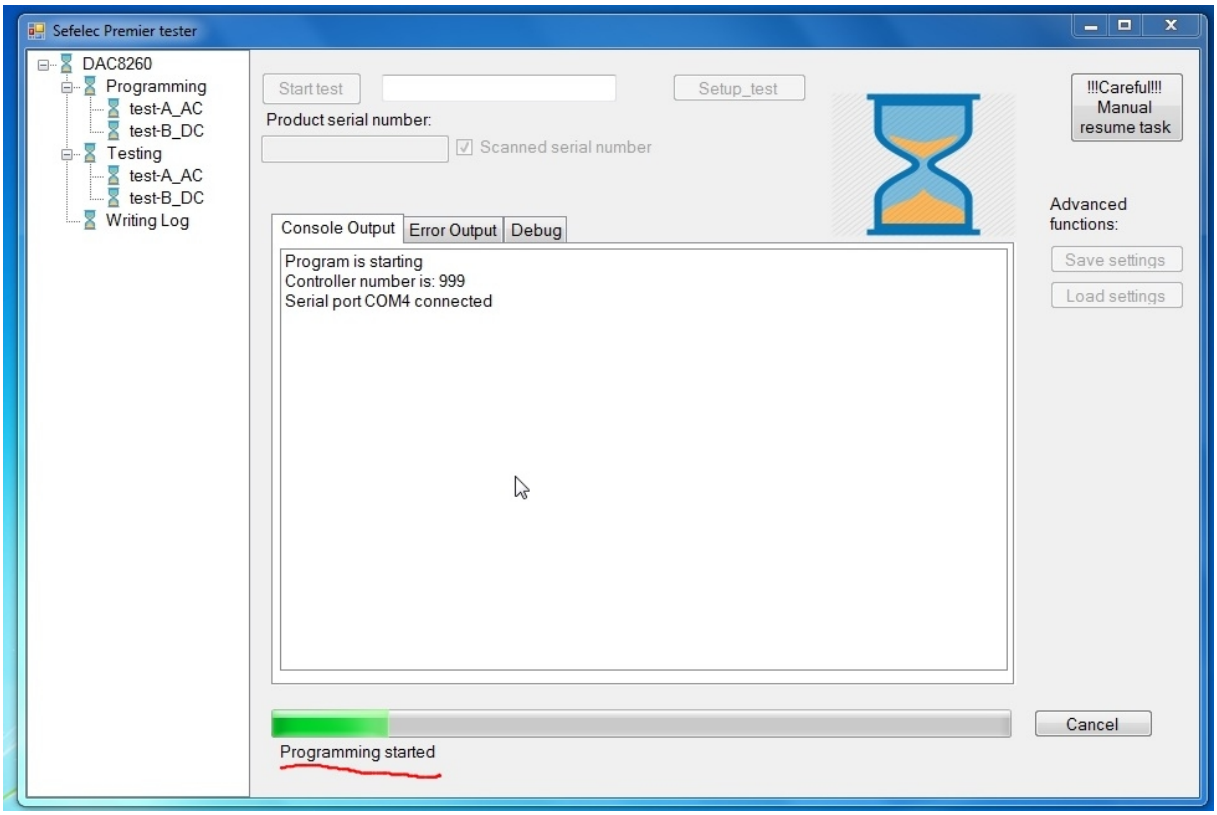
Automaatselt algab testri programmeerimine. Kui see ei alga, siis sulge programm ja alusta uuesti.

Kui programmeerimine algab, siis avaneb hetke pärast järgmine pilt:

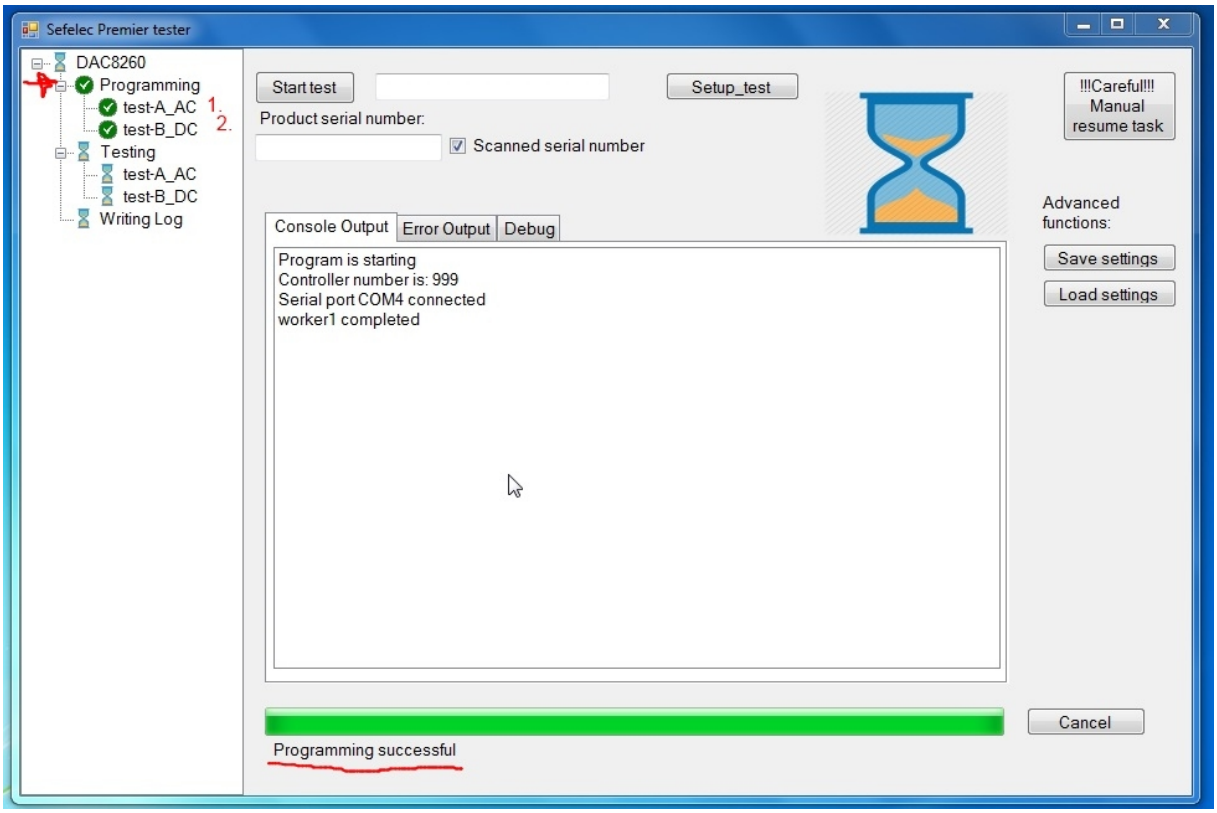


1. Kontrolli, kas testide andmed (1. ja 2., teste võib olla ka muu arv) on õiged. Kui nad vastavad tootele, siis vajuta ’OK’.

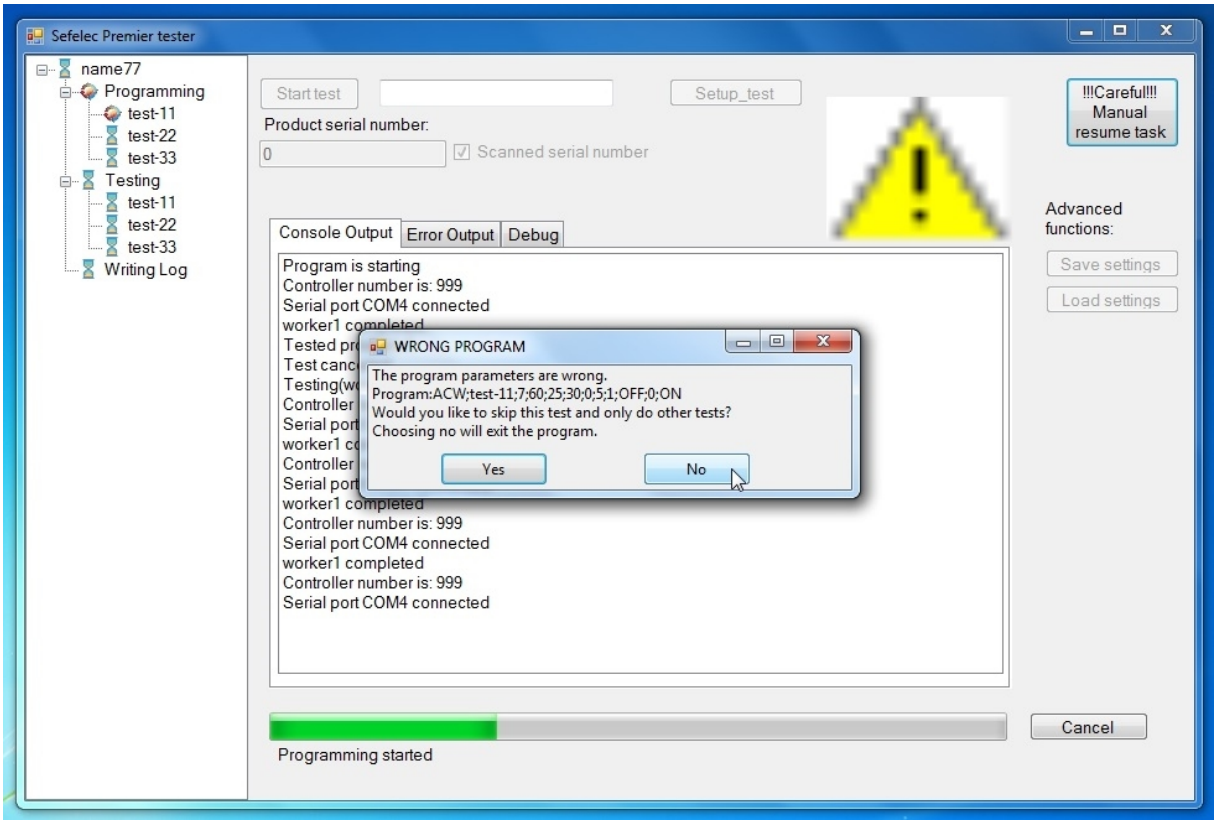
Avaneb selline pilt:



2. Oota kuni lõpeb programmeerimine ja avaneb selline pilt (’Programming successful’):



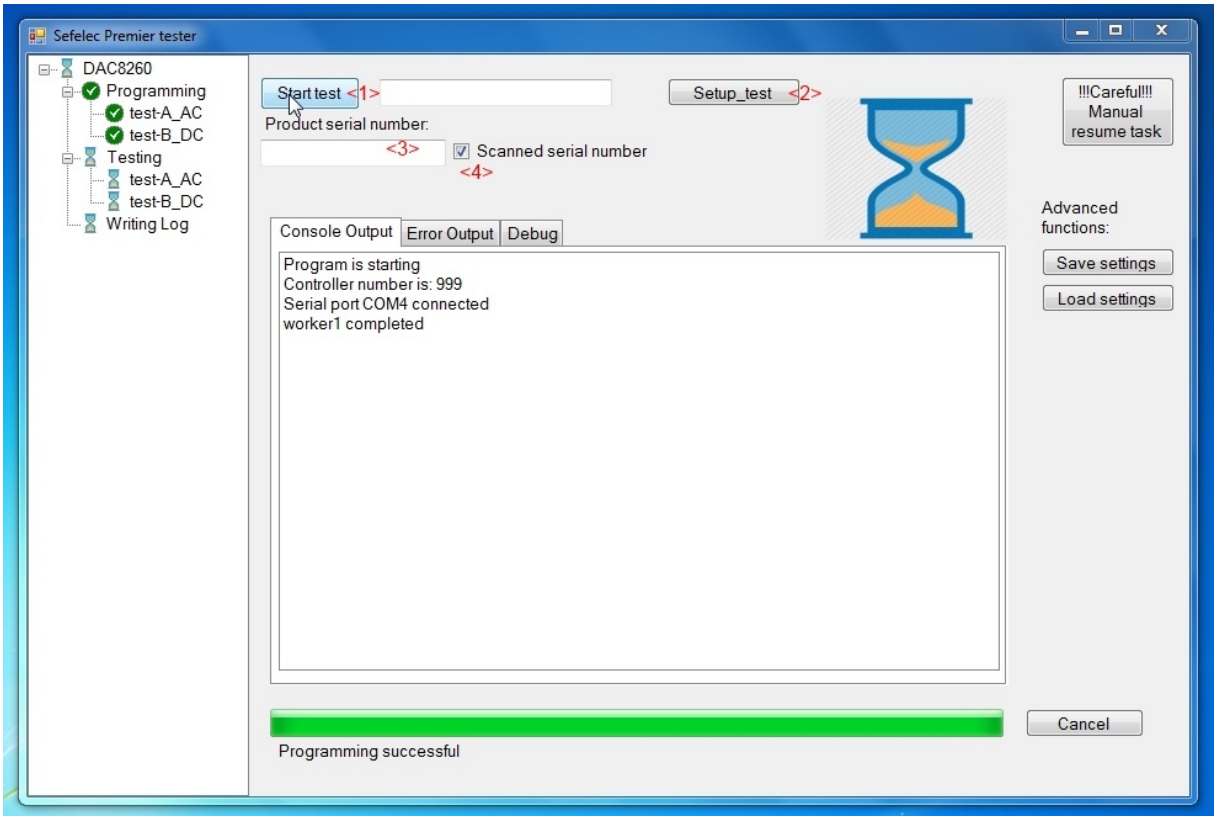
2.1 Vead



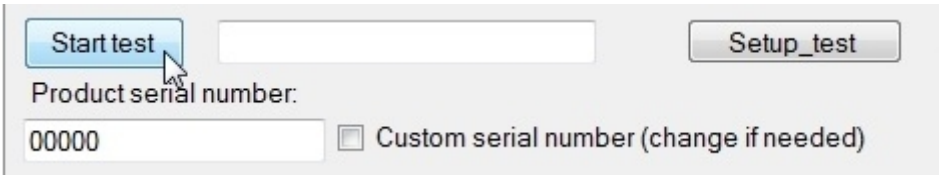
Programmi parameetrid on valed. Kui näidatav test ei ole vajalik, siis vajuta ’Yes’ ja testimisel ei kasutata seda testi. Peaaegu alati on kõik testid vajalikud. Sel juhul vajuta ’No’, välju programmist ja räägi inseneriga, et parandada testi parameetrid.

3. Testimine

Avaneb järgmine pilt:



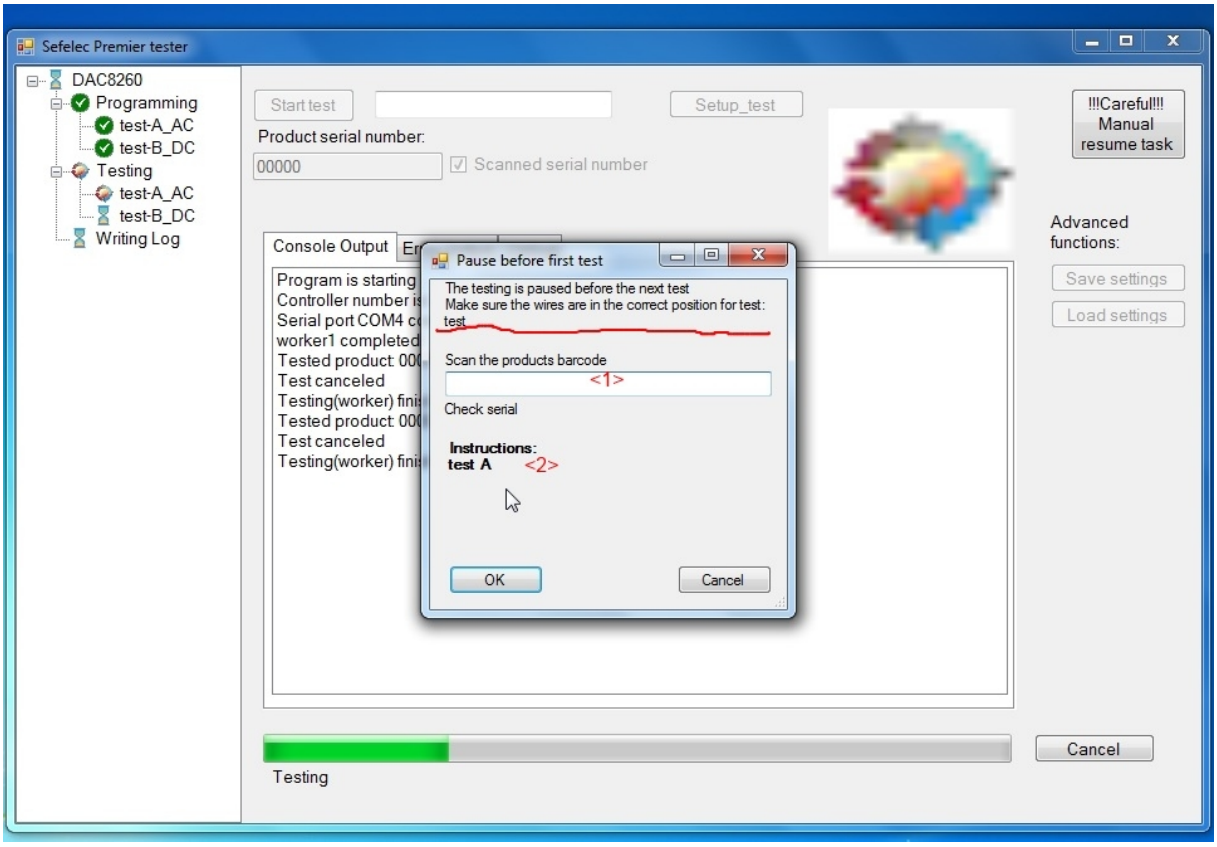
1. <4> Vali, kas skänneerid toote seerianumbri või sisestatakse järjestikused numbrid. Kui tootel ei ole skaneeritavat seerianumbrit, siis eemalda valik <4> ja sisesta nulltoote seerianumber <3>. Sel juhul on iga toote number 1 võrra suurem kui eelmisel. Esimese toote number on sisestatud nulltoote number+1.



2. Kui oled testiks valmis, vajuta ’Start test’.

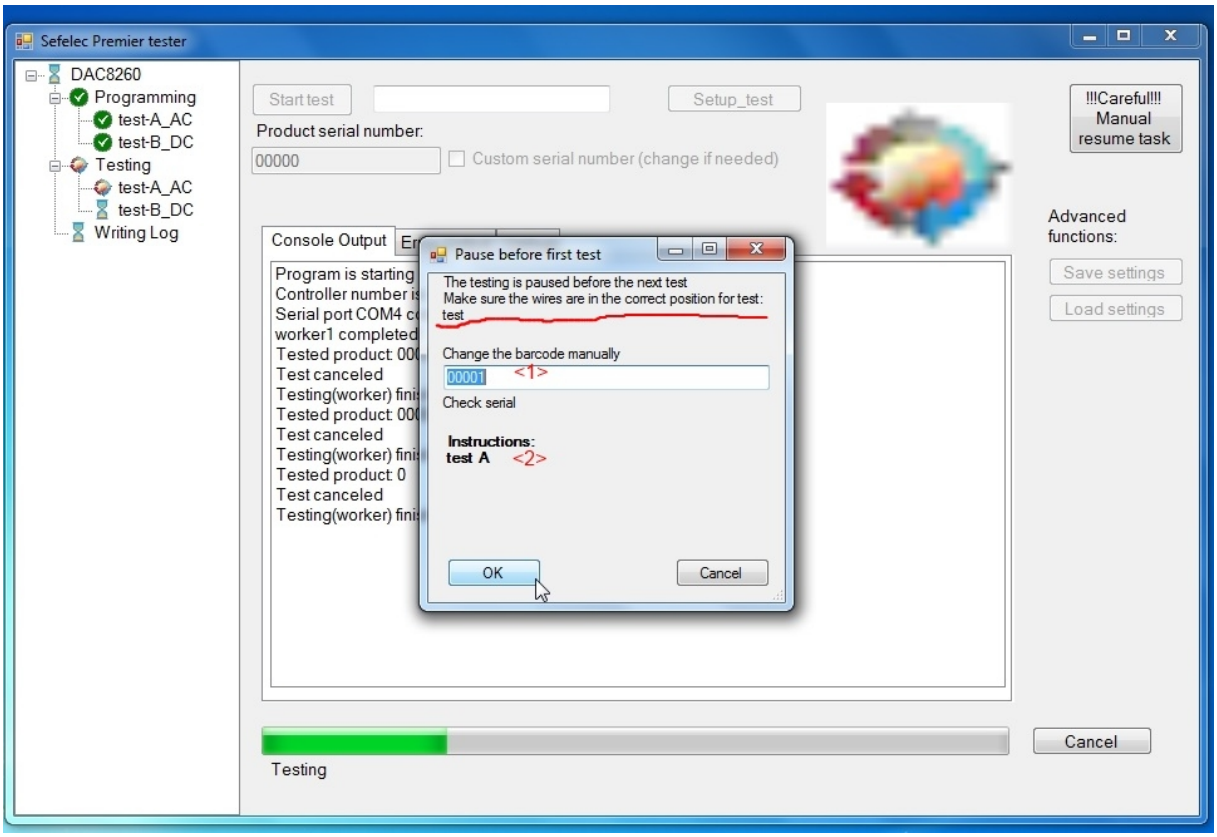
NB! ÄRA järgmises dialoogis veel ’OK’ vajuta.

Kui kasutad skänneeritud seerianumbrit:



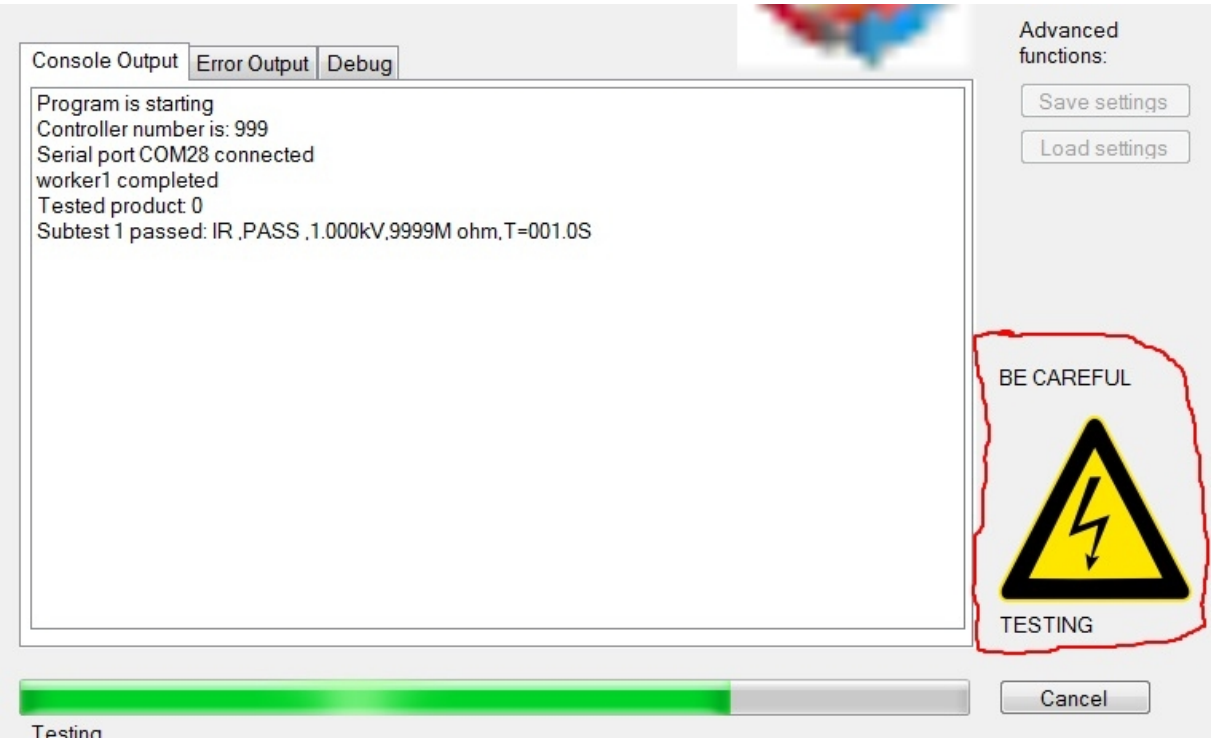
Peale koodi skänneerimist ilmub see kasti <1>.

Kui kasutad teist varianti:



Toote kood on näha lahtris <1>.

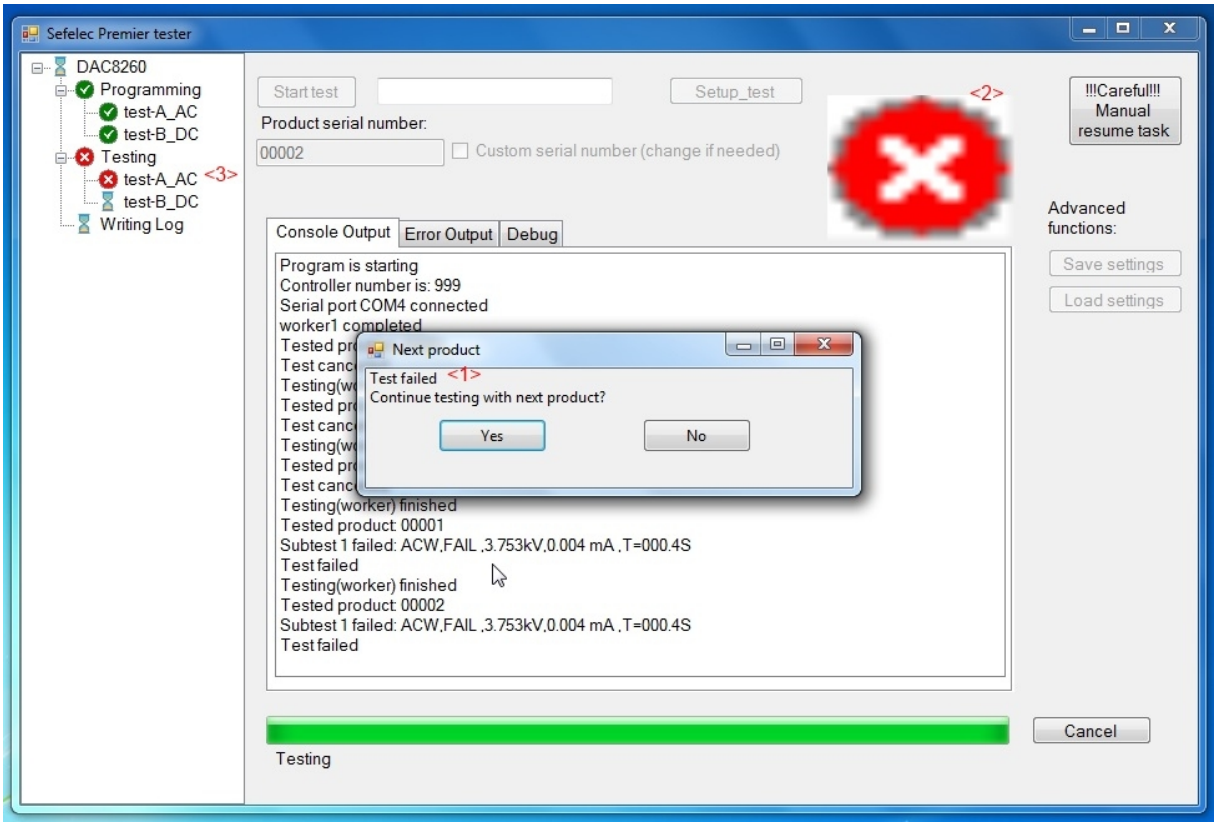
Mõlemal juhul on näidatud ka lisajuhised <2>. **Enne ’OK’ vajutamist peavad testri otsikud olema testimise asendis. ’OK’ vajutamine alustab kohe testi.**



Nähes seda ikooni on test käigus. Sellel ajal ei tohi liigutada testri otsikuid. Testi toimumist näitab see ikoon ja vilkuv ’high voltage’ tuli testril.

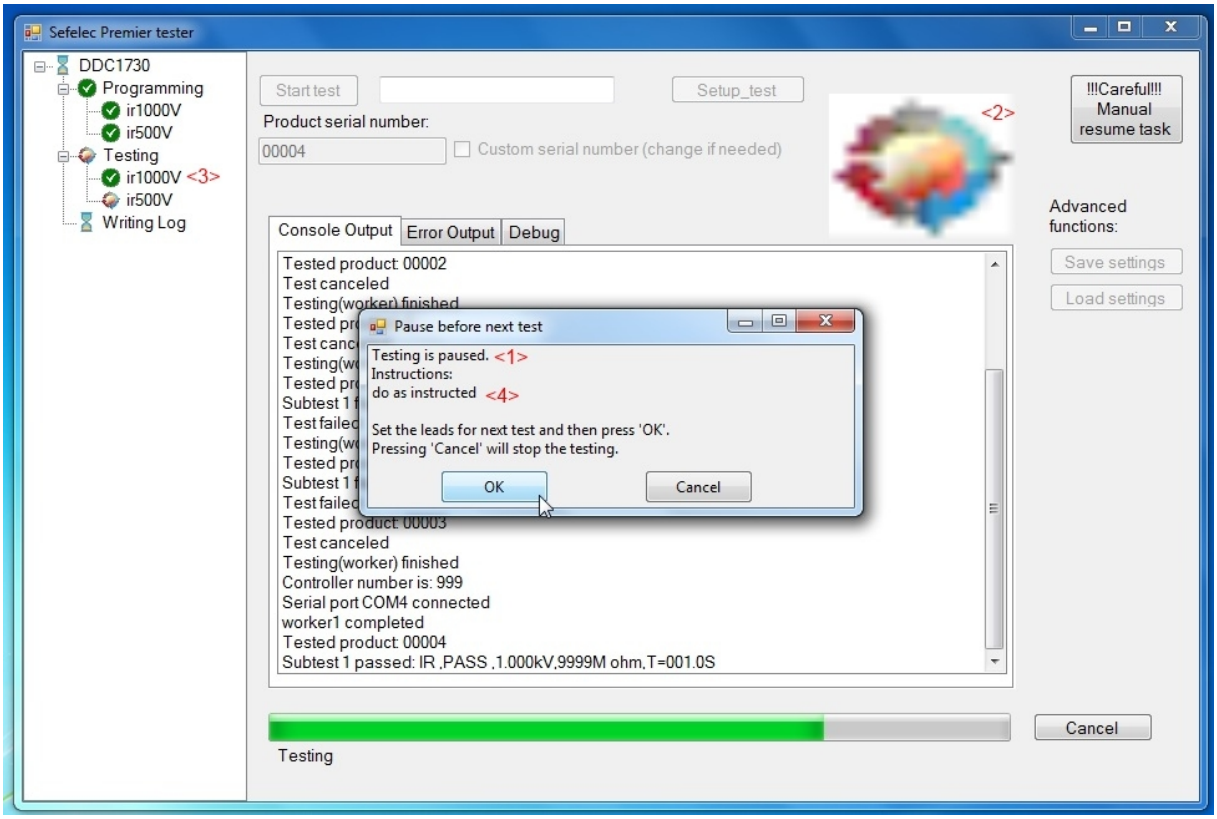
3. Vajuta ’OK’.
4. Oota kuni avaneb järgmine aken. **Testi ajal vilgub testril ’high voltage’ tuli. Sellel ajal ei tohi testri otsikuid liigutada!**

Kui test läbi kukub:



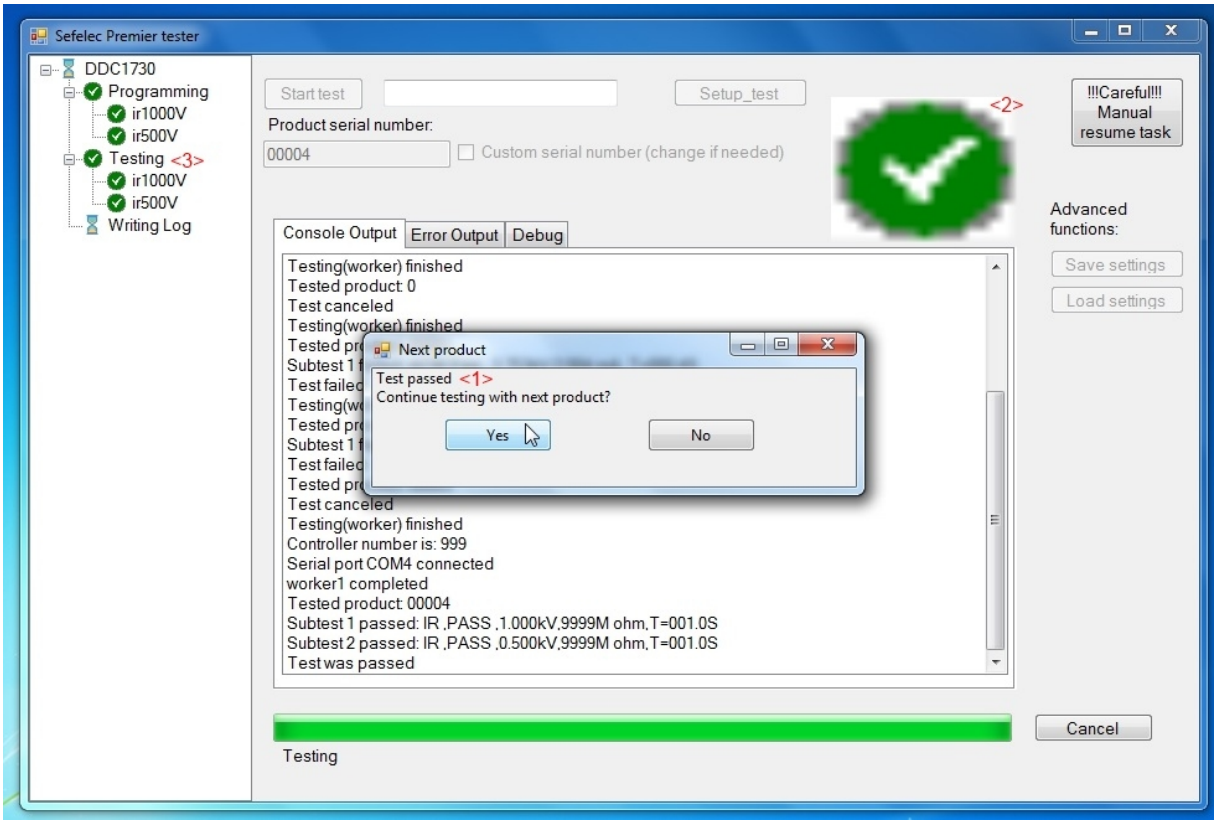
Testi tulemust näitavad 2 ikooni <2> ja <3>. Testi tulemus on kirjas aknas <1>. Testi ebaõnnestumisel jäetakse järgmised testid vahele. ’Yes’ vajutamine alustab järgmise toote testimist, test veel ei käivitu.

Kui test õnnestub:



Testi läbimisel (kui on veel teste järjekorras) peatatakse testimine ajutiselt <1>. Testi tulemust näitab ikoon <3>. Ikoon <2> näitab, et test on jätkumas. Näha on ka järgmise testi lisajuhendid <4>.

Viimase testi läbimisel:

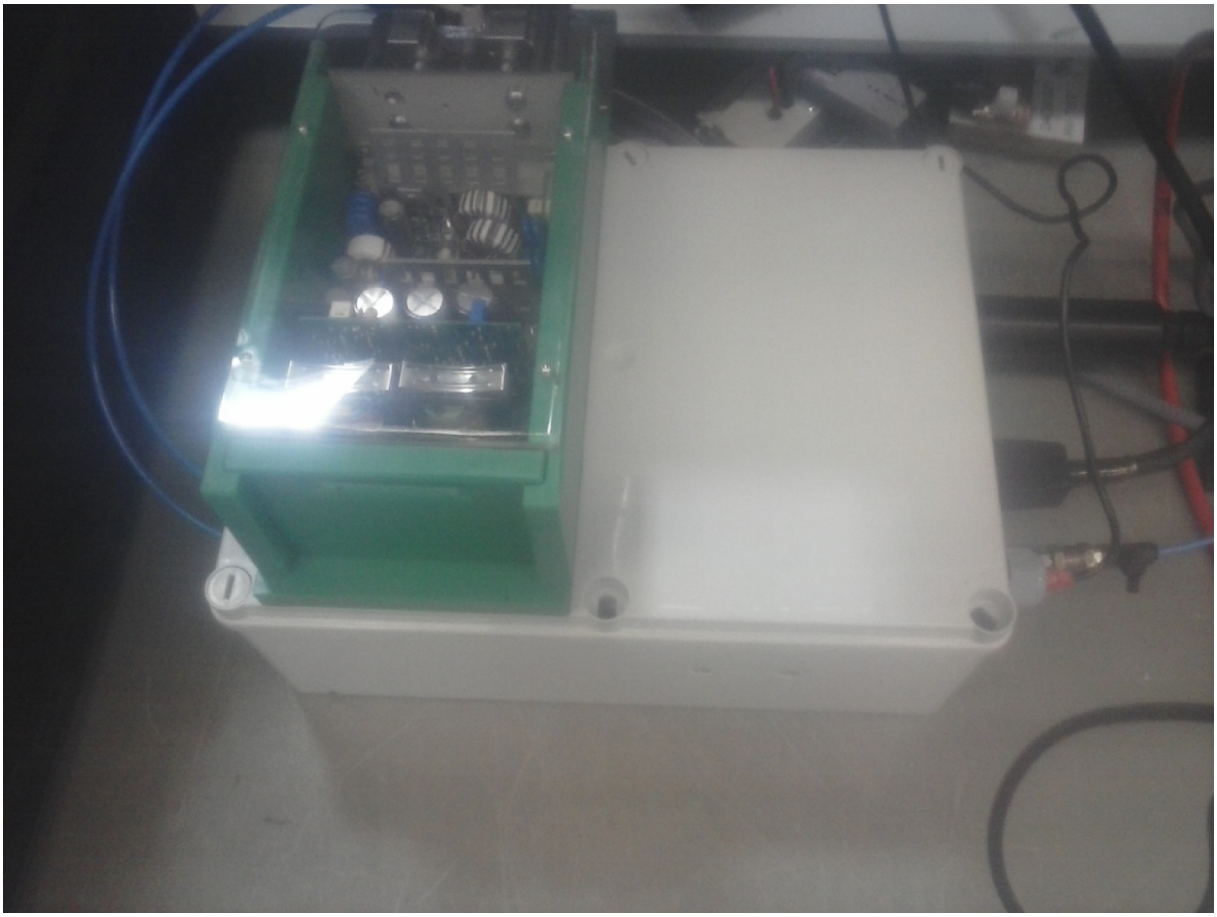


Kogu testi tulemus on kirjas <1> ja seda näitavad ka ikoonid <2> ja <3>. Vajutades ’Yes’ jätkub testimine järgmise tootega.

5. Vajuta ’Yes’ või ’No’. Kui vajutad ’Yes’ jätka juhendiga peale punkti 2.

L8.2. Juhend rakisega programmile

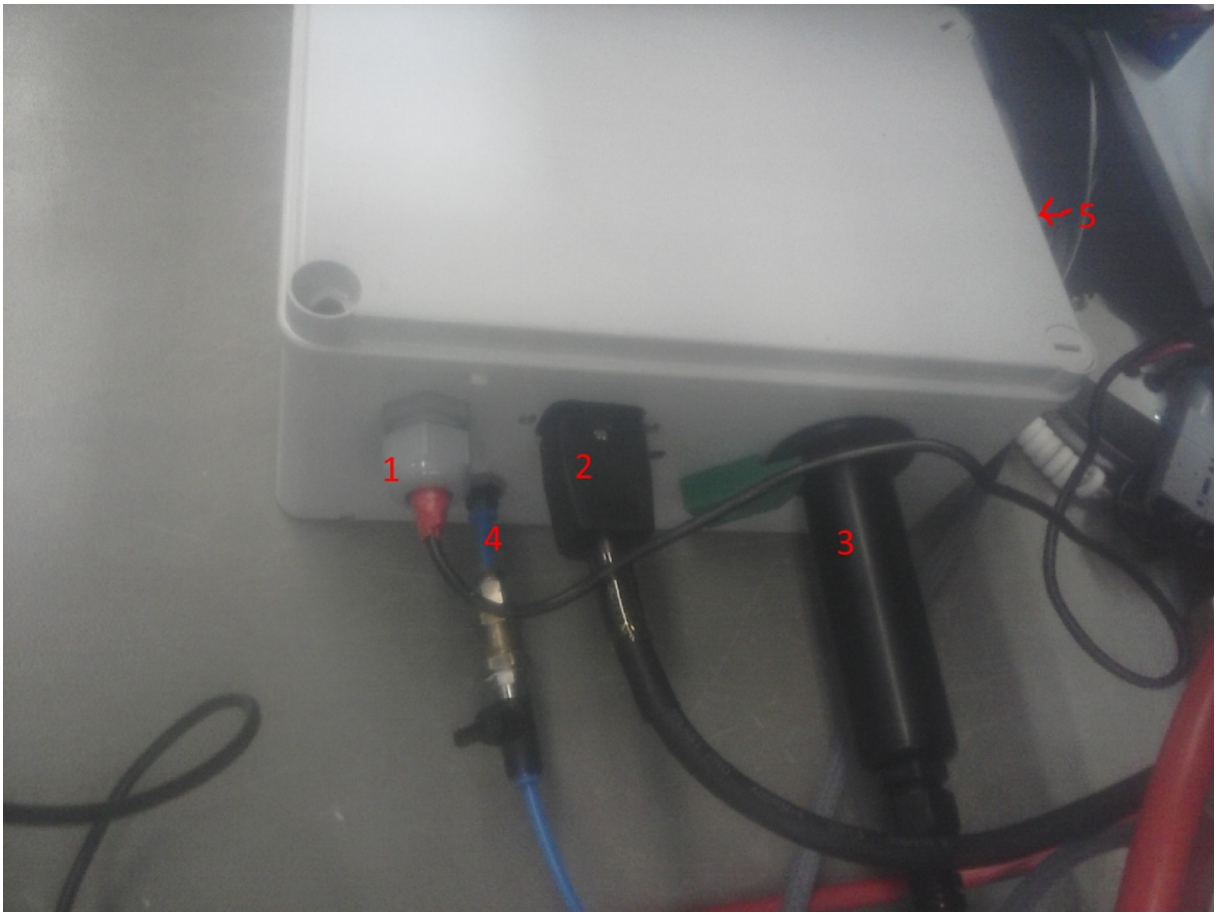
SEFELEC PREMIER 2804 TESTRI KASUTAMINE RAKISEGA

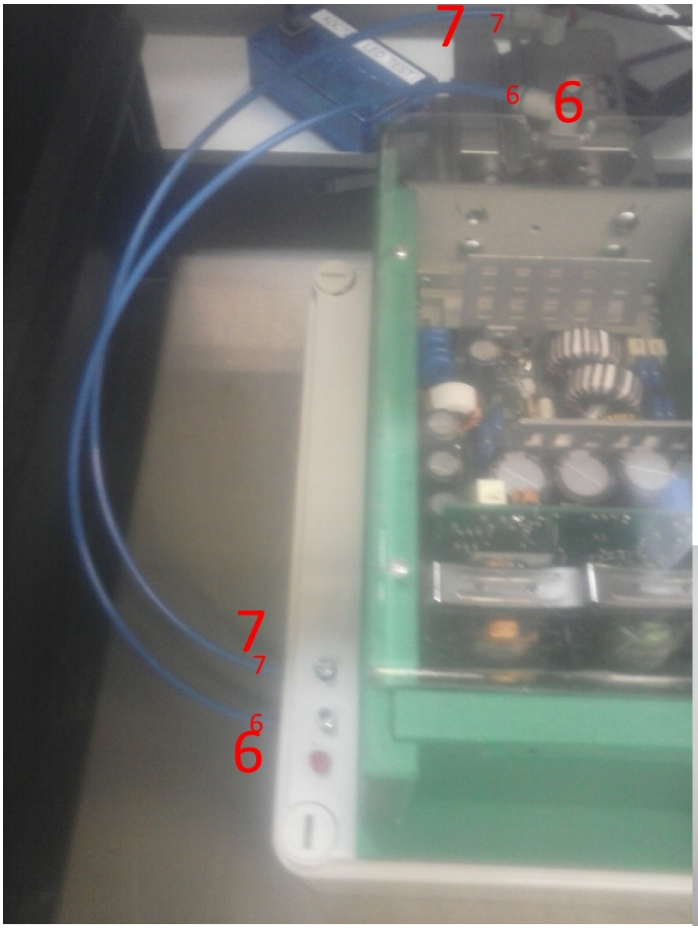


1. Testri programmi valimine

9. Lülita sisse tester ja arvuti. Vaata, et nad oleksid ühendatud USB kaabliga

10. Vaata, et rakis oleks ühendatud. Juhtmed peavad olema ühendatud järgmiselt:

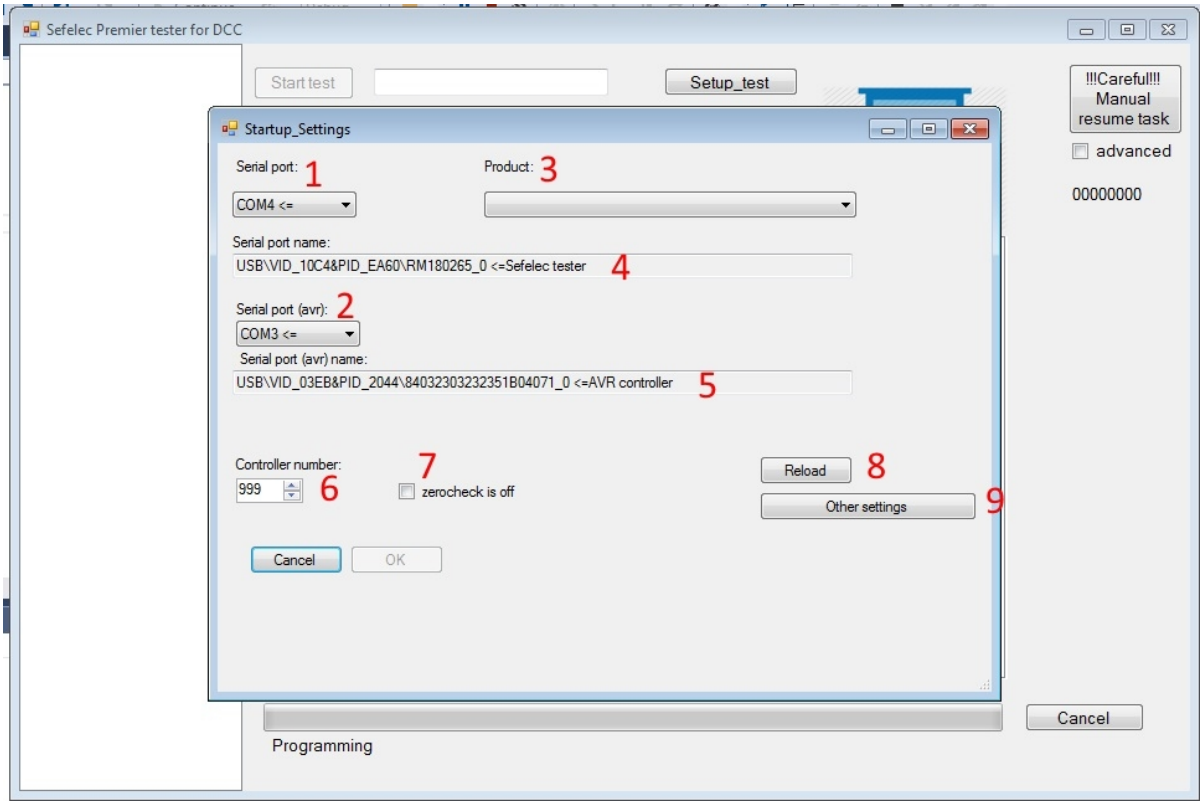




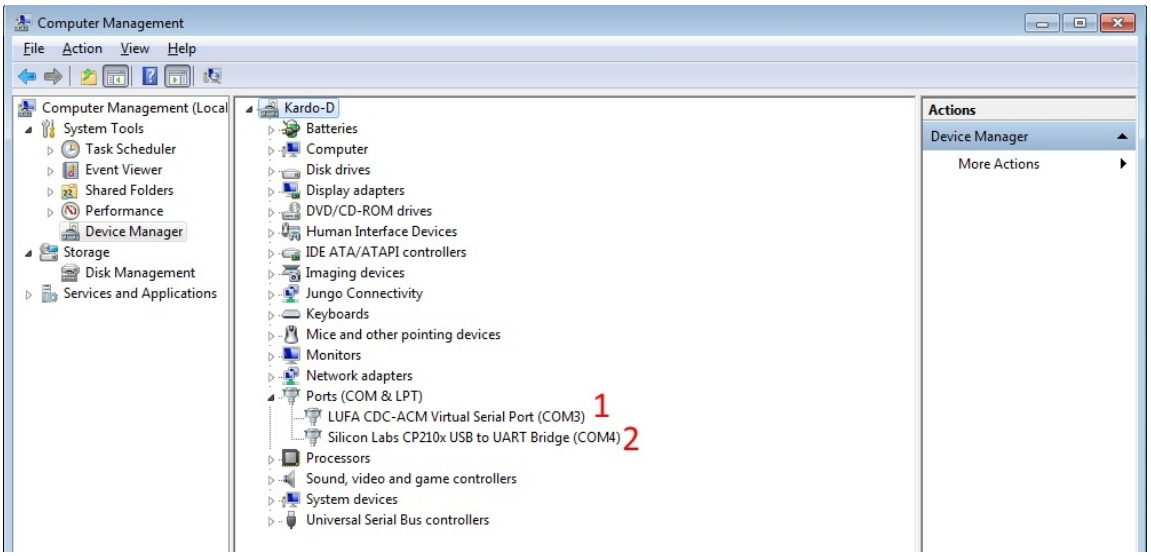
<1> AVR mikrokontrolleri juhe (vt alumist parempoolset pilti), peab olema ühendatud arvutiga. <2> toide, peab olema vooluvõrgus. <3> testri otsik, peab olema kindlalt kinnitatud(vt alumist keskmist pilti) <4> õhu sissetulek, peab olema ühendatud ja ventiil avatud (käepide voolikuga samas suunas, vt alumist vasakpoolset pilti). <5> testri maandus, ühendatakse rakisega ilma krokodill otsikuta. <6> <7> õhuvoolikud silindrisse.

Soovitavalt peaks tester soojenema 30min enne testidega alustamist, et tagada parim tulemus. Kui toode ei läbi testi ja sisse lülitamisest on möödas alla 30 minuti, siis oota kuni aeg täis saab ja proovi uuesti.

11. Käivita arvutis programm ’Sefelec tester_DDC’. Avaneb selline pilt:



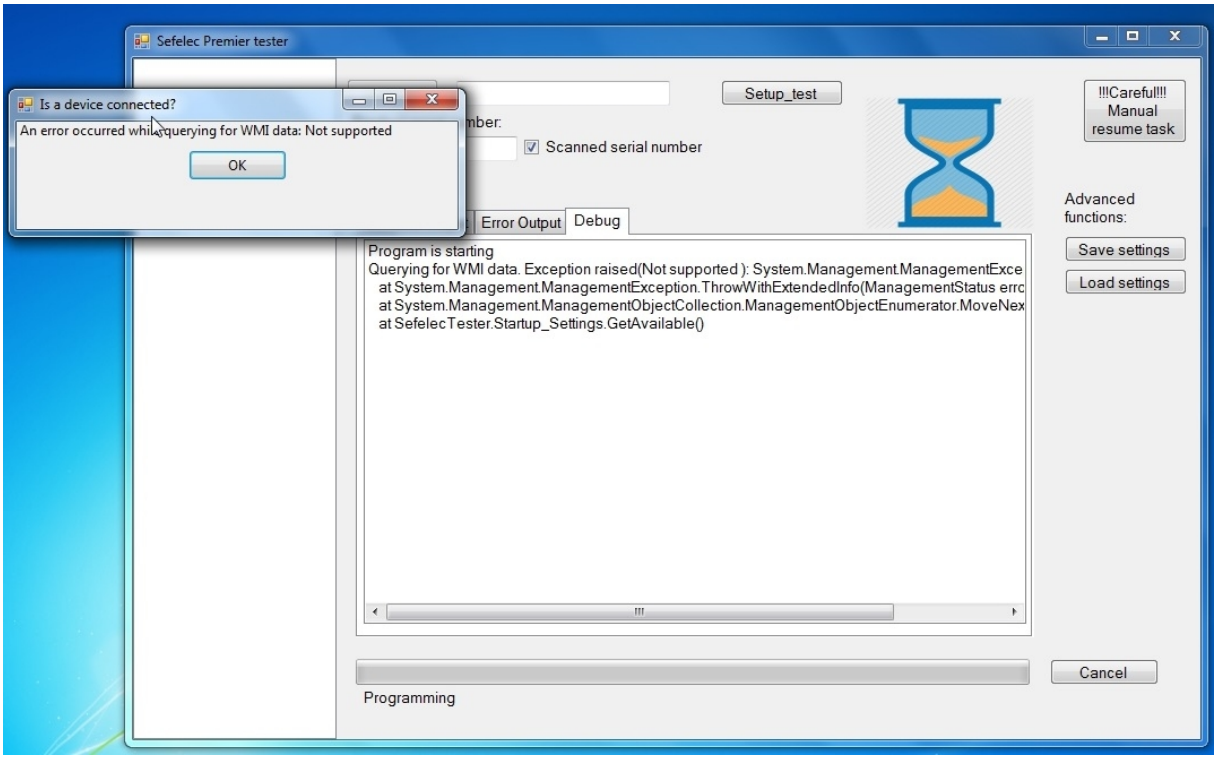
12. <1>- vali testeri COM port (nool näitab varianti, mis on tõenäoliselt õige); <2> vali AVR kontrolleri COM port (nool näitab varianti, mis on tõenäoliselt õige); <3>- vali kasutatav test; <4>- testeri COM pordi andmed (Sefelec'i toodete puhul on lisatud vastav märge); <5> AVR kontrolleri COM pordi andmed (Atmel kontrollerit puhul on lisatud vastav märge); <6>- vali oma id number; <7>- vali, kas kasutad 'zerocheck' funktsiooni (NB! tavaliselt ei kasutata); <8>- lae uuesti praegune aken; <9>- lisafunktsioonid.
13. Vali kasutatavad COM pordid <1> ja <2>. Kui sa ei tea, millist valida, siis vaata järgi 'Device managerist'.



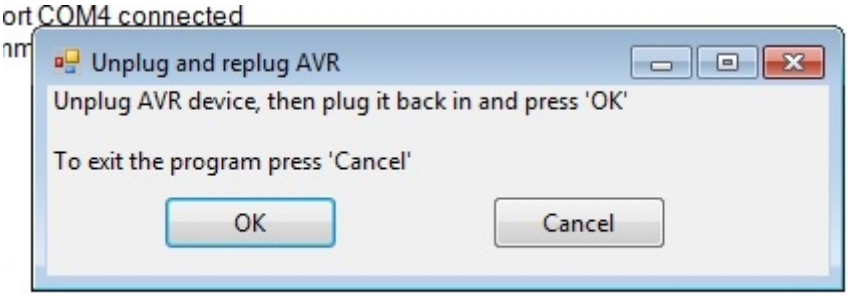
14. Vali testitav toode <3>.
15. Sisesta oma id number <6>.
16. Kui vaja teostada ’zerocheck’ siis vali vastav kast <7>.
17. Vajuta ’OK’ või klaviatuurilt ’Enter’. Kui kõik läheb hästi, jätka sammuga ’Testri programmeerimine’.

Kui nimekirjas ei ole toote testi, siis tuleb see lisada (räägi inseneriga). Kui nimekirjas ei ole õiget COM porti, siis kontrolli, et juhtmed oleksid arvutiga korralikult ühendatud ja vajuta ’Reload’ <8> ning tee uuesti sammud 4 kuni 9.

3.1 Võimalikud vead



Tester ei ole sisse lülitatud või ei ole arvutiga ühendatud. Kontrolli üle ja paranda vead. Seejärel vajuta ’OK’ ja jälgi järgmise akna juhiseid.



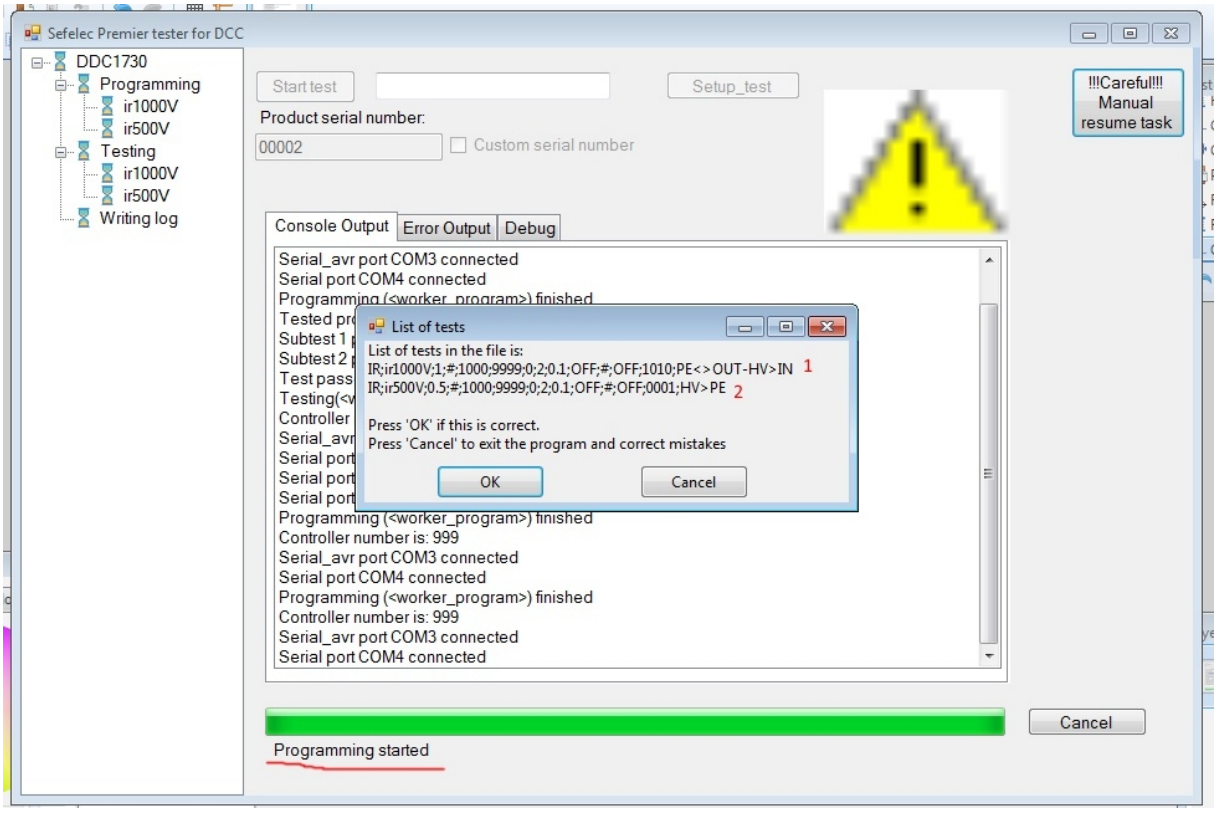
Probleem ühenduses kontrolleriga. Ühenda kontroller arvutist lahti ja ühenda uuesti. Seejärel vajuta ’OK’.

Probleem ühenduses testriga. Ühenda tester arvutist lahti ja ühenda uuesti. Seejärel vajuta ’Yes’.

2. Testri programmeerimine

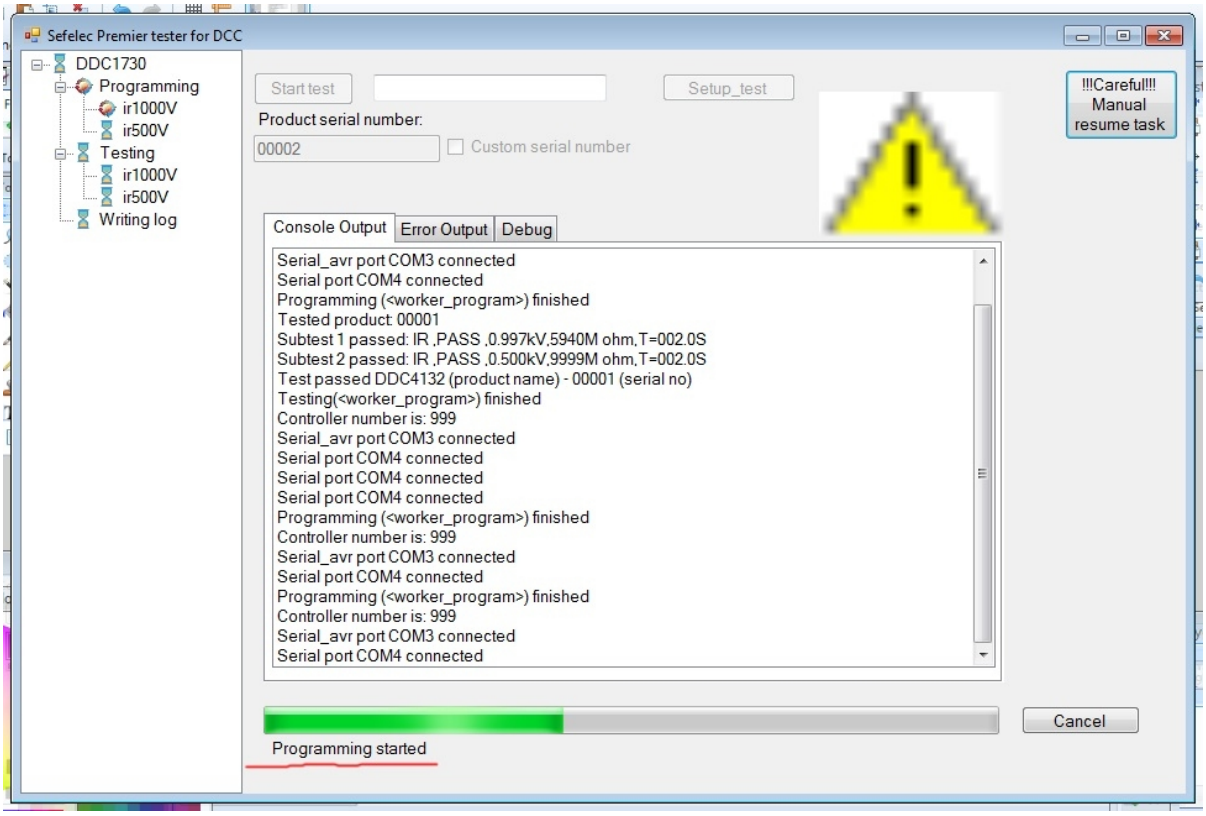
Automaatselt algab testri programmeerimine. Kui see ei alga, siis sulge programm ja alusta uuesti.

Kui programmeerimine algab, siis avaneb hetke pärast järgmine pilt:

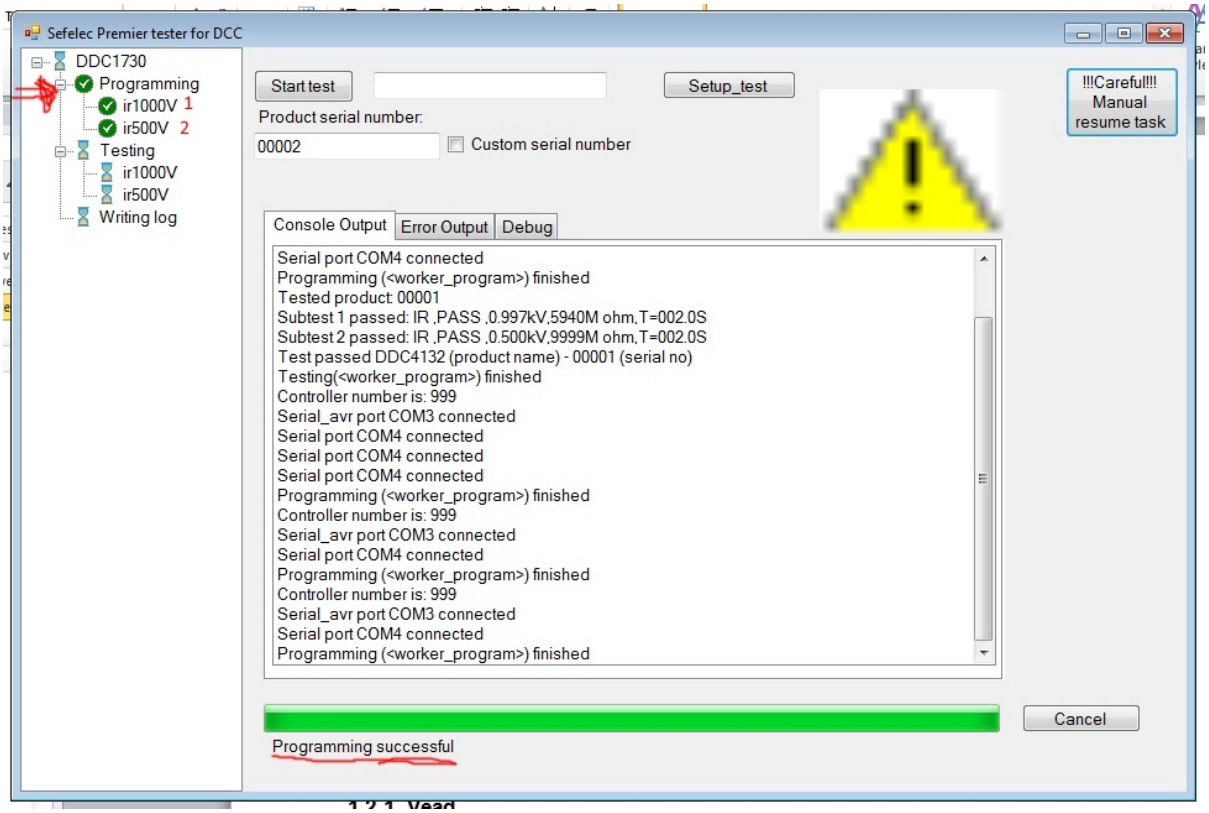


3. Kontrolli, kas testide andmed (1. ja 2., teste võib olla ka muu arv) on õiged. Kui nad vastavad tootele, siis vajuta ’OK’.

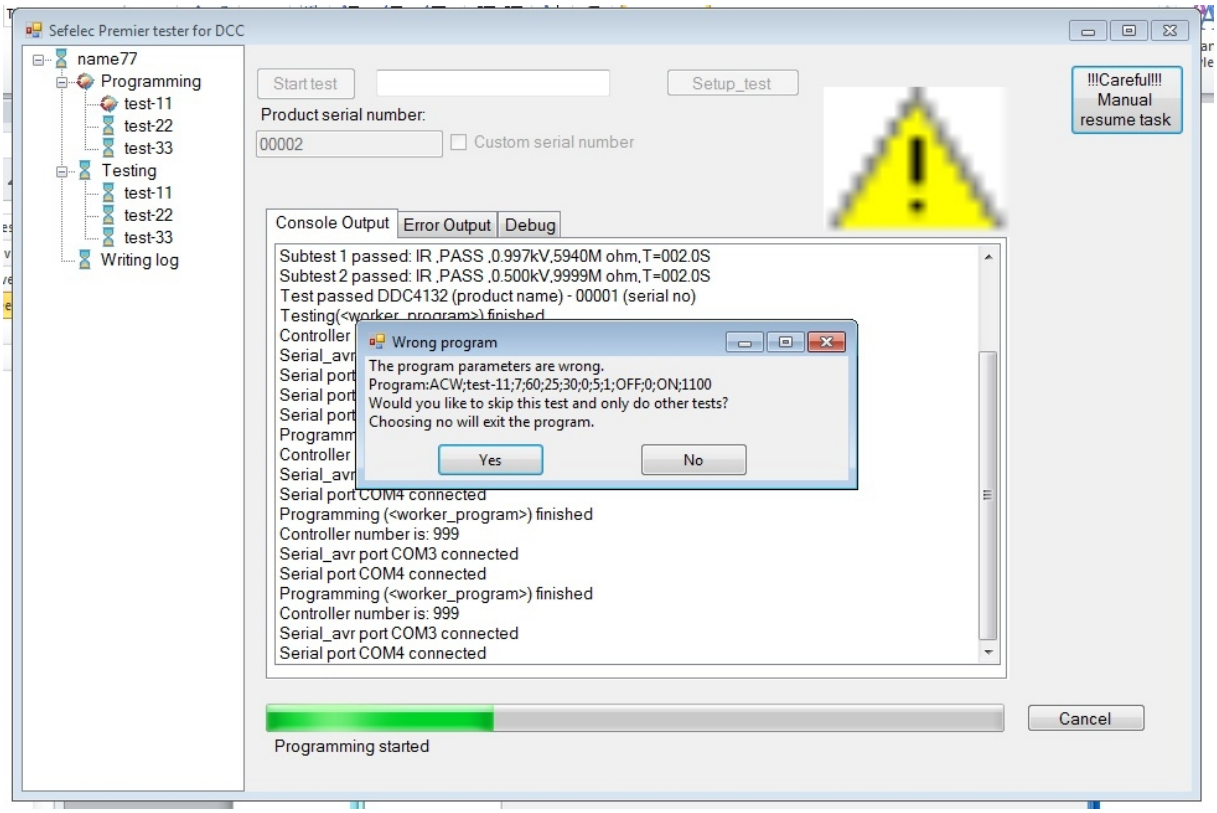
Avaneb selline pilt:



4. Oota kuni lõpeb programmeerimine ja avaneb selline pilt (’Programming successful’):


Sefelec Premier tester for DCC
DDC1730
Programming
ir1000V 1
ir500V 2
Testing
ir1000V
ir500V
Writing log
Start test
Setup_test
Product serial number:
00002
Custom serial number
!!!Careful!!!
Manual
resume task
Console Output
Error Output
Debug
Serial port COM4 connected
Programming (<worker_program>) finished
Tested product: 00001
Subtest 1 passed: IR ,PASS ,0.997kV,5940M ohm,T=002.0S
Subtest 2 passed: IR ,PASS ,0.500kV,9999M ohm,T=002.0S
Test passed DDC4132 (product name) - 00001 (serial no)
Testing(<worker_program>) finished
Controller number is: 999
Serial_avr port COM3 connected
Serial port COM4 connected
Serial port COM4 connected
Serial port COM4 connected
Programming (<worker_program>) finished
Controller number is: 999
Serial_avr port COM3 connected
Serial port COM4 connected
Programming (<worker_program>) finished
Controller number is: 999
Serial_avr port COM3 connected
Serial port COM4 connected
Programming (<worker_program>) finished
Cancel
Programming successful

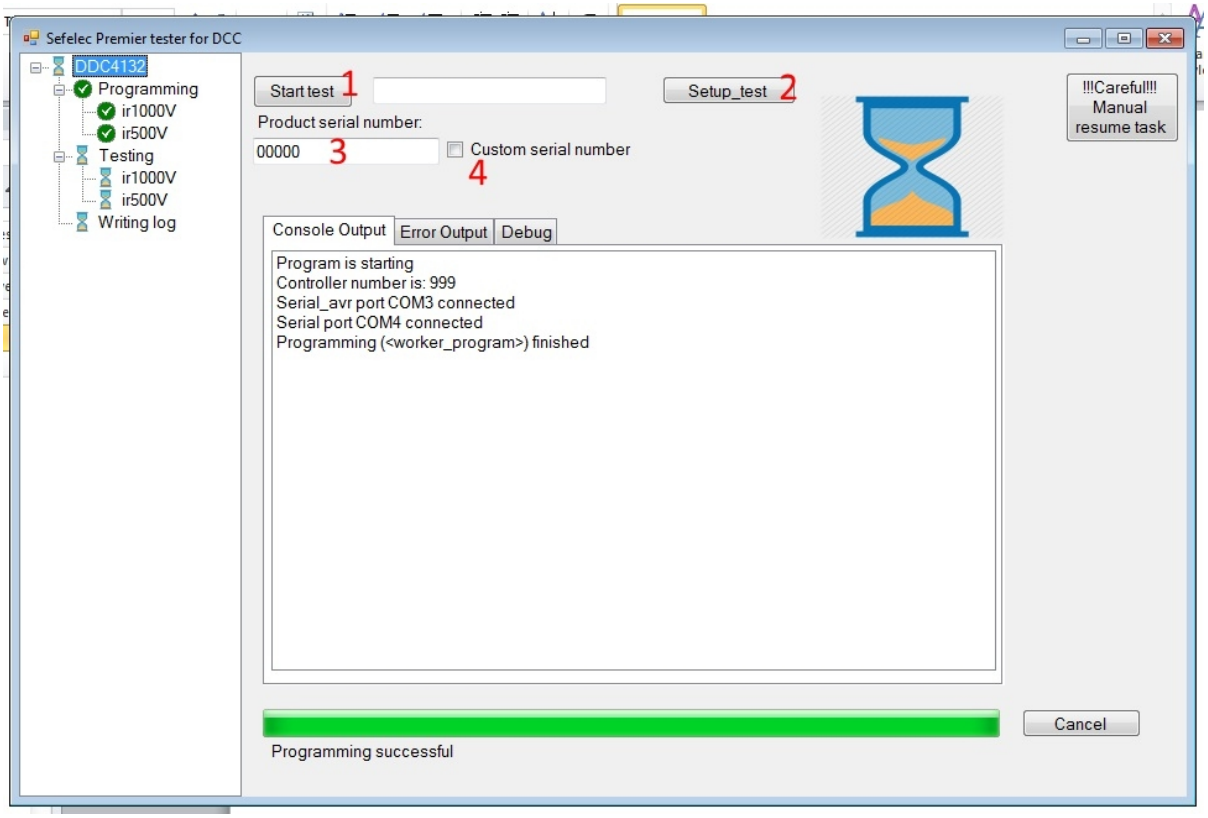
4.1 Vead


Sefelec Premier tester for DCC
name77
Programming
test-11
test-22
test-33
Testing
test-11
test-22
test-33
Writing log
Start test
Setup_test
Product serial number:
00002
Custom serial number
!!!Careful!!!
Manual
resume task
Console Output
Error Output
Debug
Subtest 1 passed: IR ,PASS ,0.997kV,5940M ohm,T=002.0S
Subtest 2 passed: IR ,PASS ,0.500kV,9999M ohm,T=002.0S
Test passed DDC4132 (product name) - 00001 (serial no)
Wrong program
The program parameters are wrong.
Program:ACW;test-11;7;60;25;30;0;5;1;OFF;0;ON;1100
Would you like to skip this test and only do other tests?
Choosing no will exit the program.
Yes
No
Serial port COM4 connected
Programming (<worker_program>) finished
Controller number is: 999
Serial_avr port COM3 connected
Serial port COM4 connected
Programming (<worker_program>) finished
Controller number is: 999
Serial_avr port COM3 connected
Serial port COM4 connected
Cancel
Programming started

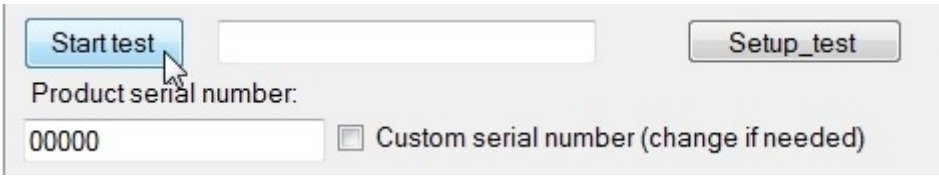
Programmi parameetrid on valed. Kui näidatav test ei ole vajalik, siis vajuta ’Yes’ ja testimisel ei kasutata seda testi. Peaaegu alati on kõik testid vajalikud. Sel juhul vajuta ’No’, välju programmist ja räägi inseneriga, et parandada testi parameetrid.

3. Testimine

Avaneb järgmine pilt:



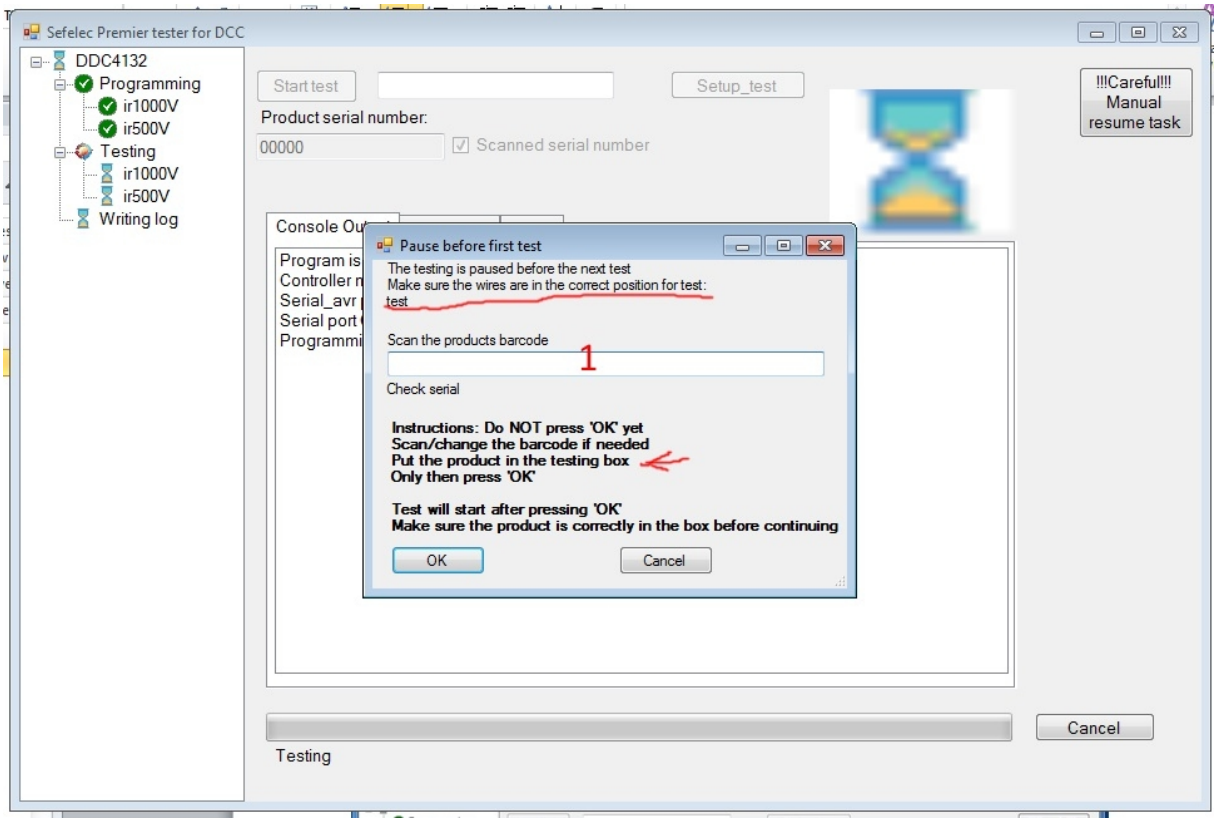
6. <4> Vali, kas skänneerid toote seerianumbri või sisestatakse järjestikused numbrid. Kui tootel ei ole skaneeritavat seerianumbrit, siis eemalda valik <4> ja sisesta nulltoote seerianumber <3>. Sel juhul on iga toote number 1 võrra suurem kui eelmisel. Esimese toote number on sisestatud nulltoote number+1.



7. Kui oled testiks valmis, vajuta (<1>) ’Start test’.

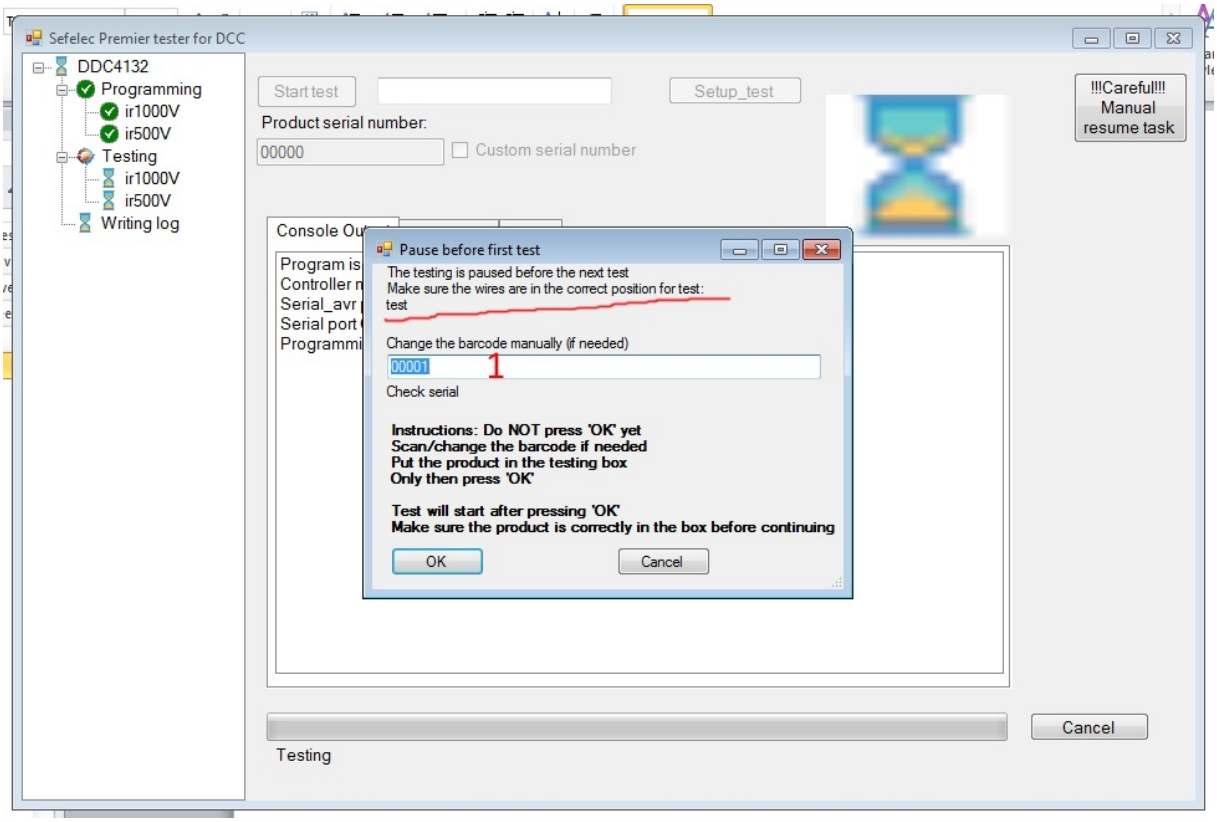
NB! ÄRA järgmises dialoogis veel ’OK’ vajuta.

Kui kasutad skänneeritud seerianumbrit:



Peale koodi skänneerimist ilmub see kasti <1>.

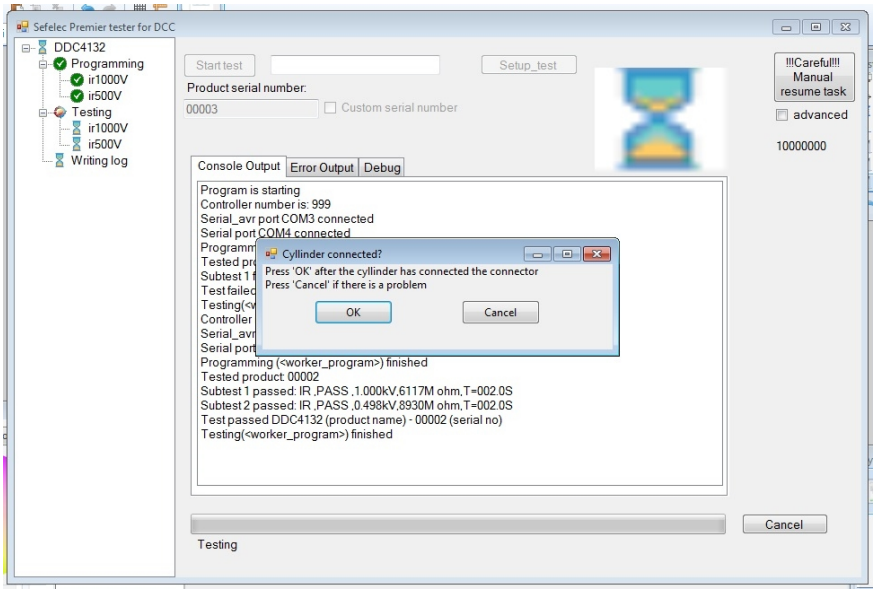
Kui kasutad teist varianti:



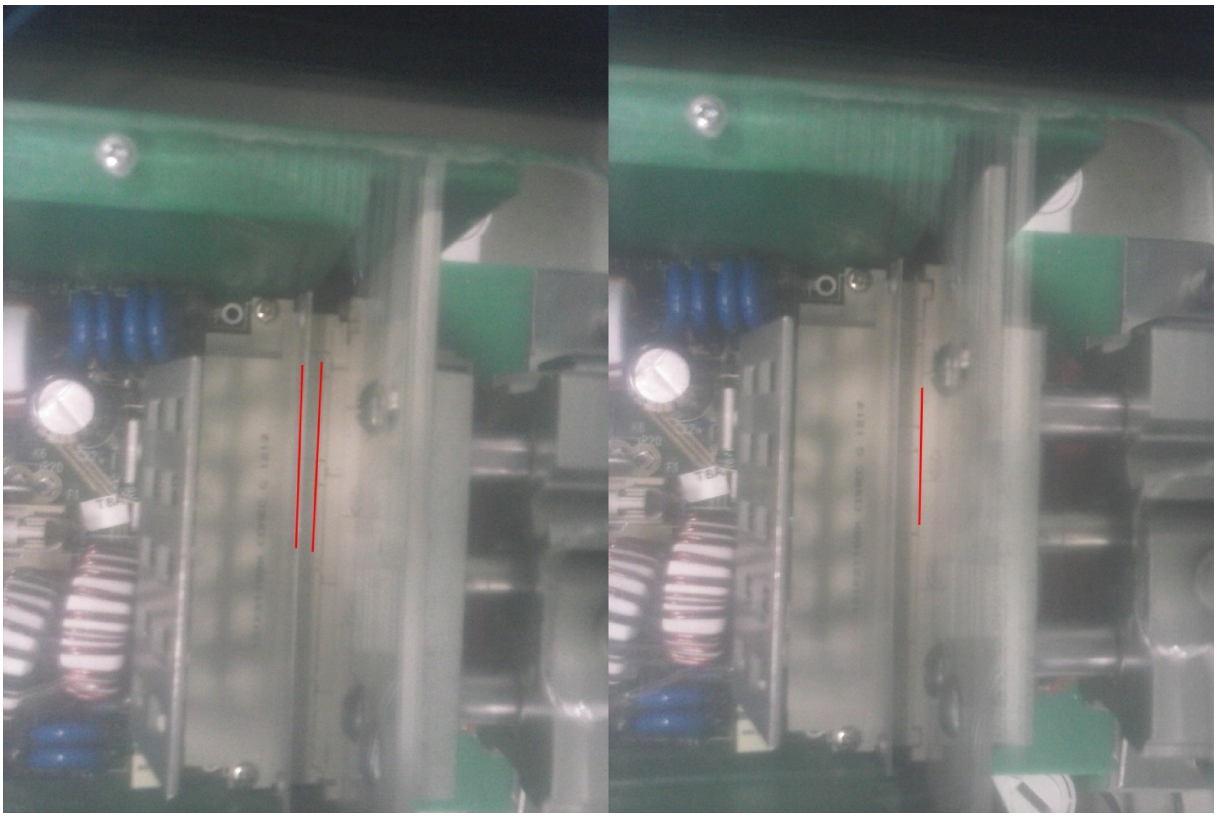
Toote kood on näha lahtris <1>.

Enne ’OK’ vajutamist peab toode olema rakises.

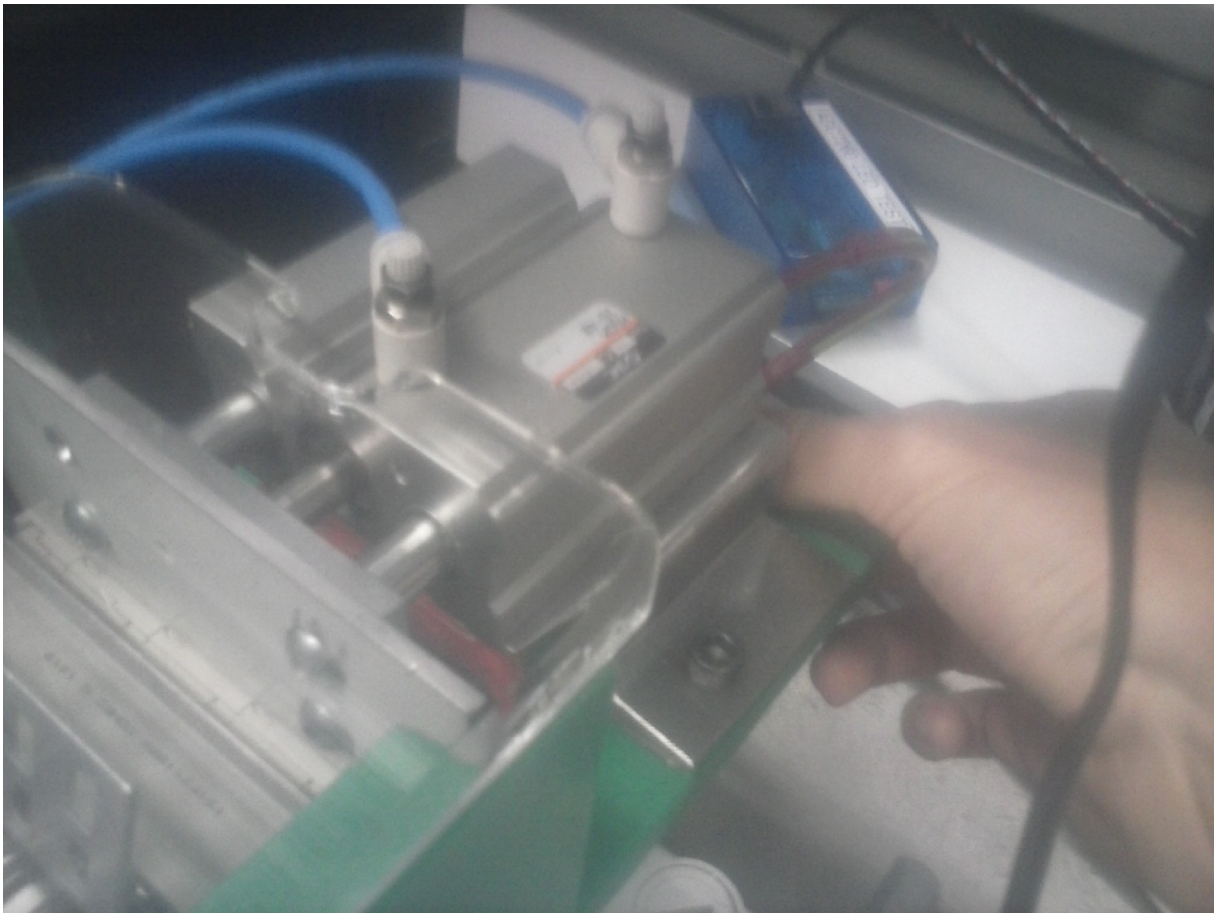
8. Sisesta toode rakisesse ja vajuta ’OK’.



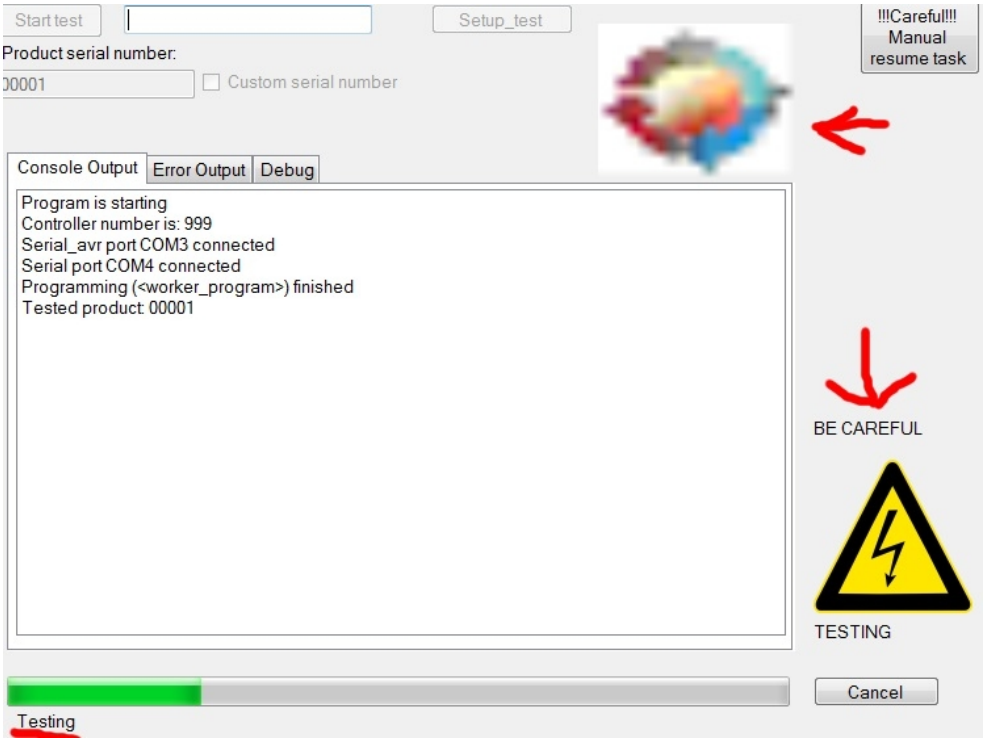
9. Kontrolli, et silinder oleks pistiku korralikult sisse lükanud. Vasakul pildil on näha vahe, mida olla ei tohi. Paremal pildil on pistik õiges asendis.



Kui pistik ei liigu ise õigesse asendisse, siis koputa sõrmedega tugevalt pistiku kohal olevale läbipaistvale plastikule või vajuta alla silindri tagumine ots niimoodi:



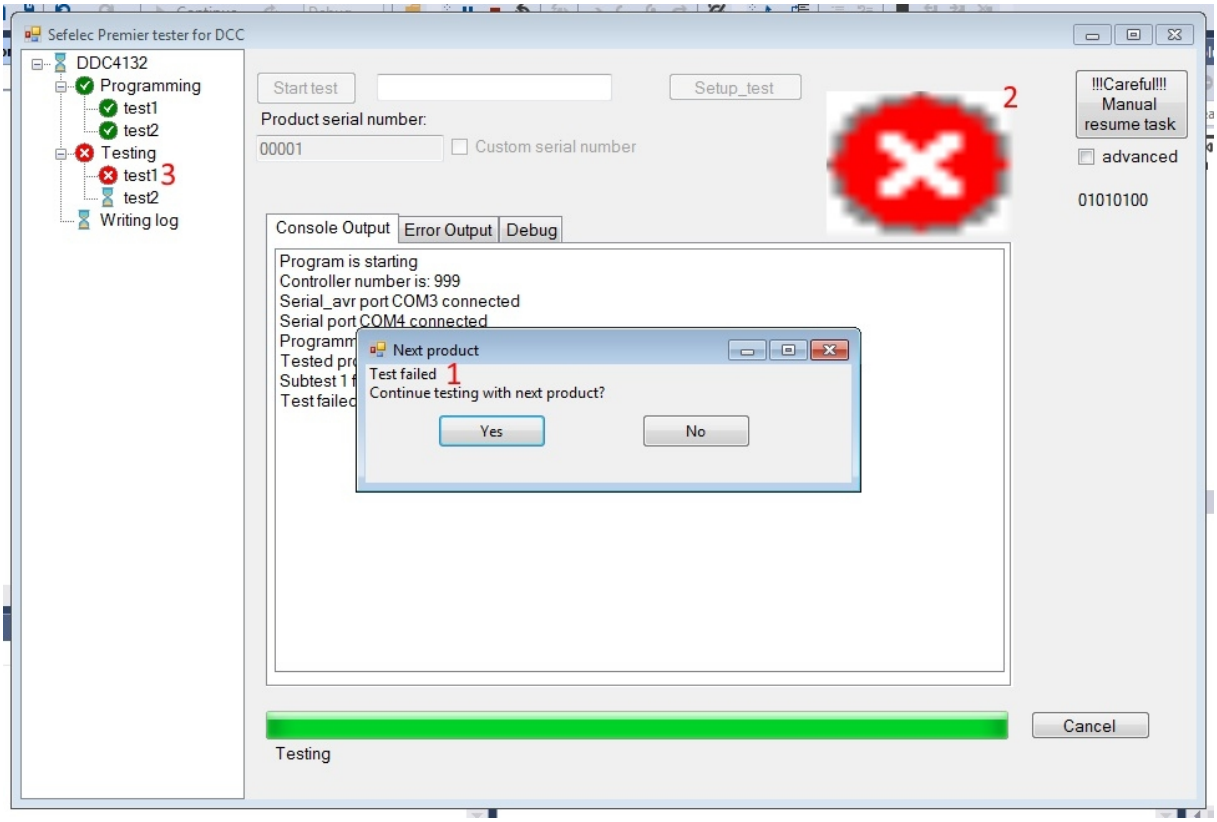
10. Kui pistik on õiges asendis, vajuta ’OK’.



Nähes seda ikooni on test käigus. Sellel ajal ei tohi liigutada rakist ega vajutada testri nuppe. Testi toimumist näitab see ikoon, vilkuv ’high voltage’ tuli testril ja kollane vilkuv tuli rakisel.

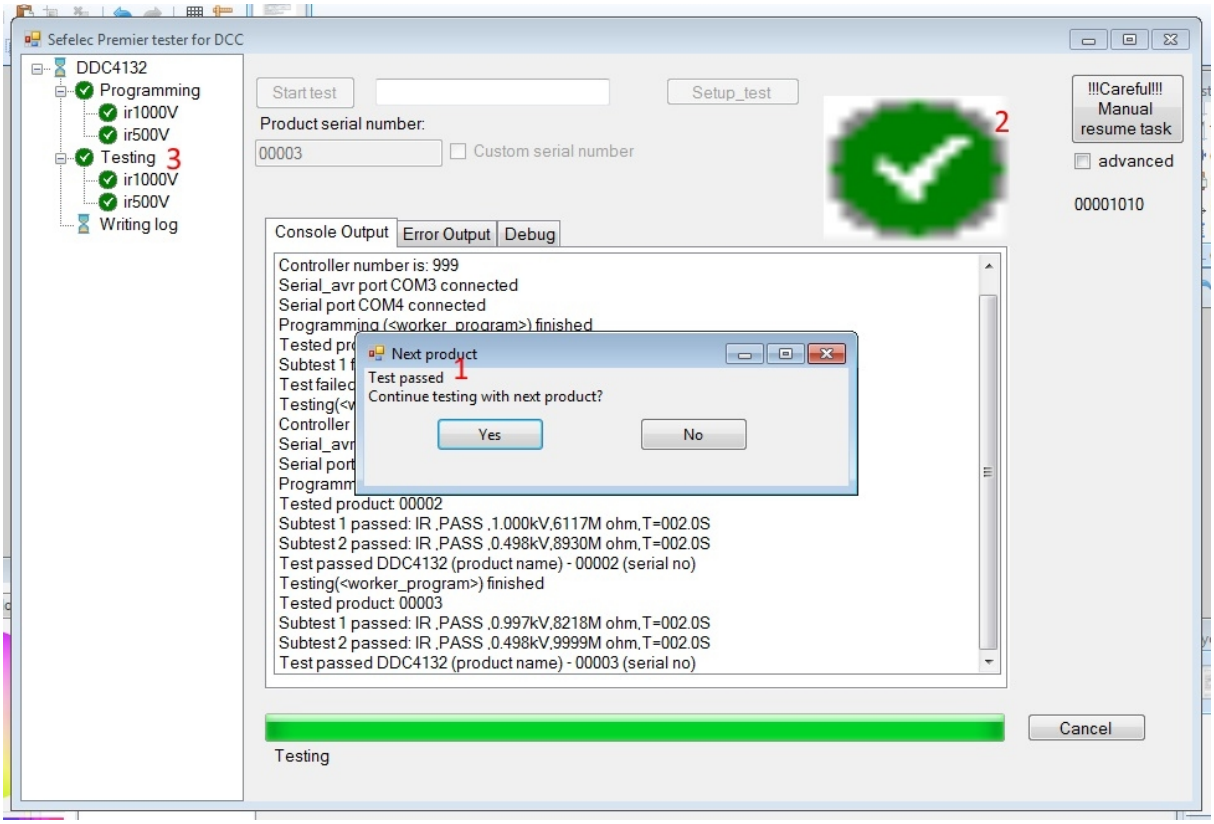
11. Oota kuni avaneb järgmine aken. **Testi ajal vilgub testril ’high voltage’ tuli. Sellel ajal ei tohi liigutada rakist ega vajutada testri nuppe!**

Kui test läbi kukub:



Testi tulemust näitavad 2 ikooni <2> ja <3> ning tuli rakisel (punane või roheline). Testi tulemus on kirjas aknas <1>. Testi ebaõnnestumisel jäetakse järgmised testid vahele. ’Yes’ vajutamine alustab järgmise toote testimist, test veel ei käivitu.

Viimase testi läbimisel:

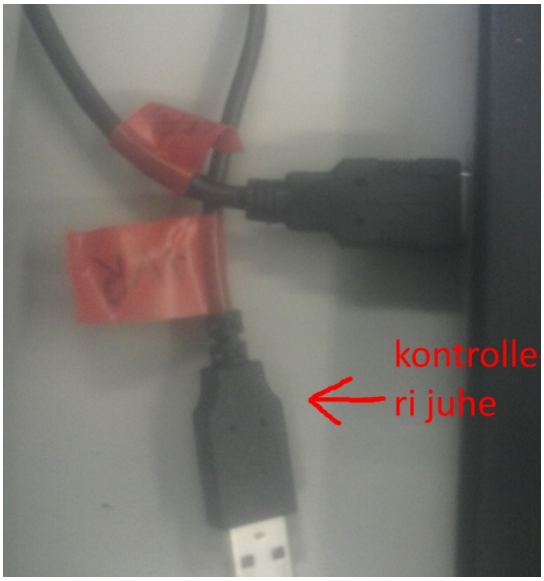


Kogu testi tulemus on kirjas <1> ja seda näitavad ka ikoonid <2> ja <3>. Vajutades ’Yes’ jätkub testimine järgmise tootega.

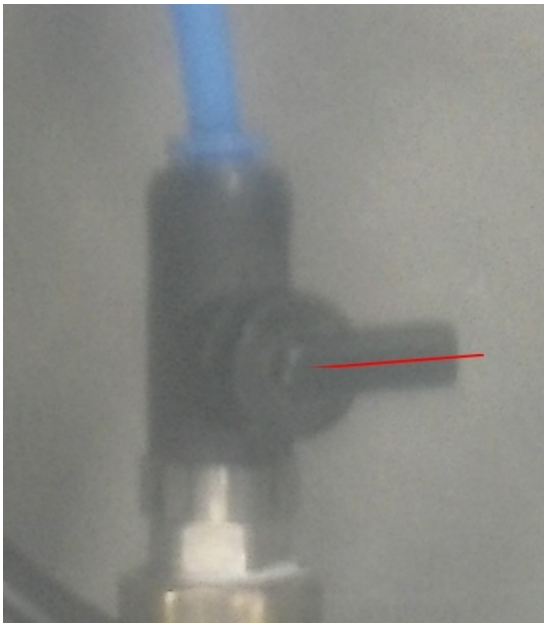
12. Võta toode rakisest välja. Kui silinder ei liigu täielikult tagasi, siis kasuta sama trikki nagu punktis 6.
13. Vajuta ’Yes’ või ’No’. Kui vajutad ’Yes’ jätka juhendiga peale punkti 2.

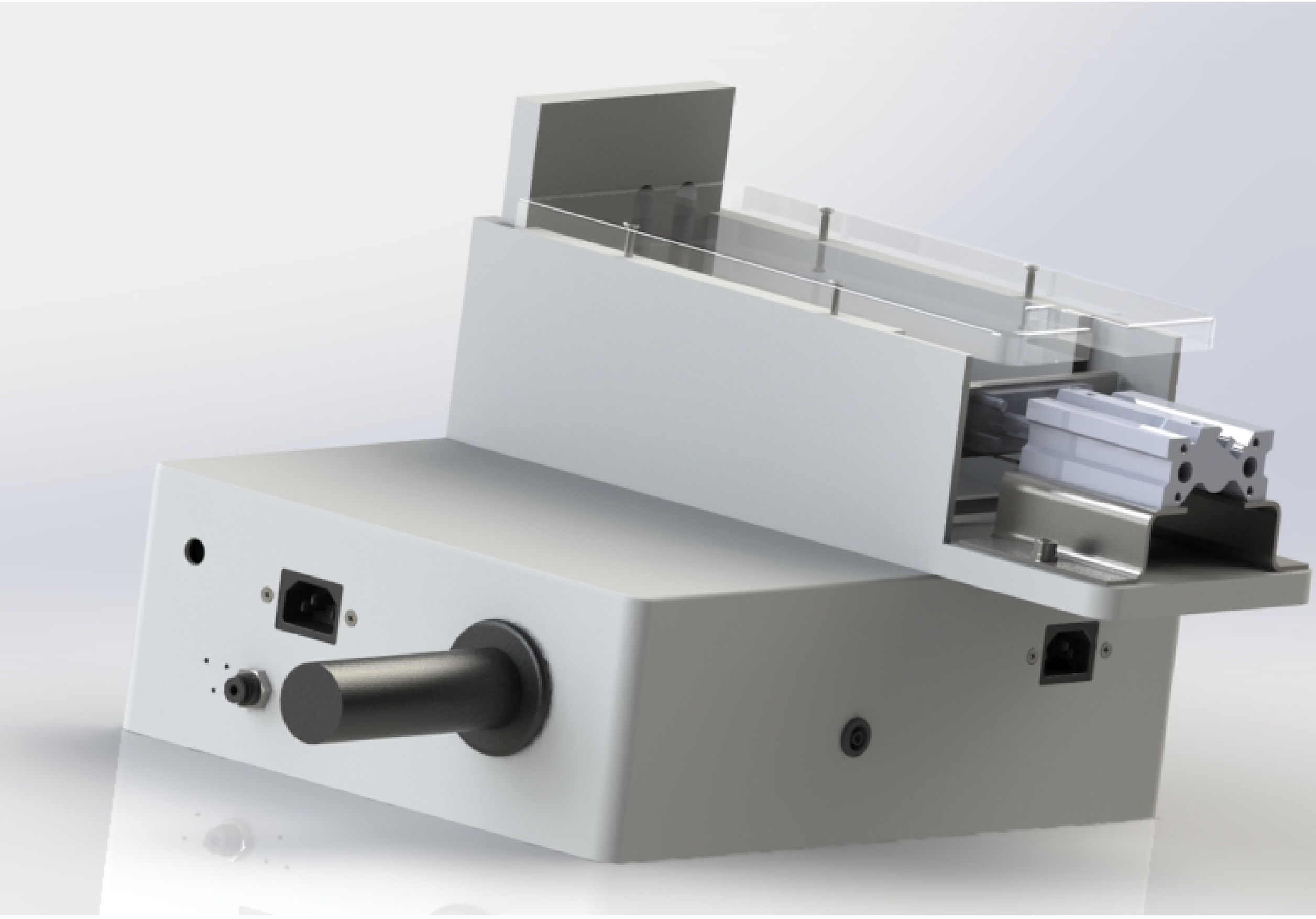
Kui lõpetad testimise, siis:

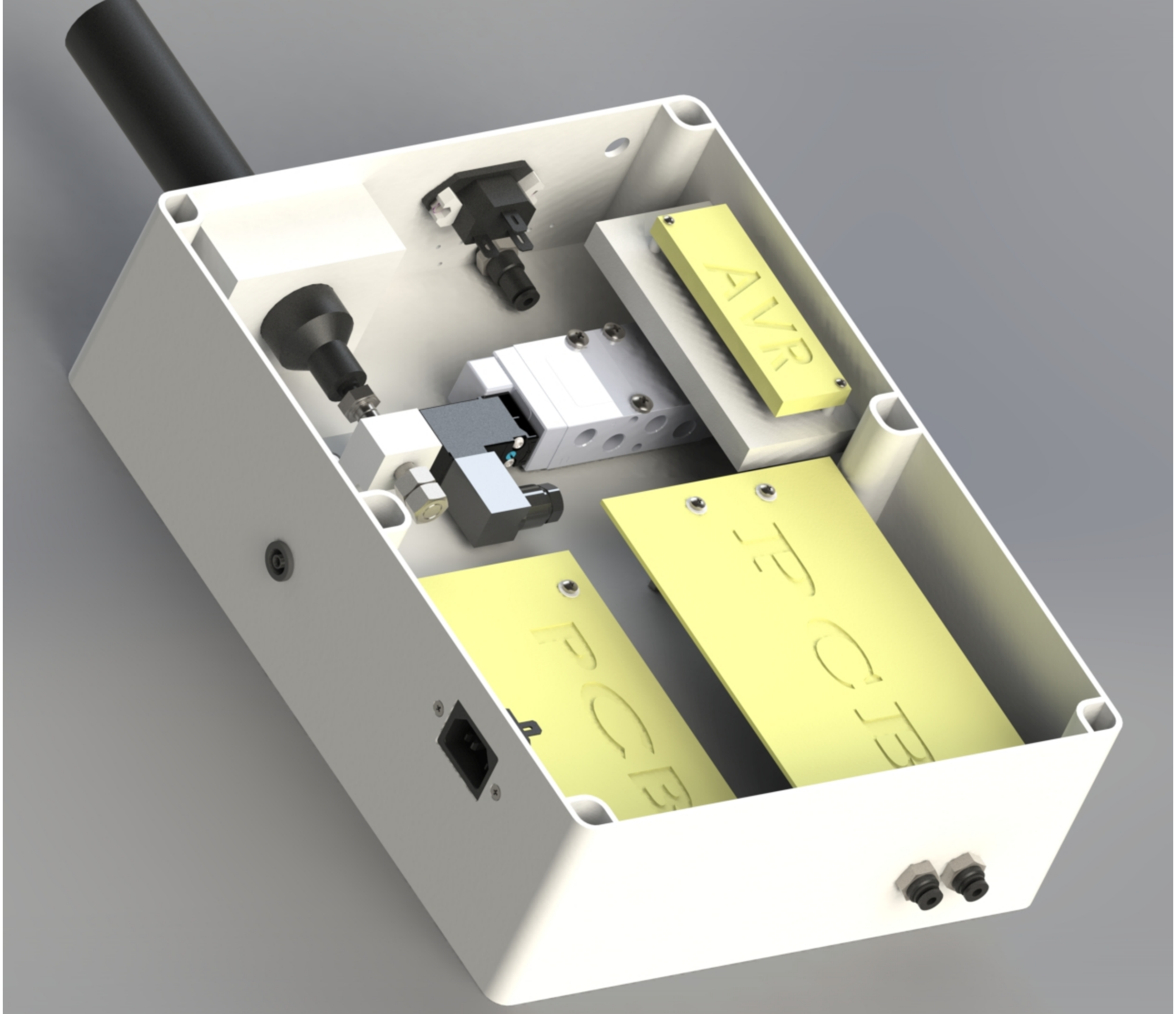
- Pane programm arvutis kinni.
- Ühenda lahti kontrolleri juhe arvutist:

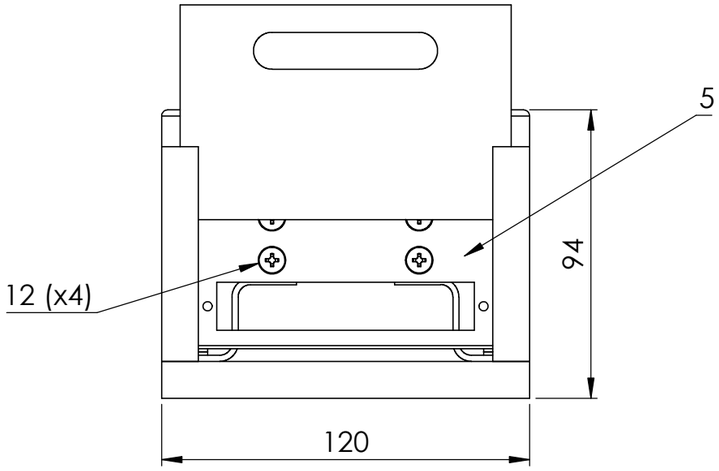


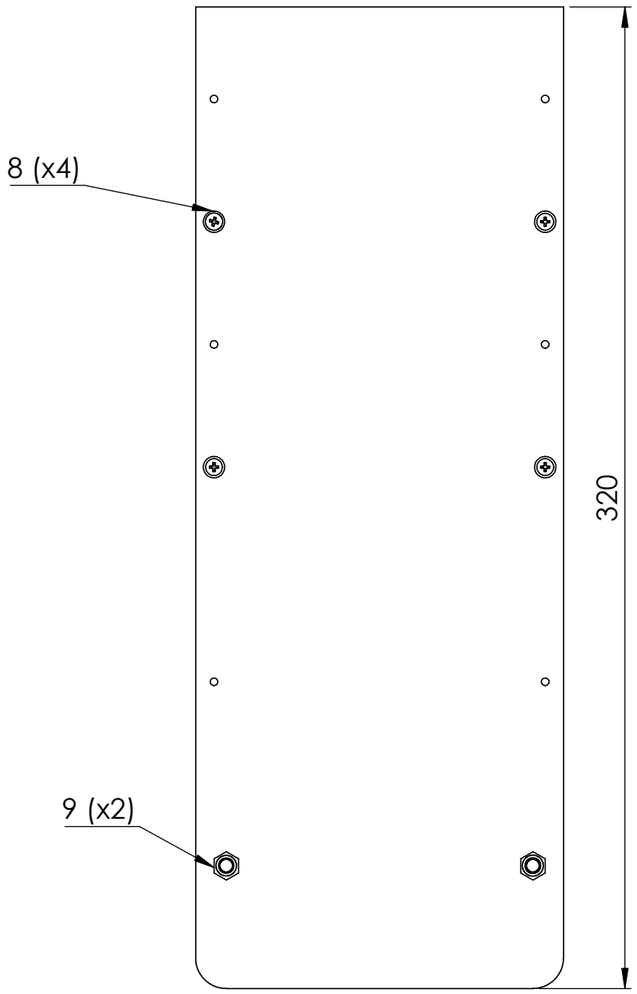
- Tõmba välja rakise toitejuhe.
- Pane õhu ventiil kinni:

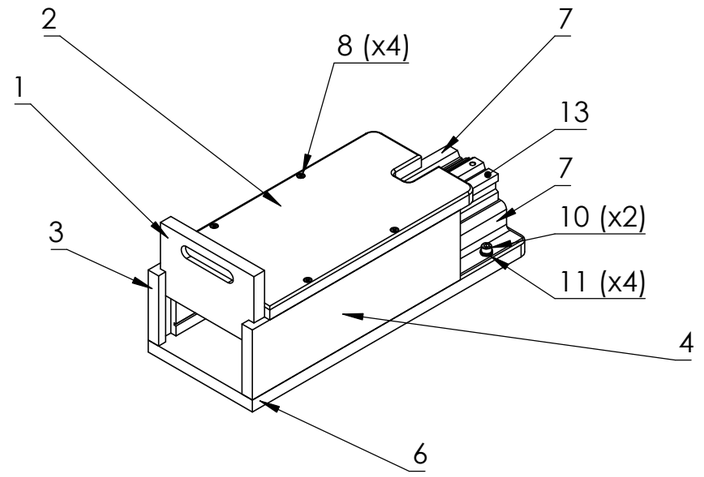





AVR

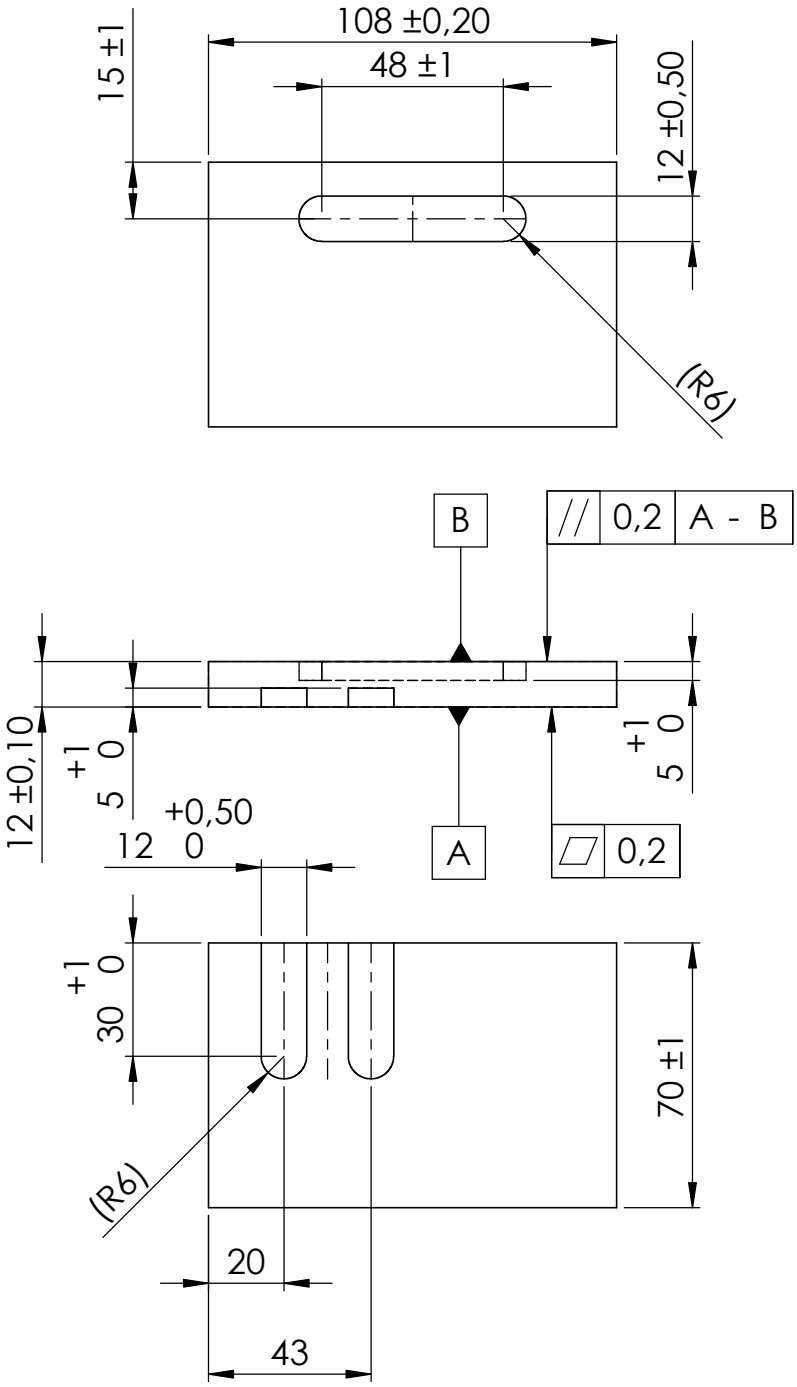




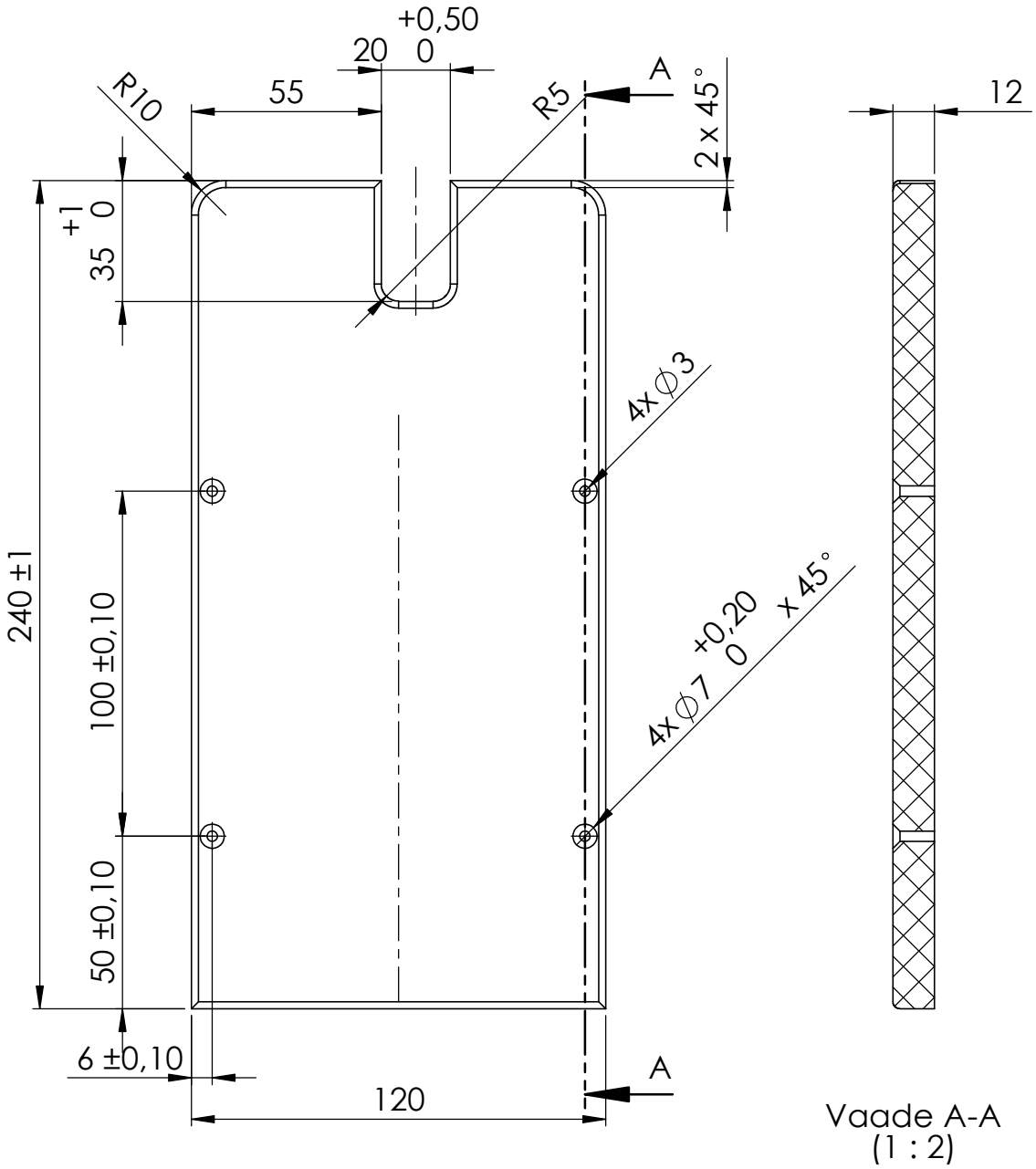


13		Silinder SMC MGQM 12-40		1	
12		Kruvi M4 x 16 mm		4	
11		Seib 5 mm x 2 mm		4	
10		Polt M5 x 16 mm		2	
9		Mutter M5		2	
8		Kruvi M3 x 25 mm		8	
7		Silindri jalg, AISI 316	P1.2.6	2	
6		Põhi, ABS	P1.2.5	1	
5		Pistiku hoidik, AISI 316	P1.2.4	1	
4		Külg parem, ABS	P1.2.3 peegelpilt	1	
3		Külg vasak, ABS	P1.2.3	1	
2		Kate, ABS	P1.2.2	1	
1		Kaan, ABS	P1.2.1	1	
Osa	*Väli*	*Nimetus, materjal*	*Tähis*	*Hulk*	*Märkus*
		Materjal	*Märkimata piirhälbed*	*Mass* 1,67	*Mõõt* 1:2

Teostas	*Kardo Aia*	*Nimetus*	
Kinnitas	*Maido Hiiemaa*	Silindri rakis	
Kontrollis	*Maido Hiiemaa*		
TALLINNA TEHNIKAÜLIKOOL *Mehhatroonikainstituut*		*Leht* 1	*Tähis* P1.2



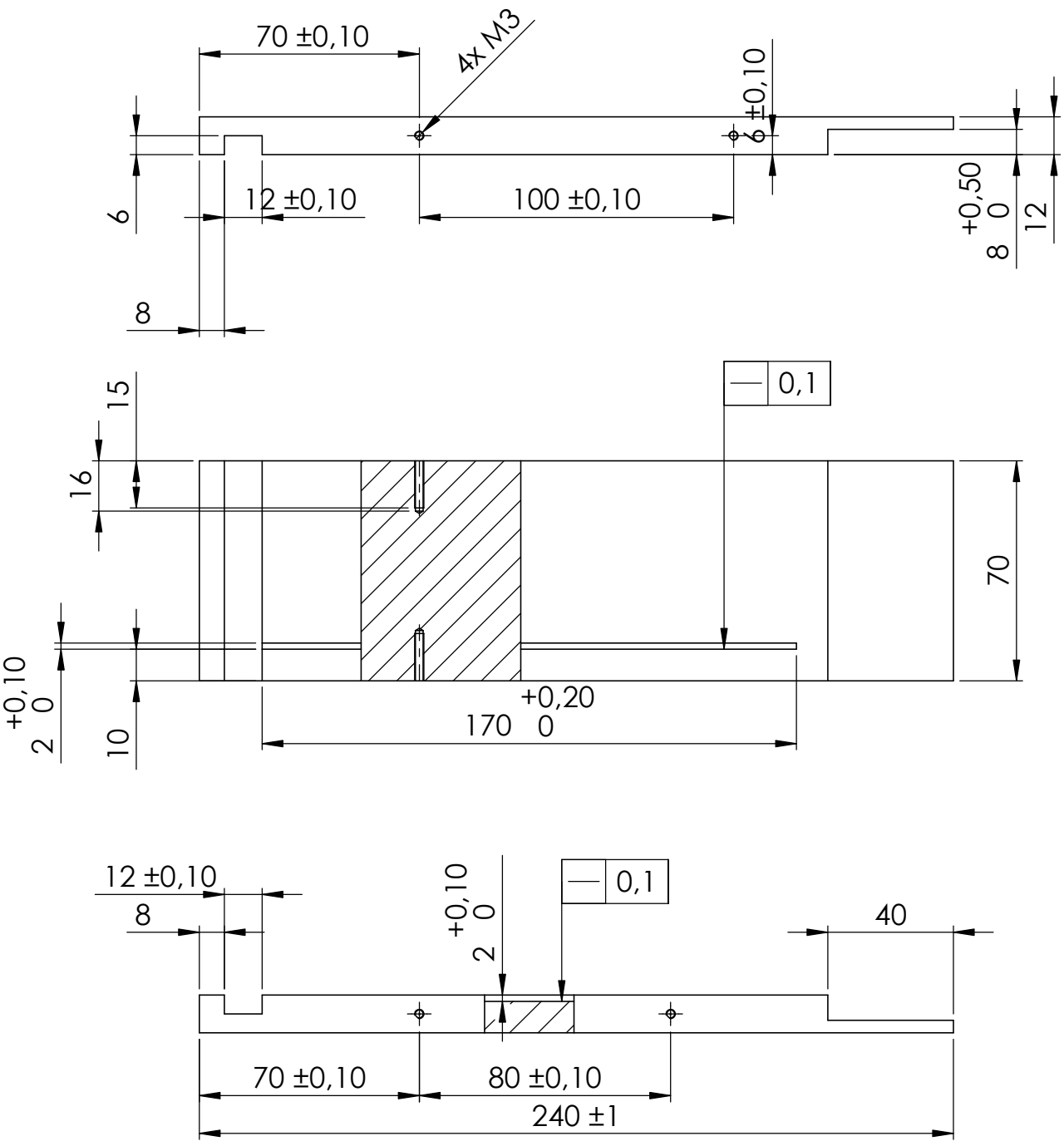
	Materjal ABS	*Märkimata piirhälbed* ISO 2678-m Ra 6,3	*Mass* 0,085	*Mõõt* 1:2
Teostas	*Kardo Aia*	*Nimetus* Kaan		
Kinnitas	*Maido Hiiemaa*			
Kontrollis	*Maido Hiiemaa*			
TALLINNA TEHNIKAÜLIKOOL *Mehhatroonikainstituut*		*Leht* 2	*Tähis* P1.2.1	



Vaade A-A
(1 : 2)

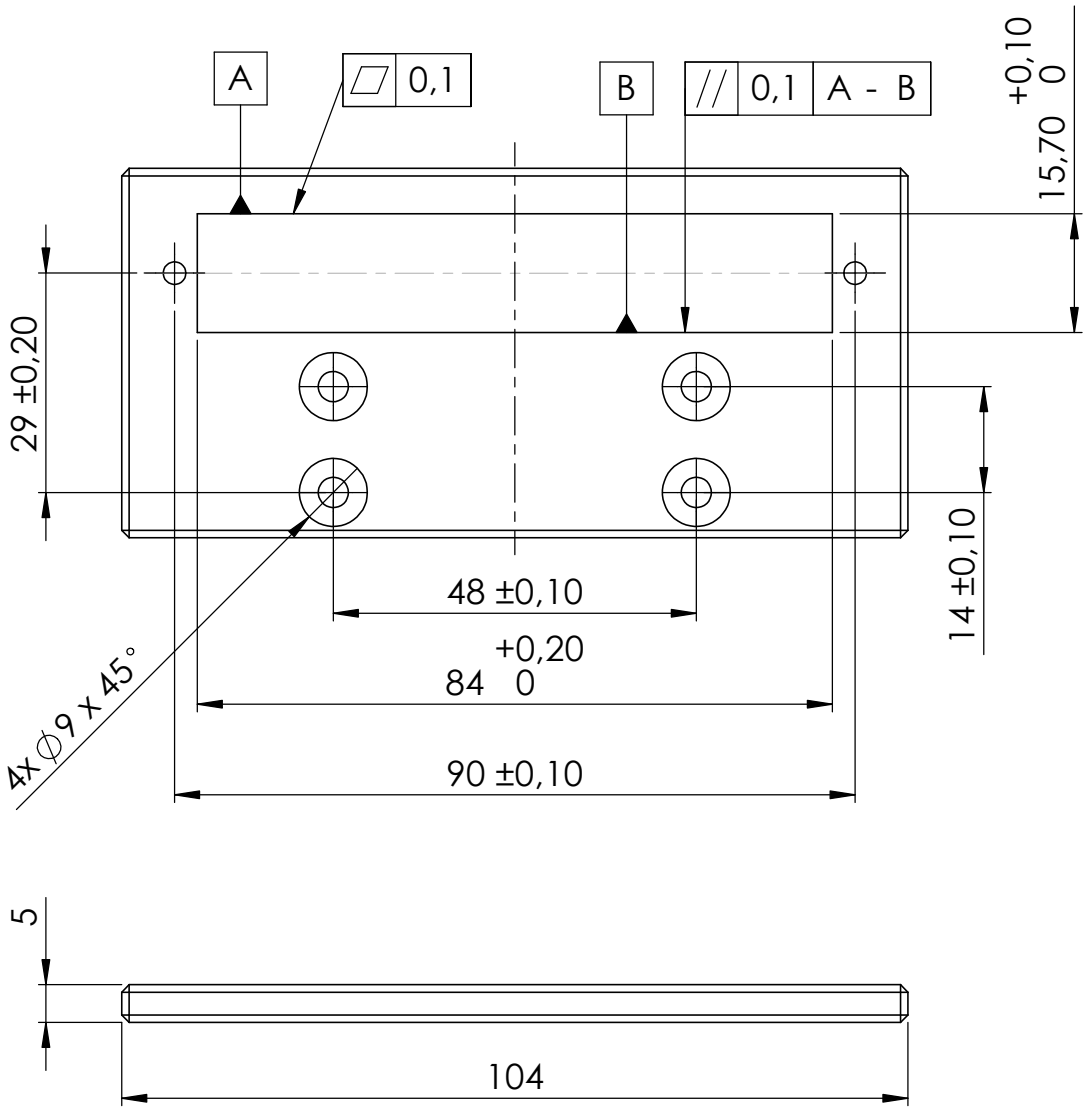
	Materjal	Märkimata piirhälbed	Mass	Mõõt
	ABS	ISO 2678-m Ra 6,3	0,36	1:2

Teostas	Kardo Aia	Nimetus	
Kinnitas	Maido Hiiemaa	Kate	
Kontrollis	Maido Hiiemaa		
TALLINNA TEHNIKAÜLIKOOL Mehhatroonikainstituut		Leht 3	Tähis P1.2.2



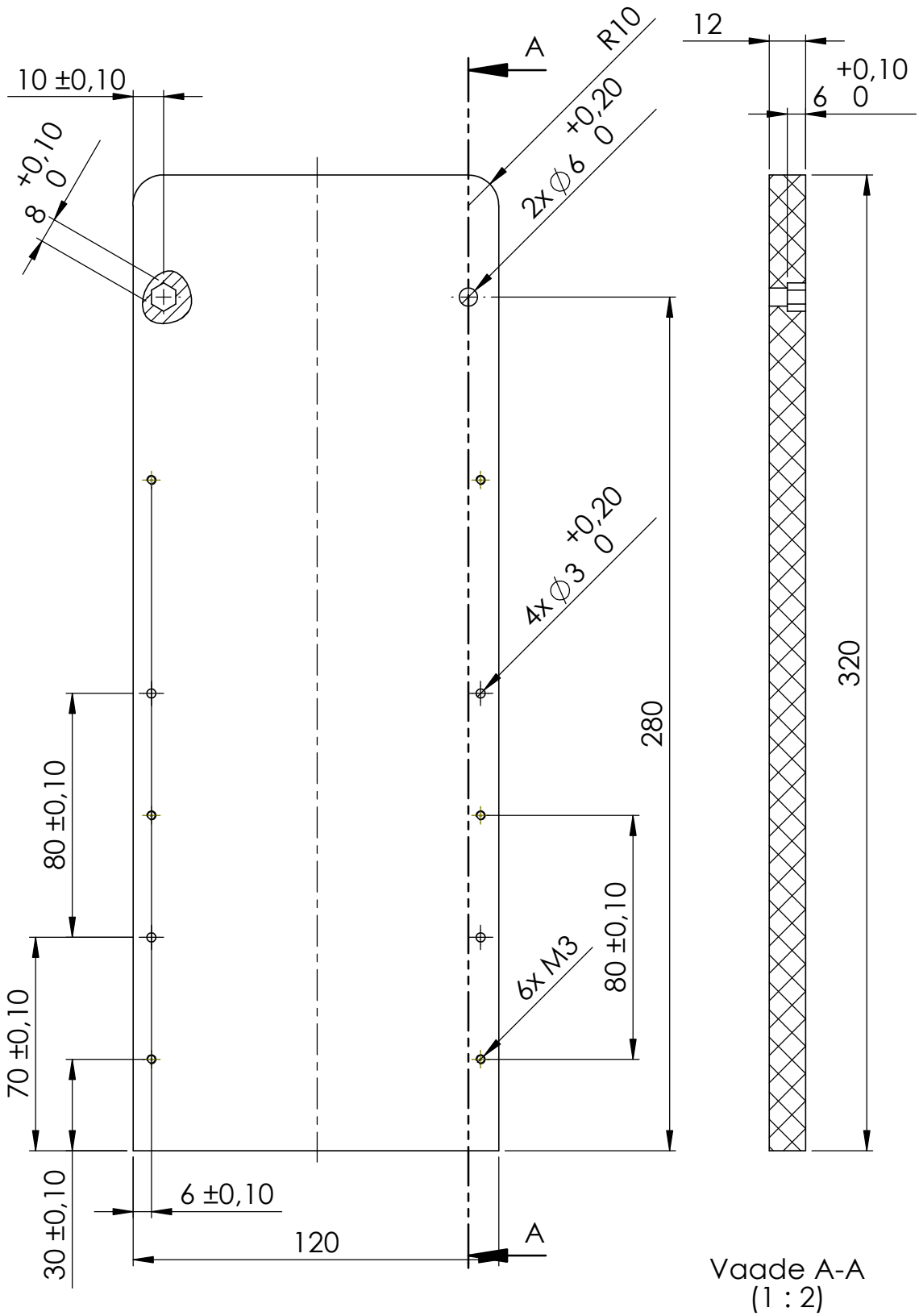
	Materjal	Märkimata piirhälbed	Mass	Mõõt
	ABS	ISO 2678-m Ra 6,3	0,18	1:2

Teostas	Kardo Aia	Nimetus	
Kinnitas	Maido Hiiemaa	Külg vasak	
Kontrollis	Maido Hiiemaa		
TALLINNA TEHNIKAÜLIKOOL Mehhatroonikainstituut		Leht 4	Tähis P1.2.3



	Materjal AISI 316	*Märkimata piirhälbed* ISO 2678-m Ra 6,3	*Mass* 0,12	*Mõõt* 1:1

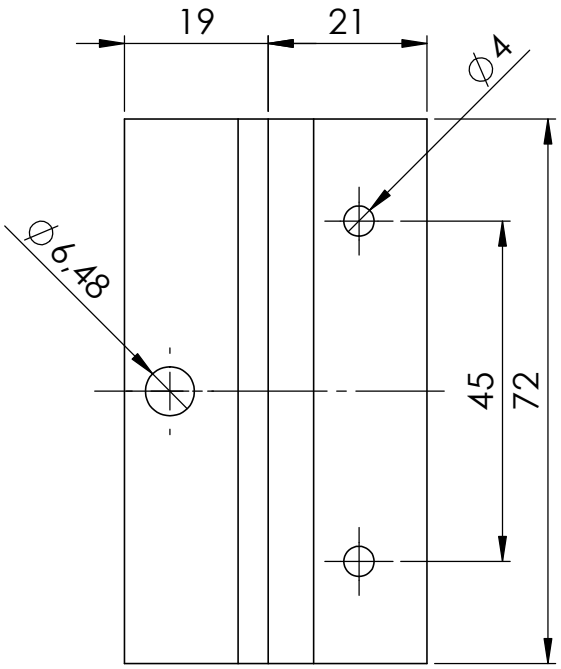
Teostas	*Kardo Aia*	*Nimetus*	
Kinnitas	*Maido Hiiemaa*	Pistiku hoidik	
Kontrollis	*Maido Hiiemaa*		
TALLINNA TEHNIKAÜLIKOOL *Mehhatroonikainstituut*		*Leht* 5	*Tähis* P1.2.4

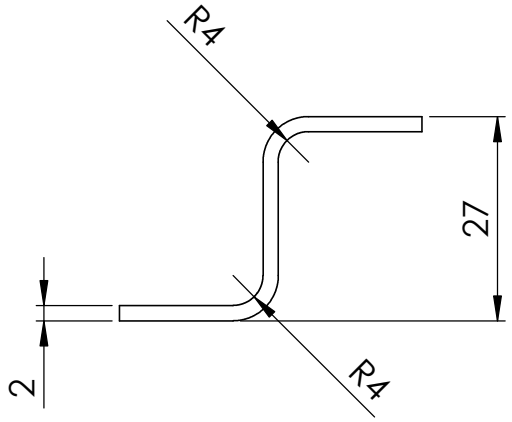


Vaade A-A
(1 : 2)

	Materjal	Märkimata piirhälbed	Mass	Mõõt
	ABS	ISO 2678-m Ra 6,3	0,47	1:2

Teostas	Kardo Aia	Nimetus	
Kinnitas	Maido Hiiemaa	Põhi	
Kontrollis	Maido Hiiemaa		
TALLINNA TEHNIKAÜLIKOOL Mehhatroonikainstituut		Leht 6	Tähis P1.2.5





	Materjal AISI 316	*Märkimata piirhälbed* ISO 2678-m Ra 6,3	*Mass* 0,07	*Mõõt* 1:1
Teostas	*Kardo Aia*	*Nimetus* Silindri jalg		
Kinnitas	*Maido Hiiemaa*			
Kontrollis	*Maido Hiiemaa*			
TALLINNA TEHNIKAÜLIKOOL *Mehhatroonikainstituut*		*Leht* 7	*Tähis* P1.2.6	