DIPLOMATERVEZÉSI FELADAT

SZATMÁRI BENDEGÚZ BENCE (R26O5T)

szigorló villamosmérnök hallgató részére

Félautomatikus tesztrendszer fejlesztése elektromos autó töltőoszlop töltésvezérlő egységéhez

Az evopro Innovation Kft. elektromos autók számára fejlesztett töltőoszlopaiban több egyedi fejlesztésű vezérlő modul található, amelyekhez automatikus tesztelési környezetet kívánunk biztosítani. A diplomaterv keretében az automatikus tesztelési környezetben a töltésvezérlő teszt alrendszerét kell kialakítani, amelynek alkalmasnak kell lennie, hogy a jelenlegi és a későbbi fejlesztési termékekhez is modulárisan biztosítson tesztelési környezetet. A töltésvezérlő több kommunikációs interfésszel rendelekezik (pl. GPIO, CAN), ezekhez szükséges a kártya és a teszt keretrendszer illesztő hardvereinek elkészítése, illetve a méréseket végző és kiértékelő szoftverszkriptek megvalósítása.

A hallgató feladatának a következőkre kell kiterjednie:

- Ismerje meg a töltőoszlop töltésvezérlő kártyájának felépítését és működését.
- Végezze el több töltésvezérlő kártya manuális bemérését és vizsgálatát.
- Ismerje meg az evopro Innovation Kft.-nél használt tesztelési környezetet és tegyen javaslatot az itt használható környzet továbbfejlesztésére és teszt metódusok kialakítására.
- Tervezze meg a kártya különböző interfészeihez kapcsolódó tesztrendszer modulokat.
- Implementálja a teszteket.
- Mérésekkel ellenőrizze és értékelje az eredményeket.
- A töltőoszlop teljes teszteléséhez tegyen javaslatot az emulációs környezet kialakítására.
- Tegyen javaslatot a további fejlesztési lehetőségekre.

Tanszéki konzulens: Dr. Orosz György

Külső konzulens: Dr. Szatmári István (evopro Innovation Kft.)

Budapest, 2016. 02. 10.

Dr. Dabóczi Tamás
egyetemi docens
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Szatmári Bendegúz Bence

FÉLAUTOMATIKUS TESZTRENDSZER FEJLESZTÉSE ELEKTROMOS AUTÓ TÖLTŐOSZLOP TÖLTÉSVEZÉRLŐ EGYSÉGÉHEZ

KONZULENSEK

Dr. Orosz György
Dr. Szatmári István (evopro Innovation Kft.)

BUDAPEST, 2016

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Szatmári Bendegúz Bence**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2016. 12. 18.

...…….
Szatmári Bendegúz Bence

Kivonat

A tesztelés fontos szerepet játszik egy termék fejlesztésében. Kihagyásával fennáll a veszélye a hibás működésnek, amely életvédelmi és pénzügyi kockázatot jelenthet. Továbbá a folyamatos tesztelés elősegíti a beüzemelési és javítási költségek csökkentését.

Egy komplex termék, mint egy elektromos autó töltőoszlop tesztelését több szinten kell megvalósítani. Legelőször modul szintű vizsgálatokat kell végezni. Majd az egyes elemekből felépülő alrendszerek integrációs tesztelését kell megtenni. Ezek után a rendszertesztek jelentik a következő lépést, majd végül a felhasználói tesztek következnek. Dolgozatom témája egy tesztrendszer fejlesztése volt, amely modul szintű tesztek elvégzésére képes.

Először bemutatom a rendelkezésemre álló tesztrendszert. Következő lépésként részletezem a kialakított tesztmetódusok felépítését. Ezután ismertetem a megvalósított áramkört, amely a töltésvezérlő kártya tesztelésére alkalmas. Részletesen leírom a tervezési megfontolásokat. Ezt követően bemutatom mind a processzor, mind a PC oldali tesztszoftver felépítését, működését, amelyekkel félautomata tesztek végrehajtása lehetséges. Diplomatervemet a kézi és félautomata tesztek összehasonlításával folytatom. Végül javaslatot teszek a töltőoszlop emulációs környezetének kialakítására. Ennek során bemutatom az autószimulációra alkalmas áramkör tervemet, amely az alrendszer integrációs tesztek elvégzését segíti.

Abstract

Testing has an important role in product development. With excluding it, there would be a chance for malfunction that could cause financial and life-threatening risks. Furthermore, continuous testing subserves the reduction of the setup and repair costs.

Testing of a complex product, like an electronic vehicle charger station must be done on different levels. First, the module level tests must be performed. Then the subsystem integration tests must be done. The system level tests are the next step after these, then the user level tests follow finally. My thesis' topic is a test system development that is capable of performing module level tests.

First, I introduce the existing test system. As a next step, I detail the structure of the developed test methods. Then I present the designed circuit, which is suitable for charge controller card tests. I describe the design considerations in detail. After this, I present both the processor and the PC-side test software's structure and operations, which makes it possible to execute semi-automatic tests. Then I continue my thesis with the comparison of manual and semi-automatic testing. Finally, I propose a solution for emulating an environment of the charger station. This includes I present my car simulator circuit design, which is suitable for carrying out subsystem integration tests.

1 Bevezetés

Manapság azok a cégek, amelyek lépést akarnak tartani a piaci versenyben, nem engedhetik meg maguknak, hogy hibás termékeket adjanak el. A kivitelező garanciát szokott vállalni a termék működésére vonatkozóan a megrendelő felé, így a nem megfelelően működő, hibás termékeket a kivitelezőnek javítania kell. Ez természetesen költségeket von magával a kivitelező számára. Egy hibás termék akár jelentheti azt is, hogy a jövőben más vállalatot fog megbízni a megrendelő a korábbi tapasztalatai alapján, tehát további bevételkiesést is okozhat. Hibás termékek készítését megfelelő tervezéssel és teszteléssel lehetséges elkerülni.

A megfelelő tervezés magába foglalja a teljes rendszerspecifikáció előállítását, amely alapján lefejleszthető a termék. A rendszerspecifikáció előállításában az ügyfélnek is részt kell vennie, hogy „közös nevezőre" juthasson mind a megrendelő, mind a kivitelező. Természetesen ez nem egy egyszerű, gyors folyamat, de ezt a továbbiakban nem részletezem.

A tesztelés elengedhetetlen része a termékfejlesztésnek, de a teszt vagy tesztrendszer kidolgozásában a megrendelő már nem vesz részt, mivel ő csak az elkészített terméket szeretné kézhez kapni. Magának a fejlesztőnek vagy kivitelezőnek kell a tesztrendszer kialakításával foglalkoznia (vagy esetleg külső cégtől is megrendelésre kerülhet). Hiába jár a tesztelés is költségekkel, a fejlesztőnek érdeke, hogy még időben, a termék átadása előtt kiderüljenek a hibái, mert így még mindig kisebb a javítás költsége, mintha vissza kellene hívni egy üzembe helyezett eszközt.

Tesztek implementálását, végzését érdemes már a fejlesztés korai szakaszában elkezdeni. A korán elkezdett teszteléssel nem a tesztelés költségének csökkenése jelenti az előnyt, hanem hogy korábban és kevesebb utánajárással (mondhatni mérnökórával) fény derülhet a rendszer hibáira [1], így azok még egyszerűbben és gyorsabban javíthatók, mint egy már bonyolult, összeállított rendszer esetén.

A nem megfelelő vagy elmaradt tesztek számos balesetet, katasztrófát okoztak már a történelem során. Ezek egy része elkerülhető lett volna teszteléssel, de például idő hiányában vagy pénzügyi okok miatt azok elmaradtak. Meg kell jegyezni, hogy egy

komplex rendszerben teszteléssel sosem lehet lefedni a rendszer viselkedését az összes bemeneti kombinációra. Azt lehetetlen biztosítani, hogy egy rendszerben nincs hiba, csak csökkenteni lehet a tesztelés során nem észlelt hibák számát. Ebből kifolyólag kimerítő tesztelés helyett a felmerülő kockázatok és a prioritások alapján lehet fókuszálni a tesztelési erőforrásokat [2].

A 2. fejezetben bemutatom az evopro Kft. által fejlesztett elektromos autó töltőoszlopot és a töltésvezérlő áramkört.

A 3. fejezetben ismertetem a rendelkezésemre álló teszthardver és tesztszoftver modulokat, valamint leírom a tesztrendszer továbbfejlesztési javaslataimat.

A 4. fejezetben bemutatom az összeállított tesztrendszert és a kidolgozott tesztmetódusokat.

Az 5. és 6. fejezetben részletesen ismertetem a töltésvezérlő tesztelésére alkalmas teszthardvert és tesztszoftvert.

A 7. fejezetben összehasonlítom a kézi és a félautomatikus méréseket.

A 8. fejezetben ismertetem az emulációs környezet kialakítására tett javaslatom, valamint bemutatom az autószimulálásra alkalmas áramköri tervemet.

A 9. fejezetben összefoglalom az általam végzett munkát, valamint szót ejtek a továbbfejleszthetőségről.

A következő fejezetben (1.1. fejezet) a használt fogalmakat tisztázom, majd a tesztelési folyamat általános felépítését mutatom be az 1.2-es pontban.

1.1 Alapfogalmak

Teszt (Test): egy rendszerkomponens, amely lehet például egy áramköri elem, egy teljes áramkör, egy szoftvermodul stb., üzembe helyezés előtti vizsgálata. A tesztelés típusa befolyásolja az elvégzését, környezetét és menetét.

Tesztfeltétel (Test condition): a vizsgálat szemszöge, amely alapján a teszt végrehajtása történik, például egy eszköz áramfelvétele, életvédelmi funkciói, felhasználói ergonómiája stb.

Elfogadási kritérium (Acceptance criteria): az az értéktartomány, amelybe ha beleesik a tesztelés eredménye, akkor a teszt sikeres volt, minden más esetben a teszten megbukott az eszköz.

Kilépési kritérium (Exit criteria): nem lehet teljesen kimerítő tesztet végezni, ezért szükséges meghatározni egy olyan értéket, határt, amelynek elérésével a tesztelést befejezettnek tekintjük [2].

Teszteset (Test case): a tesztelt eszköz egy adott szempont szerinti vizsgálata. A szempontot a tesztfeltétel, a sikerességét pedig az elfogadási kritérium határozza meg. Minden esetben egy kimenete van (sikeres vagy sikertelen), de ezen kívül más kiegészítő információkat is tartalmazhat, például a teszt során mért értékeket.

Tesztkészlet (Test suite): tesztesetek gyűjteménye. Meghatározhat teszteset sorozatokat, szekvenciákat, előfeltételeket az egyes tesztek végrehajtásához. A tesztesetnél nagyobb információtartalommal bír, megadhatja, hogy mi indokolja a tesztesetek sorrendjét. Például egy kommunikációs protokoll tesztelése esetén leírja az üzenetek küldési sorrendjét, valamint, hogy milyen válaszokat vár az egyes üzenetekre.

Tesztelési terv (Test plan): leírja, hogy milyen tesztek elvégzése szükséges, mi a fókusza az elvégzendő vizsgálatoknak. Meghatározza a tesztelési feltételek alapján, hogy milyen tesztkészletre van szükség, esetleg prioritási sorrendet ad meg közöttük, de nem ad teszteset szintű tájékoztatást.

Tesztelési forgatókönyv (Test scenario): leírja, hogyan kellene viselkednie a tesztelt eszköznek a tesztelés alatt. Például ha a felhasználó hibásan adja meg a bejelentkezési adatait, akkor ne engedje tovább a rendszer, amíg helyesen meg nem teszi.

DUT (Device under test): tesztelt eszköz.

1.1.1 Tesztelés besorolása automatizáltság alapján

Operátor (Operator): tesztelést végző vagy felügyelő személy. Az operátori beavatkozás szerint megkülönböztetünk kézi, félautomatikus és automata teszteket.

Kézi tesztelés (Manual testing): az operátor minden egyes tesztelési lépést valamilyen beavatkozással indít el, valamint az eredmények mentését, naplózását, esetleg a kiértékelését is ő végzi. Az operátornak át kell látnia, értenie kell az eszköz működését. Például egy áramkörben meg akarja vizsgálni, hogy a mennyire zajos egy tápegység IC kimenete. Ehhez oszcilloszkópot használ, az eszköz egyik csatornáját AC csatolt állapotba kapcsolja, beállítja a megfelelő feszültség- és időfelbontást, valamint a mérendő tulajdonságokat, majd a mérőpontra helyezi az oszcilloszkóp mérőfejét, készít egy képernyőképet a mérésről, amelyet elment egy adathordozóra. Ahogy ebből a leírásból is

érzékelhető, a folyamat időigényes, ezért nagy szériaszámú tesztek esetén érdemes automatizálni a műveletet. Az automatizálás megvalósítása előtt elengedhetetlen egy-egy ehhez hasonló kézi mérés elvégzése, mert szükség van az egyes beállítások, várható értékek vagy értéktartományok meghatározására, hogy teszteseteket, elfogadási kritériumokat lehessen készíteni azok felhasználásával.

Félautomatikus tesztelés (Semi-automatic testing): az operátor feladatainak száma csökkentésre kerül a kézi teszteléshez képest, valamint a teendőinek elvégzéséhez kevesebb tudással kell rendelkeznie a tesztelendő eszközről. Félautomata tesztrendszer esetén, az előző példából kiindulva, az operátornak csak a mérőfej tesztelési pontra helyezését kell elvégeznie, valamint jelezni a tesztrendszer felé, hogy ezt megtette, folytatódhat a mérés. A vizsgálat kiértékelése szoftveresen történik a kézi tesztelés alapján meghatározott értéktartományok segítségével, valamint a teszt menedzsment szoftver képes naplózni a mért értékeket. Emberi (operátori vagy szakértői) beavatkozásra csak hibás vagy teszten megbukott termék esetén van szükség.

Automatikus tesztelés (Automatic testing): az operátor feladata annyiban merül ki, hogy behelyezi a tesztrendszerbe a vizsgálandó eszközt, majd futtatja a mérést, amely lehet egy egyszerű teszteset, de egy komplex egész rendszerre kiterjedő tesztszekvencia is. A kiértékelés automatikusan történik meg, szakértői beavatkozásra mindössze akkor van szükség, ha megbukott az eszköz a folyamaton.

1.1.2 Tesztelés besorolása célterület alapján

Hardverteszt (Hardware test): a vizsgált eszköz hardveres tulajdonságainak vizsgálata. Például egy áramkör áramfelvétele, egy jel felfutási ideje vagy egy digitális kimenet feszültségszintje stb.

Funkcionális teszt (Functional test): az eszköz egy adott funkciójának vizsgálata. Például egy kiadott jel logikai 0 vagy 1 értékének meghatározása, egy AD átalakító helyes értéket rendel a mért feszültséghez vagy egy motor helyesen reagál a vezérlőjelének megváltozására.

Modul teszt (Module test): A rendszer (pl.: töltőoszlop) egy önállóan kezelhető egységének (pl.: töltésvezérlő áramkör) vizsgálata. Magába foglalja a hardveres és funkcionális teszteket. Célja a tesztelt egység önálló működésének validációja.

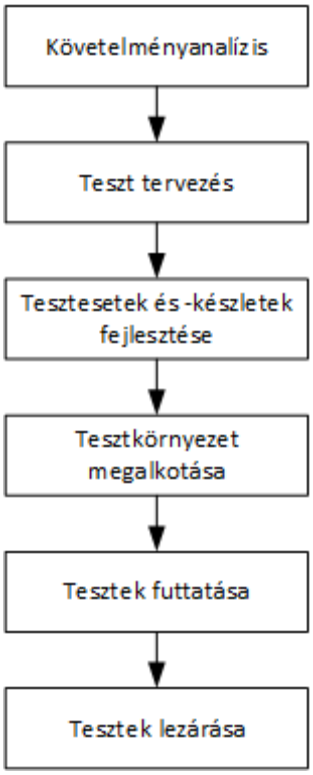
1.2 Tesztelési folyamat

A tesztelési folyamat lépéseit az 1. ábra mutatja. Az első lépés a követelményanalízis, amelyben a tesztrendszer fejlesztője a tesztspecifikáció alapján meghatározza az erőforrás-szükségleteket, eszközkövetelményeket. A tesztrendszer kialakítását, fejlesztését általában nem a tesztelt eszköz fejlesztői szokták végezni. Ennek oka, hogy az ember sokkal nehezebben találja meg a hibát saját rendszerében (például egy rosszul értelmezett követelmény miatt), vagy nem biztos, hogy minden eset tesztelésére gondolt. A követelményanalízis során össze kell gyűjteni a tesztelési feltételeket, amelyekre alapozva már lehet teszteseteket vagy tesztelési tervet kidolgozni. Ezek előtt még fontos az automatizáltság mértékét meghatározni. Ennek a tulajdonságnak az eldöntését pénzügyi és technikai szempontok befolyásolják, például:

- mennyire bonyolult teszteket kell végezni,
- mekkora költséggel járna, mennyi időbe kerülne a fejlesztés,
- milyen darabszámban kerülne gyártásra a termék,
- megtérülne-e a tesztrendszer kifejlesztés stb.

Ezeknek a szempontoknak a mérlegelésével már meghatározható, hogy kézi, félautomatikus, vagy automatikus tesztelés a kézenfekvő.

A tesztelési tervek megalkotásakor mind a lentről felfelé, mind a fentről lefelé történő „építkezés” megvalósítható. Az első esetben az 1. ábra *Teszt tervezés* és *Teszteset fejlesztés* blokkja helyet cserél, mivel a tesztelési terv egy magasszintű leírása az elvégzendő teszteknek. A Top-Down elv annyiban lehet jobb, mint a Bottom-Up, hogy az egyszerűen megfogalmazható követelmények felől közelít. Például: „CAN buszon képes legyen kommunikálni a két fél”. Ebből könnyen lehet tesztszekvenciákat alkotni (pl.: „a küldő fél üzenetét fogadta és értelmezte a fogadó”), majd azokat felbontani tesztesetekre (pl.: „a küldő sikeresen kiadta az üzenetet”). A teszt tervezés magába foglalja a kilépési kritérium meghatározását, míg az elfogadási kritériumok a teszteset fejlesztés során adhatók meg, például kézi teszteléssel mért értékek alapján.



1. ábra: Tesztelési folyamat Top-Down elv esetén

Az első három lépésnek (követelményanalízis, teszttervezés, teszteset fejlesztés) az elvégzésével a tesztspecifikáció közel teljes, végleges állapotba kerül (a később felmerült igények okozhatnak még bővülést, módosulást), így már megalkotható a tesztkörnyezet. Manapság számos eszköz, mind hardver, mind szoftver, áll rendelkezésre tesztek végrehajtásához. Tesztvezérlő vagy tesztmenedzsment szoftverek közül ebben a dolgozatban a National Instruments által fejlesztett TestStand-et [5] fogom részletesebben bemutatni (3.2.1. fejezet).

2 Töltőoszlop bemutatása

Ebben a fejezetben bemutatom a töltőoszlopot, valamint a töltésvezérlő kártyát.

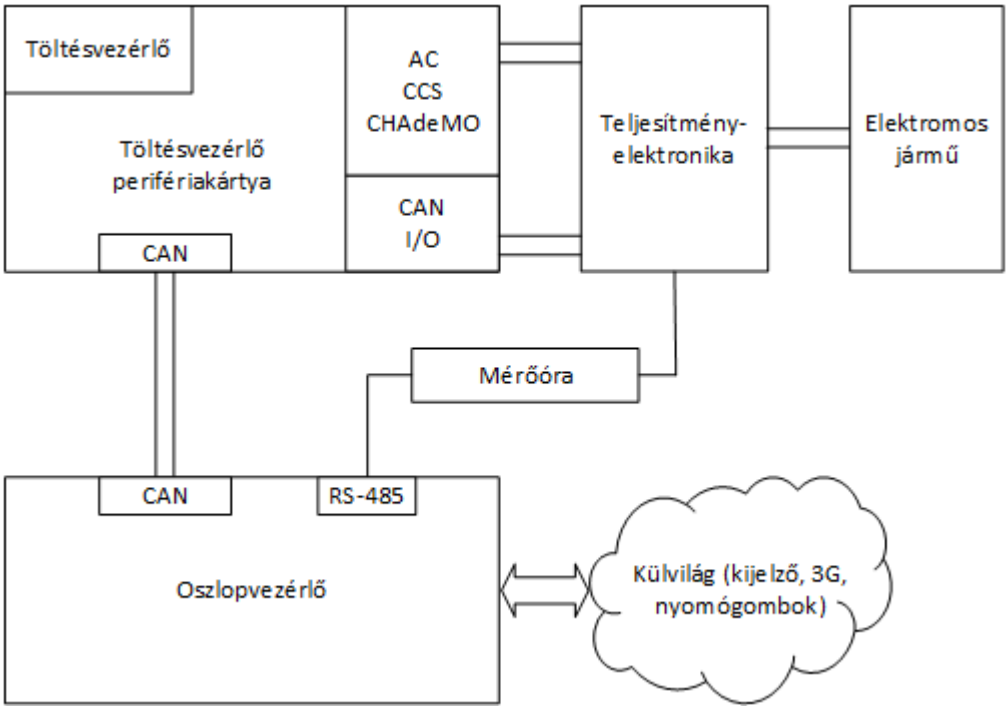
2. ábra: Töltőoszlop [6]

A 2. ábra mutatja a töltőoszlopot. A vezérlőelektronikát az evopro Innovation Kft., a dizájntervet a Maform Kft. [7], a teljesítményelektronikai komponenseket a PROCON Hajtástechnika Kft. [8] készítette. A töltőoszlop egyidejűleg biztosítani képes AC (43kW töltési teljesítmény) és DC (50kW töltési teljesítmény) töltést két jármű részére. CCS (Combined Charging System), CHAdeMO és AC töltési protokollokat támogat. A felhasználónak lehetősége van mobilapplikáción vagy webes felületen keresztül időpontfoglalásra, státusz lekérdezésre. A töltőoszlop 3G-n vagy LAN-on keresztül képes internetes kapcsolatot létesíteni. A felhasználó felé LCD kijelzőn szolgáltat információkat az oszlop, aki RFID kártyás azonosítás után nyomógombok segítségével tudja kezelni a menüt. A valós fogyasztás mérését a beépített mérőóra valósítja meg.

A töltőoszlop blokkvázlatát a 3. ábra mutatja. Az oszlopvezérlő elektronika tartja a kapcsolatot a külvilággal (kezeli a 3G modemet, meghajtja a kijelzőt stb.). A vezérlést az oszlopvezérlő áramkörön található Toradex SBC (Single Board Computer) [9] valósítja meg, amelyen beágyazott Linux operációs rendszer fut. Az oszlopvezérlő áramkör az evopro Innovation Kft. terméke.

A töltési protokollokat, tehát a töltésvezérlést egy Texas Instruments DSP implementálja. Ezeket a protokollokat a 8.2-es fejezetben mutatom be. A töltésvezérlő áramkör egy perifériakártyához csatlakozik, amely szintillesztésért, izolációért felel. Mindkét áramkör az evopro Innovation Kft. terméke.

Az oszlop- és a töltésvezérlő CAN buszon kommunikálnak egymással. A töltésvezérlő processzor szintén CAN-en kommunikál a teljesítményelektronikai modulokkal és az elektromos járművel (CCS és CHAdeMO protokoll esetén). A mérőóra RS-485-ön kapcsolódik az oszlopvezérlőhöz.

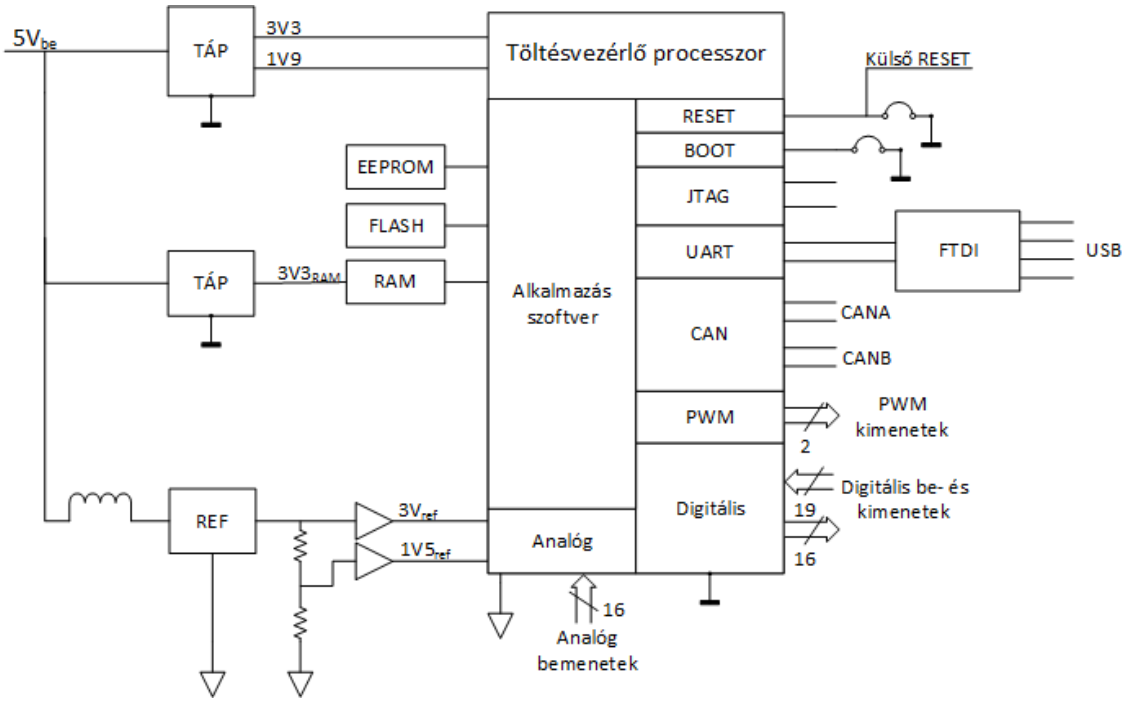


3. ábra: Töltőoszlop blokkvázlat

2.1 Töltésvezérlő kártya

Feladatom egy tesztrendszer kidolgozása volt a töltőoszlop töltésvezérlő áramköréhez, ezért a továbbiak megértésében a töltésvezérlő kártya rövid bemutatása elengedhetetlen.

A töltésvezérlő áramkör blokkvázlatát a 4. ábra mutatja. 5V tápfeszültséget vár el a külvilágtól, amelyből előállítja a processzor és a külső RAM működéséhez szükséges feszültségértékeket (3,3V, 1,9V illetve 3,3V), valamint az analóg mérésekhez a referenciafeszültségeket (3V és 1,5V).



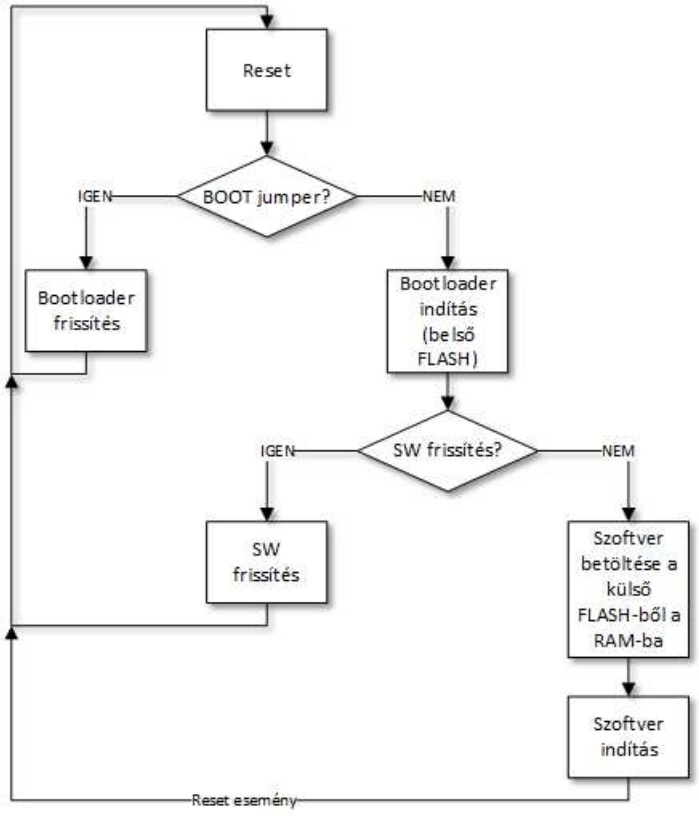
4. ábra: Töltésvezérlő áramkör felépítése

A kártyán egy Texas Instruments DSP található. Be- és kimeneteivel (16 analóg, 19 digitális be-, 16 digitális és 2 PWM kimenet) a töltésvezérlést és monitorozást valósítja meg. 2 db független CAN interfészen kommunikál az oszlopvezérlővel, a teljesítményelektronikával és az elektromos járművel. A DSP UART soros kommunikációval csatlakozik egy FTDI chipre, amely segítségével PC-ről is elérhető soros eszközként. JTAG interfésze debug célokra használható. A processzor BOOT bemenete jumper-rel, a RESET vonala jumper-rel és külső digitális vezérléssel kezelhető.

A processzor két belső és három külső memóriát alkalmaz. A belső FLASH tárolja a bootloader-t, a külső FLASH az alkalmazáskódot, a külső EEPROM a töltőszoftverben

alkalmazott változók alapértékeit. A belső RAM-ban az alkalmazás, a külső RAM-ban pedig a soros vonalon történő debuggolást segítő szoftverkomponens fut.

Az eszközön futó szoftver működését a 5. ábra mutatja. Reset esemény után a BOOT jumper állapotától függően lehetőség nyílik a bootloader szoftver frissítésére vagy annak futtatására. Ha a bootloader futása alatt nem érkezik kérés az alkalmazás szoftver frissítésére, akkor betölti az alkalmazást a RAM memóriába. Az alkalmazás futását csak RESET esemény (akár szoftveres, akár hardveres) szakíthatja meg, amely hatására a ciklus újraindul. A bootloader és az alkalmazás is letölthető a töltésvezérlőre soros porton keresztül a megfelelő PC oldali szoftver használatával.



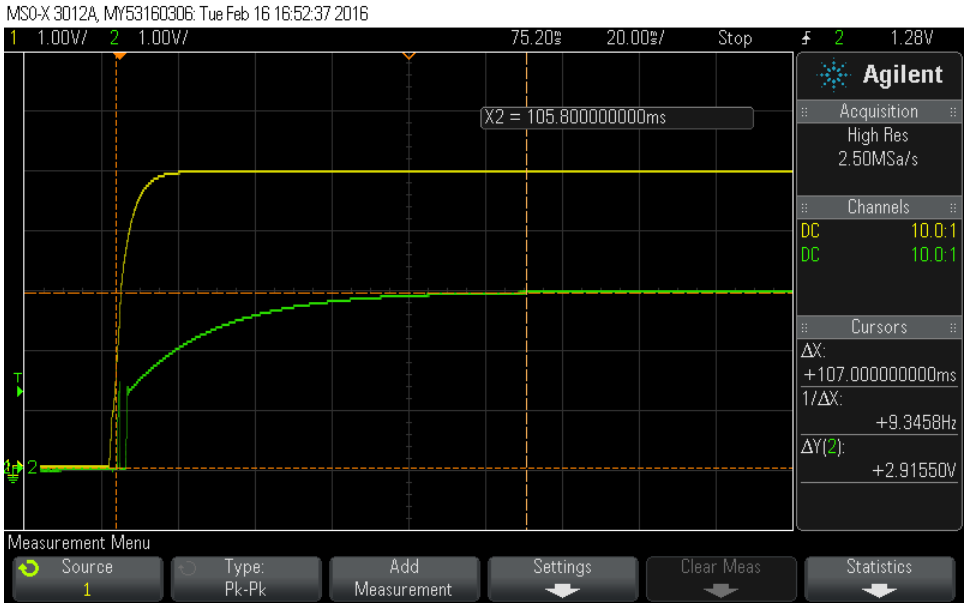
5. ábra: Töltésvezérlő processzoron futó szoftver működése

2.1.1 Manuális bemérések

Több töltésvezérlő kártya manuális bemérését is elvégeztem. A mérések célja a követelményanalízis felállítása volt, amely alapján a tesztspecifikációt el tudtam készíteni. A mérési eredményeket egy táblázatba naplóztam. A vizsgált komponensek a következők voltak:

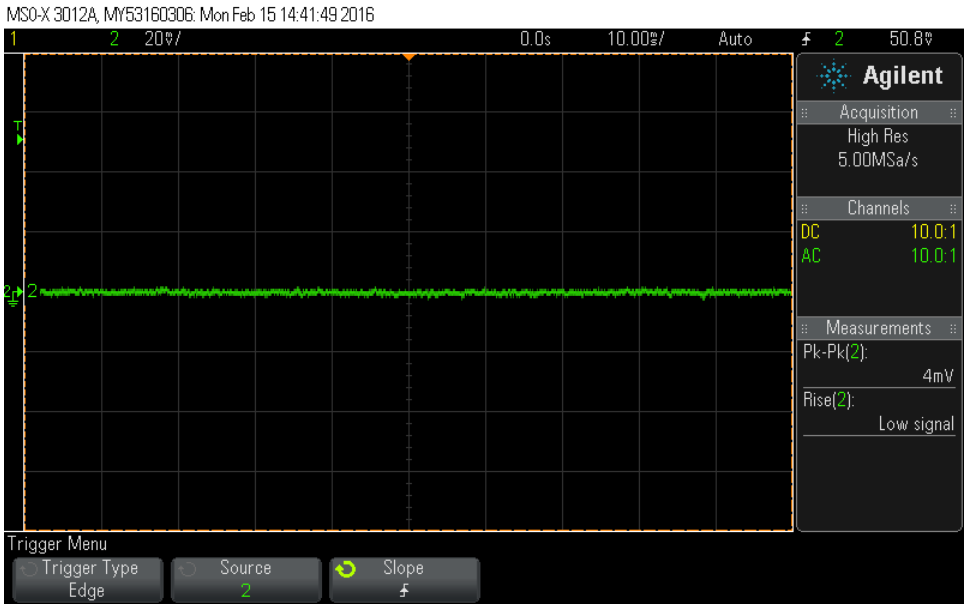
1. Táp- és referenciafeszültségek (érték, felfutási idő, zajosság/hullámosság),
2. Eszköz áramfelvétele,
3. Analóg bemenetek (rövidzár a vonalak között, feszültségérték helyes kiolvasása),
4. Digitális be- és kimenetek (rövidzár a vonalak között, logikai érték helyes kiolvasása),
5. PWM kimenetek (rövidzár a vonalak között, pontos kitöltési tényező),
6. CAN interfészek (rövidzár a vonalak között, üzenetküldés és -fogadás).

Az első két pont elemeinek mérését oszcilloszkóppal és digitális multiméterrel végeztem, a másik négy típusú méréshez az evopro Innovation Kft. által fejlesztett, soros porton kommunikáló, PC oldali vezérlőalkalmazást használtam (Wterm). Az oszcilloszkópos mérésekről képernyőképet készítettem, az alábbiak a 3V-os referencia bemérésekor készült ábrák:



6. ábra: 3V-os referencia felfutásidő mérés

Az ábrák egy jól működő, referenciának tekintett kártya mérése során készültek. A vizsgált jel az ábrákon zöld színnel látható. A kurzorok segítségével 107ms-os felfutási időt mértem (6. ábra), zajosságra a $4mV_{pk\text{-}pk}$ értéket pedig az oszcilloszkóp határozta meg (7. ábra). A továbbiakban ezeket az értékeket referenciának tekintettem.



7. ábra: 3V referencia zajosság/hullámosság

A manuális bemérések során felprogramoztam az FTDI chip-et (FT Prog [10]), a belső és külső FLASH (bootloader és alkalmazás szoftver) memóriát (C2Prog [11] illetve Wterm), ezután pedig a funkcionális teszteket végeztem el.

A funkcionális tesztek elvégzéséhez a Wterm alkalmazás adott visszajelzést, hogy melyik bemeneti vonalon milyen értékek találhatók, valamint lehetőségem volt a kimenetek manuális állítására (8. ábra).

OUT			IN0			IN1			ADC[0:7]			ADC[8:15]			PWM		
O_A0	( )	W	I_A0	( )	R	I_A16	(*)	R	ADC ch[0]	0.015d	R	ADC ch[8]	0.000d	R	PWM ch[0]	50d	W
O_A1	( )	W	I_A1	(*)	R	I_A17	(*)	R	ADC ch[1]	0.001d	R	ADC ch[9]	0.007d	R	PWM ch[1]	50d	W
O_A2	( )	W	I_A2	(*)	R	I_A18	(*)	R	ADC ch[2]	0.013d	R	ADC ch[10]	0.008d	R			
O_A3	(*)	W	I_A3	(*)	R				ADC ch[3]	0.009d	R	ADC ch[11]	0.002d	R			
O_A4	( )	W	I_A4	(*)	R				ADC ch[4]	0.013d	R	ADC ch[12]	0.008d	R			
O_A5	( )	W	I_A5	(*)	R				ADC ch[5]	0.010d	R	ADC ch[13]	0.009d	R			
O_A6	( )	W	I_A6	(*)	R				ADC ch[6]	0.016d	R	ADC ch[14]	0.010d	R			
O_A7	( )	W	I_A7	(*)	R				ADC ch[7]	1.844d	R	ADC ch[15]	0.010d	R			
O_A8	( )	W	I_A8	( )	R												
O_A9	( )	W	I_A9	( )	R												
O_A10	( )	W	I_A10	( )	R												
O_A11	( )	W	I_A11	( )	R												
O_A12	( )	W	I_A12	( )	R												
O_A13	( )	W	I_A13	( )	R												
O_A14	( )	W	I_A14	( )	R												
O_A15	( )	W	I_A15	( )	R												

8. ábra: Be- és kimenetek kezelése Wterm segítségével

Az ábrán látható, hogy a be- és kimeneti vonalak külön-külön egységbe vannak szervezve, mind típus, mind irány szerint. AD csatorna esetén a mért feszültségértéket jeleníti meg. PWM kimenet esetén a kitöltési tényező állítható. Digitális be- és kimeneteknél az üres zárójelek a logikai alacsony, míg a csillaggal kitöltött zárójelek a logikai magas szintet jelölik. A W jelzés az írható (vezérelhetőség a soros alkalmazáson keresztül), az R pedig a csak olvasható értékek mellett található.

3 Tesztelési környezet

Egy tesztrendszer kialakításához a következő lehetőségek állnak rendelkezésre:

- teljesen egyéni, a tesztelt termékre szabott hardver és szoftver rendszer kialakítása,
- megvásárolható hardvermodulok és szoftveres keretrendszer használata, amelyekhez kiegészítő hardver tervezése és a tesztszoftver implementálása szükséges, mert így lesz a tesztelt termékre szabva.

Az első megoldás hátránya a sok hibalehetőség, mivel egy teljesen új rendszer kidolgozására van szükség. Ha egy összetett, részben újrahasznosítható tesztrendszer megalkotása a cél, akkor szükséges egy tesztelési keretrendszer kialakítása is. A megfelelően skálázható tesztelési protokoll megvalósítása időigényes és összetett feladat, továbbá a validációt ebben az esetben sem lehet elhagyni. A teljesen egyéni megoldás előnye viszont, hogy az igényekhez igazítva megválogatható mennyi és milyen interfésszel, kivezetéssel rendelkezik.

A második opció megvásárolható moduljai elemi szinten validálva beszerezhetők, nem szükséges például az áramkörök élesztésével időt tölteni. Természetesen a rendszer szintű validáció nem kerülhető el. A kiegészítő hardver- és a szoftvermodulok a tesztrendszer fejlesztésének része, ezért ezeket ugyanúgy ellenőrizni kell, mintha a tesztelt eszközök lennének. Viszont ennél az opciónál rendelkezésre áll egy, az iparban is használt, standard architektúra, így a tesztrendszer validációja kisebb ráfordítást igényel.

Minden projektnél érdemes mérlegelni, hogy melyik megoldás a megfelelő, ehhez számos szempontot kell figyelembe venni, például: idő- vagy pénzügyi keret, humánerőforrás stb. Az evopro Innovation Kft. National Instruments és saját fejlesztésű, tesztelést támogató, hardver- és szoftvereszközökkel rendelkezik, ezek megismerése elengedhetetlen volt számomra a töltésvezérlő teszthardverének és tesztszoftverének fejlesztésekor.

3.1 Rendelkezésre álló hardver modulok

A National Instruments [12] alapvetően mérési, tesztelési és vezérlési eszközök fejlesztésével foglalkozik. Egy-egy hardver megvásárlása mellé firmware-t és API-t biztosítanak, a fejlesztésekhez pedig a LabVIEW [13] programozási környezet használatát ajánlják, de elérhető C vagy C# nyelvű API is. A LabVIEW egy grafikus programozási felület és adatfolyam alapú nyelv egyben, amely átláthatóságával, könnyen vizualizálható működésével a fejlesztési folyamatok felgyorsítását teszi lehetővé. A környezet lehetőséget biztosít más nyelven írt kódok importálására a LabVIEW programba vagy például DLL hívások végrehajtására is. Szinte bármilyen személyi számítógépen futtatható, Windows, Linux és MAC verzió is elérhető.

A tesztrendszer a következő National Instruments eszközöket foglalja magába:

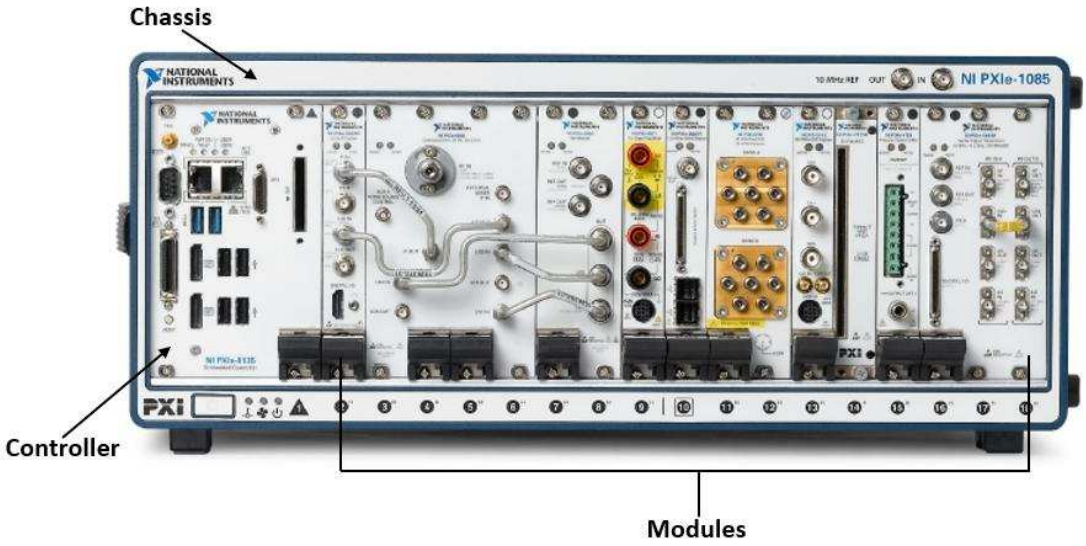
- a PXI ipari PC-t (3.1.1 fejezet),
- az NI multifunkciós adatgyűjtőkártyát (3.1.2 fejezet),
- és az NI VirtualBench integrált mérőműszert (3.1.3 fejezet).

A tesztrendszer nem NI fejlesztésű eszközei:

- a SYSTEC CAN interfész (3.1.4 fejezet),
- a TENMA programozható tápegységek (3.1.5 fejezet),
- valamint az evopro Innovation Kft. által fejlesztett tesztműszer interfészáramkör (3.1.6 fejezet).

3.1.1 PXI ipari számítógép

A PXI busz (9. ábra) a PCI, illetve a PCI Express busz, szinkronizációs és trigger vonalakkal kiterjesztett változata [14]. A PXI kommunikációs-, mérő-, és vezérlőmodulok (a felsorolás közel sem teljes) csatlakoztatására ad lehetőséget. Tetszőlegesen feladatra szabható, hogy milyen modulok kerülnek beépítésre. Az általam használt ipari PC egy 18 PXI modult (Modules) csatlakoztatni képes keretből (Chassis, PXIe-1075 [15]) és egy nagy számítási teljesítményű Intel Core i7-es processzor alapú PXI Express vezérlőből (Controller) [16] áll. A PC-n Windows operációs rendszer alatt fut a LabVIEW programozási környezet.



9. ábra: PXI ipari PC [17]

3.1.2 Multifunkciós adatgyűjtőkártya

Az ipari PC-be csatlakoztatva található a rendszerben az úgynevezett DAQ (Data Acquisition) mérőmodul (10. ábra) [18]. Az eszköz 16 ADC csatornával rendelkezik (differenciális mérés esetén 8 pár), amelyek 16 bites felbontással képesek mérni +/-0,1V-tól +/-10V-ig terjedő tartományon. Mintavételi sebessége 1MS/s a csatornáira összesen, így ezzel a kártyával akár felfutásokat is lehet mérni, ha csak egy van használatban. Az eszközben megtalálható 24 digitális I/O vonal (ebből 2 időzítő- vagy számlálóként is használható), összesen 40mA árammal terhelhetők, ezért csak logikai szintek kiadására alkalmasak, áramköri elemek meghajtásra nem. A kártya 0-5V-os logikai jelszinteket használ.

10. ábra: PXI-6250 adatgyűjtőkártya [18]

Hátránya, hogy ez a típus nem rendelkezik analóg kimenettel, ezért analóg jelek kiadását más módon, például programozható tápegység kimenetének használatával kell megoldani.

3.1.3 VirtualBench integrált mérőműszer

A VirtualBench (11. ábra) [19] ötféle műszert „tömörít” egyben: logikai analizátor, oszcilloszkóp, multiméter, tápegység, függvénygenerátor, valamint 8 digitális I/O is megtalálható benne.

11. ábra: VirtualBench [19]

A digitális vonalak a DAQ mérőmodulhoz hasonlóan logikai jelszintek kiadására alkalmasak (0-3,3V-os jelszint, áramterhelhetőség 4mA), de SPI és I2C master-ként is használhatók.

Két tápegysége van, maximális kiadható feszültségük +6V és +/-25V, melyek terhelhetősége rendre 1A illetve 0,5A. Az ezekkel kiadott maximális feszültségszint és áramkorlát programozottan állítható 0,1%-os pontossággal. Kis áramfelvételű eszközök meghajtásához elegendők lehetnek a tápegységek, de a töltőoszlop áramköreinek felfutás-vizsgálatához kevés áramot tudnak leadni, ezért külső, nagyobb terhelhetőségű, programozható tápegységekre van szükség a tesztekhez.

A VirtualBench két oszcilloszkóp csatornával rendelkezik, amelyek méréshatára $0{,}1V_{pk\text{-}pk}$ feszültségtől $40V_{pk\text{-}pk}$ feszültségig állítható. A csatornák 8 bites AD átalakítóval rendelkeznek, ezért érdemes a vertikális irányú felbontást mindig a lehető legkisebbre beállítani, hogy a mért jel még beleférjen. Maximális mintavételi sebessége 1GS/s egy csatornára, mindkettő alkalmazása esetén 500MS/s. Mindkét csatornához tartozik egy-egy belső tároló, amelyben 1MS/csatorna adatot tud tárolni. Ez azt jelenti, hogy

kétcsatornás üzemmódban 500 MS/s sebességen maximum 2ms ideig képes mintavételezni, ezután a 2ms-nál korábbi minták felülíródnak. Természetesen a mintavételi sebesség csökkentésével a mérés időtartama megnövelhető.

A műszerhez a National Instruments C nyelvű és LabVIEW-s API-t is biztosít. A VirtualBench USB porton keresztül kezelhető, így csatlakoztatható a PXI ipari számítógéphez és beilleszthető a tesztrendszerbe.

3.1.4 CAN modul

12. ábra: SYSTEC USB-CAN modul [21]

A tesztrendszer része egy, a SYSTEC GmbH [20] által gyártott, USB-CAN átalakító (USB-CANmodul2 [21], 12. ábra), amely két független CAN interfésszel rendelkezik. Támogatja a kis- és nagysebességű CAN kommunikációt (10kbps-1Mbps), valamint a standard és bővített üzeneteket is. Kezeléséhez Windows operációs rendszereken használható Microsoft VisualC++, LabVIEW, .NET API és Linux-on SocketCAN alapú API.

3.1.5 Programozható tápegységek

A tesztelt eszközök tápellátását két egycsatornás TENMA 72-2540 programozható tápegység biztosítja (13. ábra). A tápegység a manuális előlapi kezelőfelületén kívül RS-232 és USB (soros eszközként ismeri fel a PC) interfészen keresztül kezelhető, sztring parancsokkal. 0-30V feszültség és 0-5A áram kiadására képes, amelyeket programozottan 10mV-os és 1mA-es pontossággal képes visszaolvasni. Öt konfigurációt képes elmenteni a memóriájába (kikapcsolás után is tárolja a

beállításokat), ezek közül négy az előlapi gombokkal is, míg az ötödik csak programozottan kezelhető.

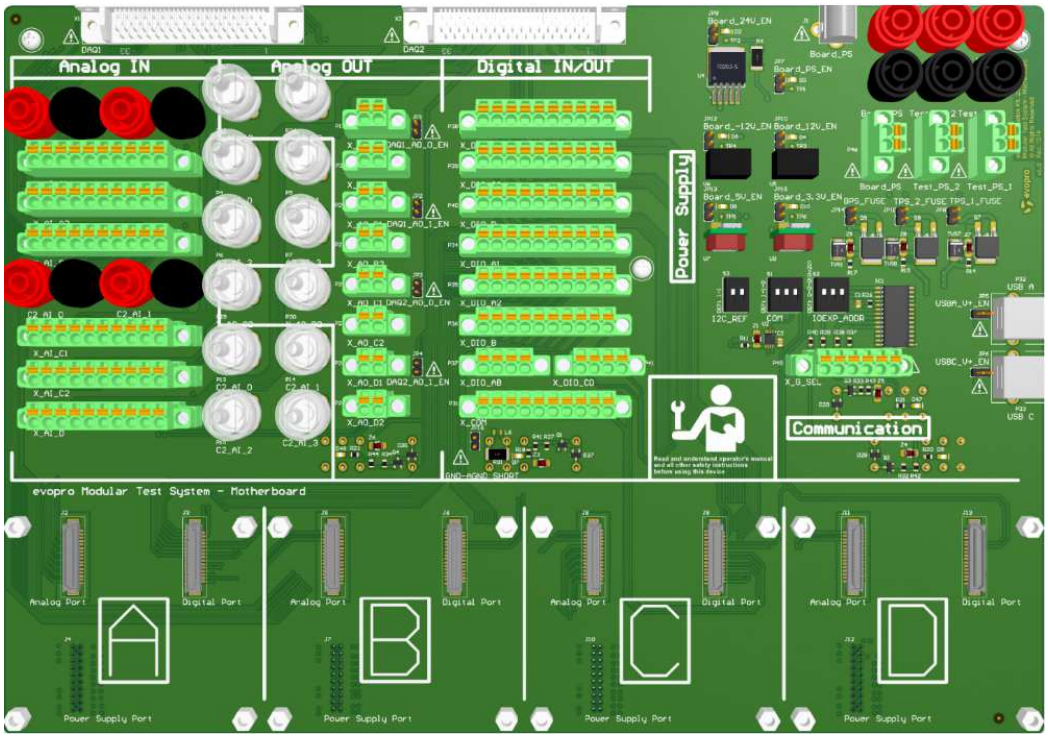
13. ábra: TENMA 72-2540 programozható tápegység [22]

3.1.6 Tesztműszer interfészáramkör

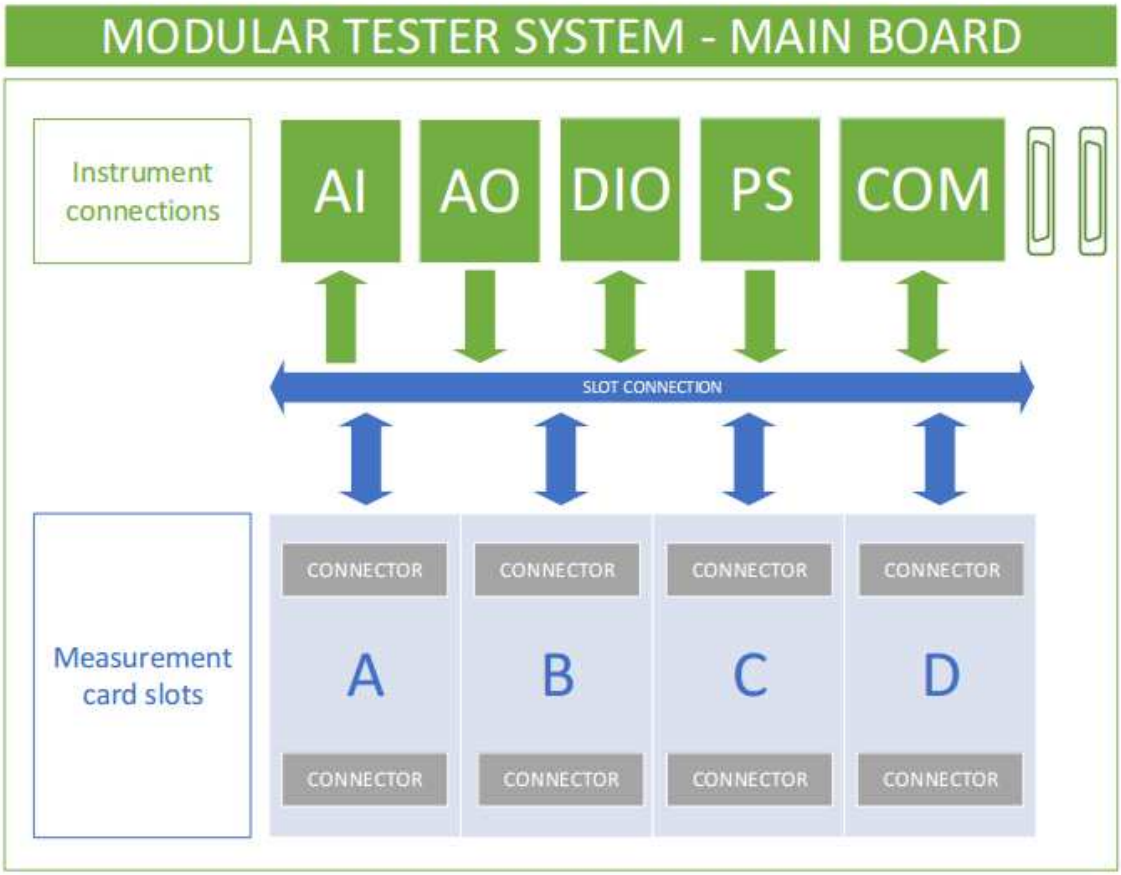
A tesztműszer interfészáramkör (14. ábra) funkciója a mérőműszerek jeleinek (analóg, digitális, kommunikációs) és a tápegységek vonalainak csoportosítása, rendszerezése és szétosztása a feltétkártyák között.

A rendszer blokkvázlatát a 15. ábra mutatja (jelölések: analóg bemenetek (AI) és kimenetek (AO), digitális be- és kimenetek (DIO), tápegység vonalak (PS) és kommunikációs jelek (COM)). Az ábra jobb oldalán látható két csatlakozó a DAQ kártyák csatlakoztatási pontját szimbolizálják.

Ez az „alaplap” maximum négy mérés- vagy eszköz specifikus feltétkártya csatlakoztatására ad lehetőséget, amelyek a tápvonalakat tüskesoron, a további jeleket pedig board-to-board csatlakozókon érhetik el. Az áramkör az evopro Innovation Kft. terméke, ismertetése azért szükséges, mert ehhez terveztem kiegészítőkártyát, amely a töltésvezérlő áramkör tesztelésére alkalmas. Ennek bemutatását az 5. fejezetben teszem meg.

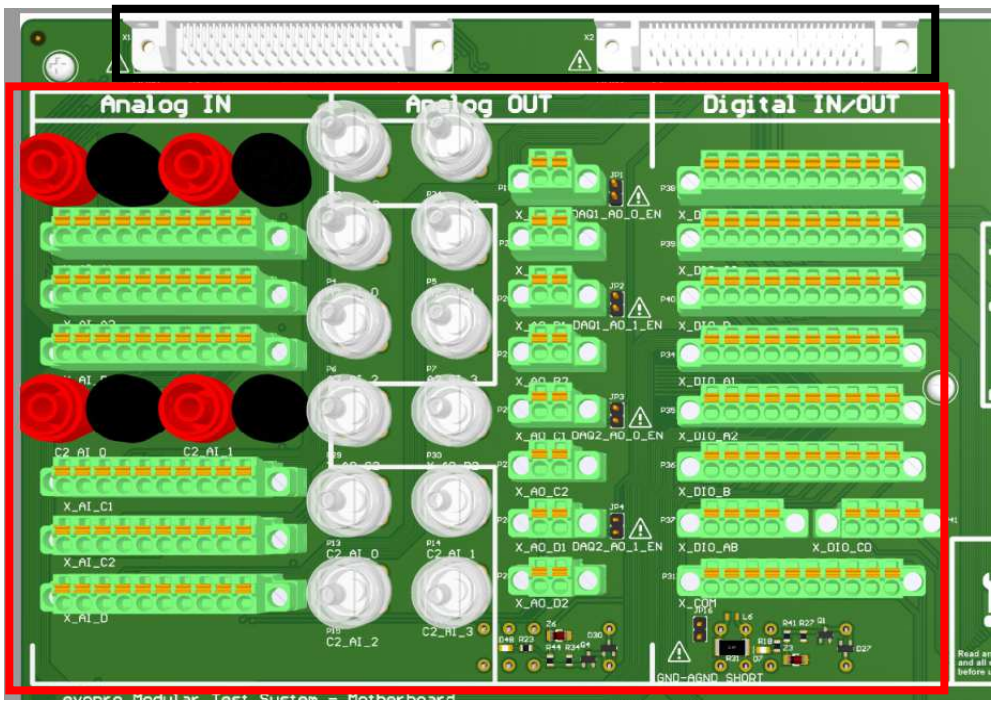


14. ábra: Tesztműszer interfészáramkör 3D ábrája [23]



15. ábra: Alaplap kártya blokkvázlat [23]

Alapvetően két NI DAQ mérőmodul jeleinek szétosztását végzi az áramkör, amelyek egy-egy 68 eres, árnyékolt kábelen (SHC68-68-EPM [24], 16. ábra, fekete keret) kapcsolódnak az alaplapra. A tesztműszer interfész áramkör DAQ csatlakozói bármilyen NI DAQ kártyával kompatibilisek. Ezeken kívül lehetőség van banán és BNC csatlakozókon további műszereket (például: oszcilloszkóp, digitális multiméter) bekötni a rendszerbe vagy helyettesíthetők is az NI mérőmodulok vonalai sorkapcsos csatlakozók segítségével (16. ábra piros keret). A sorcsatlakozókon visszamérhetők a DAQ kártyák jelei vagy amennyiben nincs csatlakoztatva a DAQ kártya, akkor ezeken keresztül más eszközök analóg és digitális jelei is beköthetők.



16. ábra: Tesztműszer interfész áramkör részlet: NI DAQ mérőmodul csatlakozók (fekete keretben) és általános csatlakozók (piros keretben) [23]

A tesztműszer interfészáramkör maximális kapacitása a DAQ kártyák függvényében a következőképpen áll össze:

- digitális be- és kimenetek: 24 (DAQ1) + 24 (DAQ2) = 48,
- analóg kimenetek: 2 (DAQ1) + 2 (DAQ2) + 4 (sorkapocs/BNC) = 8,
- analóg bemenetek: 16 (DAQ1) + 16 (DAQ2) + 8 (sorkapocs/BNC/banán) = 40,
- kommunikációs vonalak: egy-egy I2C- (sorkapocs), SPI- (sorkapocs) és USB busz (USB csatlakozó).

A DAQ1 jelei az A-B, a DAQ2 pedig a C-D helyhez rendelt csatlakozón érhetők el. Minden csatlakozási helyen található egy-egy tápvonal, digitális és analóg jeleket

csoportosító csatlakozó. A jelek kiosztása nem egyenletes, az A és a C hely kiemelt szereppel bír. Az 1. táblázat foglalja össze a huzalozási viszonyokat:

Kártyahely	A és C	B és D
Digitális be- és kimenetek	16	8
Analóg kimenetek	2	2
Analóg bemenetek	8	8
Kommunikáció	I2C, SPI, USB	I2C

1. táblázat: Feltétkártya helyek kapacitása [23]

Az alaplapra 3 független tápegység csatlakoztatható. Ezekből kettő a tesztelt komponenst, egy magát az alaplapot látja el, amelyből az interfészáramkör további feszültségszinteket állít elő (24V, 12V, 5V, 3,3V és -12V) tápegység IC-k segítségével. Mindegyik elérhető az összes kártyahelyen.

Az alaplapra bejövő tápvonalakon egy-egy P-FET-es, fordított polaritás elleni védelem található az áramkörön. A DAQ kártyák analóg vonalaira legfeljebb +/-12V-os feszültséget lehet kapcsolni, e fölötti értékek megjelenését diódás védelem akadályozza meg. A digitális be- és kimeneti vonalak védelméről (maximum 5V-ra korlátozás) a feltétkártyákon szükséges gondoskodni.

Az interfész áramkörre több eszköz csatlakoztatása lehetséges, amelyek külön-külön földvonalakkal rendelkeznek. A következő földrétegek vannak megkülönböztetve:

- AGND: az alaplapi analóg földréteg (ide csatlakoznak az NI DAQ kártya analóg kimeneteinek földjei),
- AIGND: az NI DAQ kártyák analóg bemenetei és az általános csatlakozókon bekötött mérőeszközök földje,
- GND: az alaplapi tápegység és a digitális be- és kimeneti vonalak földje,
- TEST_GND_1: a tesztelt áramkört ellátó egyik tápegység földje,
- TEST_GND_2: a tesztelt áramkört ellátó másik tápegység földje.

Ezeket relé kapcsolások segítségével közösíteni lehet, annak függvényében, hogy az adott mérési elrendezés során milyen igények merülnek fel. A relék kezelése egy I2C buszon vezérelhető portbővítővel [25] vagy külső, 3,3V-os logikai magas jelszintet használó digitális vonalakkal történik. A következő összekapcsolások lehetségesek:

- GND – TEST_GND_1,
- GND – TEST_GND_2,
- GND – AGND,
- AGND – AIGND.

3.2 Rendelkezésre álló szoftvermodulok

A kódmodulok NI LabVIEW-ban, a tesztek NI TestStand-ben kerültek implementálásra. A következő alfejezet célja az NI TestStand környezet rövid bemutatása (3.2.1. fejezet). A 3.2.2. fejezet a rendelkezésemre álló tesztszoftver architektúráját ismerteti.

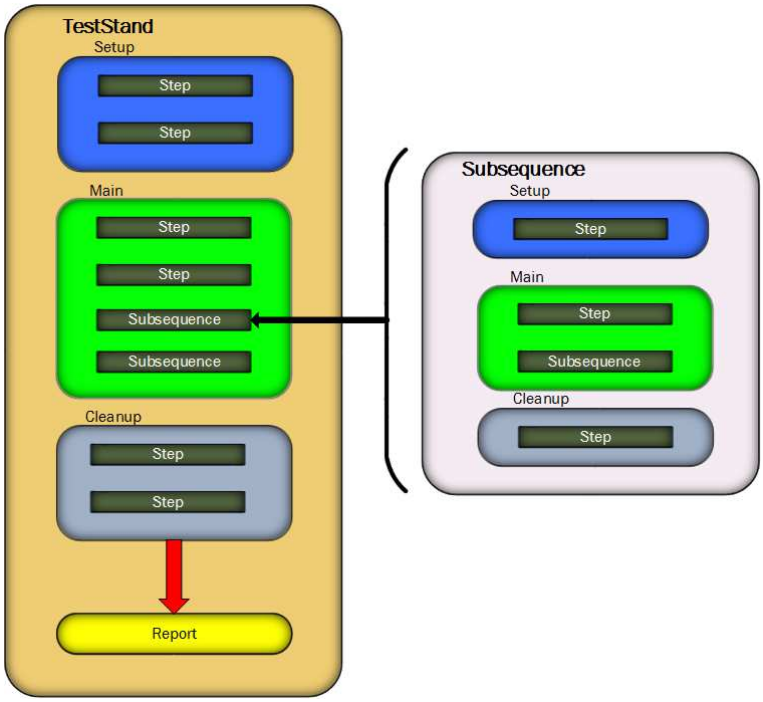
3.2.1 National Instruments TestStand

A TestStand [5] a National Instruments PC-s környezetre fejlesztett ipari standard tesztmenedzser szoftvere. Célja az automatizált tesztelési és hitelesítési rendszerek fejlesztésének felgyorsítása. Lehetővé teszi különböző programozási nyelveken (például LabVIEW, C, .NET) írt tesztkódok egy környezetből futtatását, kiértékelését.

A tesztszoftver alapvető elemei a tesztlépések (Steps). A tesztlépéseket szekvenciákba (Sequences) szervezetten lehet végrehajtani. Egy TestStand szekvencia három szakaszból áll (17. ábra), ezek futtatási sorrendben:

1. Setup: előkészítés, például erőforrás-foglalás.
2. Main: tesztlépések futtatása, végrehajtása.
3. Cleanup: tesztek futásának lezárása, például erőforrás-felszabadítás.

Egy tesztlépés végrehajtása jelenthet alszekvencia (Subsequence) hívást is, amely bármely szakaszában történhet a futó tesztnek. Az alszekvencia felépítése is ugyanolyan, mint a főszekvenciáé, ez is tartalmazhat további alszekvenciákat.

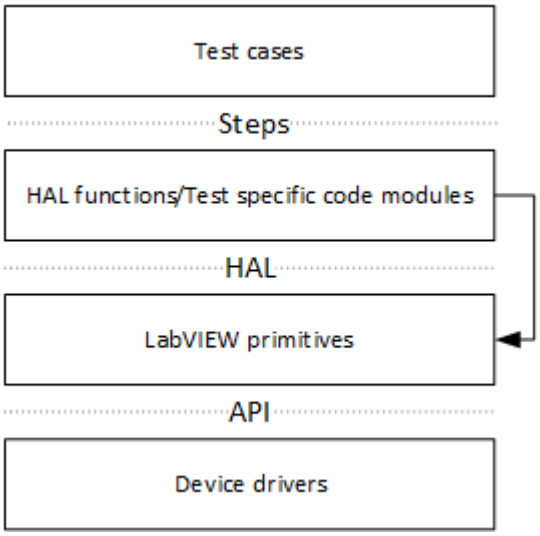


17. ábra: Főszekvencia felépítés TestStand környezetben

A TestStand környezet a tesztfuttatásokról jegyzőkönyvet (Report-ot) készít. A dokumentáció formátuma ASCII, HTML vagy XML lehet. Minden egyes tesztlépéshez meg lehet adni egy elfogadási kritériumot, amely alapján a szoftver el tudja, hogy dönteni sikeres volt-e a lépés végrehajtása. Az elfogadási kritérium mellett meg lehet adni további naplózandó paramétereket az egyes lépésekről. Például áramfelvétel mérése esetén tudni szeretnénk, hogy mekkora volt a mért érték, nem csak teszt végkimenetelét (sikeres vagy sikertelen) akarjuk elmenteni.

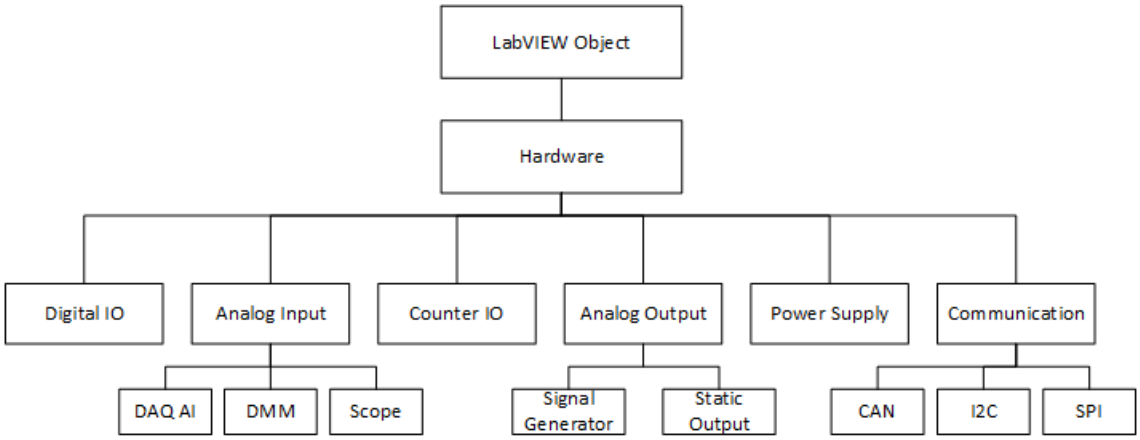
3.2.2 Tesztszoftver felépítése

A rendelkezésemre álló tesztrendszer magába foglalta a hardvermodulok kezelésére alkalmas szoftvert. Felépítését tekintve rétegekre van bontva (18. ábra), legfelül találhatók a tesztesetek, amelyek TestStand-ben implementált szekvenciákból állnak. Az egyes szekvenciák lépései a hardver absztrakciós réteg (HAL) metódusok vagy a teszt specifikus kódmodulok hívásaiból állnak. A HAL egységes kezelőfüggvényeket biztosít a LabVIEW primitívekben implementált gyártói LabVIEW API-k, függvénykönyvtárak, vagy ha ezek nem álltak rendelkezésre, akkor az eszközkezeléshez szükséges kódmodulok eléréséhez. A teszt specifikus kódmodulok a hardver absztrakciós réteg kikerülésével, direkt hívásokat valósítanak meg a LabVIEW primitívek felé.



18. ábra: Tesztszoftver egységei

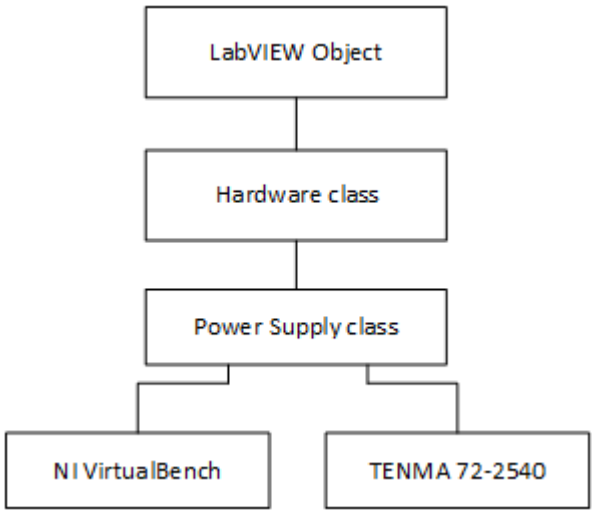
Azért van szükség a HAL rétegre, mert, ha hardverközeli szoftver módosítására vagy újabb eszköz integrálására van szükség, akkor ezek a változások nem érintik a felső szoftver szinteket. A hardver absztrakciós réteg LabVIEW-ban készült, kihasználva az objektum orientált programozást lehetővé tevő elemeket. A HAL felépítését a következő ábra mutatja (19. ábra). Mindegyik típusú eszköz közös ősosztállyal (Hardware) rendelkezik. Ennek előnye, hogy minden leszármazott hasonló lesz egymáshoz, egységesek lesznek a kezelőfüggvényeik. A felépítést bemutató osztálydiagram ágainak legalsó elemei után már a konkrét implementációk következnének, de ezeket az ábra nem tartalmazza.



19. ábra: Objektum orientált hardver absztrakciós réteg felépítése

Konkrét leszármazott implementációkra ad példát a 20. ábra. Egy osztályba tartozik a VirtualBench és a TENMA tápegység is, kezelésükhöz a közös szülőosztályuk (Power Supply class) metódusait kell meghívni. A metódusok hívása csak az alkalmazott

eszközt kijelölő paraméterezésben tér el (például: TENMA, 1-es csatorna vagy VirtualBench, 2-es csatorna), ez teszi lehetővé a kódmodulok újrahasznosítását.



20. ábra: HAL példa, tápegység osztály

A HAL réteg osztályai kapcsolatban is állhatnak egymással. Erre jó példa a tesztműszer interfészáramkörön található, földkapcsoló relék vezérlését végző I2C-s portbővítők kezelése. Az implementáció a digitális IO osztályba tartozik, ennek köszönhetően a HAL fölötti szoftver rétegek az I2C portbővítő áramkörök egyes portjait ugyanolyan digitális portokként kezelik, mint például az NI DAQ kártyáét. Azonban az implementációban a függvények az I2C osztály metódusait hívják meg.

3.3 Megvalósított továbbfejlesztések a tesztrendszeren

Az evopro Innovation Kft. tesztelési környezete munkám során volt először alkalmazásban. Ebben a pontban összevontan fogom leírni a hardveres és szoftveres módosítási javaslataimat, valamint a tapasztalt problémákat és az arra adott megoldásaimat.

3.3.1 Multifunkciós adatgyűjtőkártya

A rendszerben alkalmazott NI DAQ kártya nem rendelkezik analóg kimenettel, de az interfészáramkör és az interfészáramkört a DAQ modullal összekötő kábel fel van készítve erre a funkcióra. A legkézenfekvőbb megoldás egy másik adatgyűjtő kártya vásárlása, amely rendelkezik analóg kimenettel, például: NI PXIe-6341 [26]. Jelen esetben a projekt költségkerete erre nem adott lehetőséget, ezért a VirtualBench egyik programozható tápkimenetét használtam erre a célra (statikus jelekre volt szükségem).

Az integrált mérőműszer függvénygenerátora is használtható analóg kimenetként, azzal akár szinusz-, háromszög- és négyszögjel is előállítható, a DC feszültségértéken kívül.

A töltésvezérlő tesztrendszer fejlesztése során beleütköztem abba a problémába, hogy a digitális IO vonalak teszteléséhez nem lett volna elegendő a DAQ vonalainak a száma. Ezt a problémát is megoldotta volna még egy adatgyűjtő kártya, de az anyagi megfontolások miatt I2C buszon kommunikáló portbővítőket [25] alkalmaztam, hogy elérjem a megfelelő számú digitális be- és kimenetet.

3.3.2 Programozható tápegységek

A TENMA tápegységek megkülönböztetése a tesztrendszer szempontjából elkerülhetetlen, mivel általában különböző feszültségszintek kiadására van szükség. Az egyes eszközök azonosításával a következő problémák léptek fel:

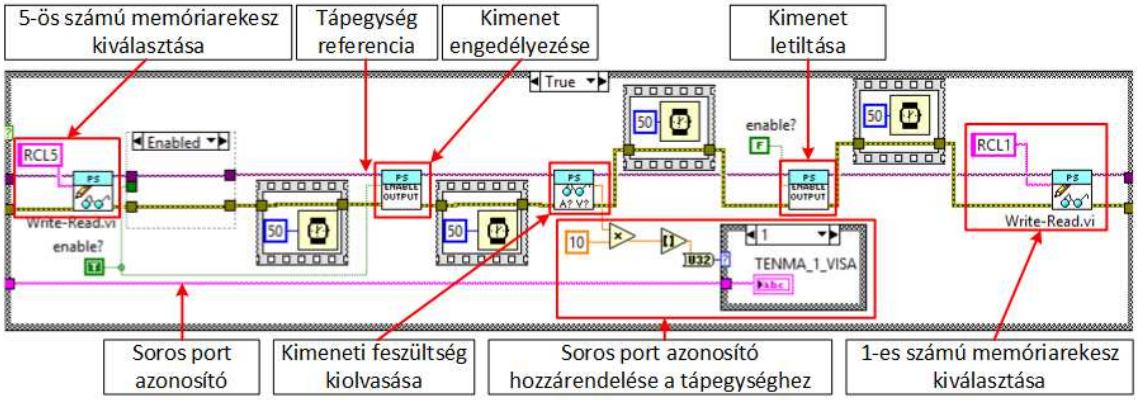
- nem rendelkeznek egyedi, programozottan lekérdezhető azonosítóval, mindkét eszköz ugyanazt az azonosító sztringet adja vissza,
- bekapcsolási vagy csatlakoztatási sorrendtől függ, hogy melyik COM porton melyik tápegység kezelésére ad lehetőséget az operációs rendszer.

További hibája az eszköznek, hogy a vezérlése instabil. Ez azt jelenti, hogy nem minden esetben hajtja végre a kiadott parancsot, hiába adta ki vezérlési utasítást a felhasználó. Ez egy kritikus hiba a tesztrendszer működtetése szempontjából.

Ezen hátrányok alapján levonható a következtetés, hogy érdemes volna más tápegységet alkalmazni. Előnyt jelentene a tesztrendszer szempontjából, ha a három tápegységet (egy VirtualBench és két TENMA) egy eszköz helyettesíthetné, tehát legalább három csatornás típusra volna szükség, mint például az R&S HMC8043 [27]. Ez viszont nem fért volna bele a projekt költségkeretébe, ezért más megoldásokhoz kellett folyamodnom.

Az első problémának a megoldására a TENMA tápegységek ötödik, csak programozottan elérhető memóriarekeszét használtam. Mindkét eszköz esetén beírtam ebbe a konfigurációs mezőbe egy kis feszültségértéket (egyik tápegység esetén 0,1V, másiknál 0,2V) és 0A-re állítottam az áramkorlátot. Ezek az adatok a hozzá nem értő felhasználótól védve vannak, felülírni nem lehet őket a nyomógombos kezelőfelülettel, csak programozottan. Ezután az automatizáltan végzett azonosítás a következőképpen implementáltam:

1. ötödik memóriarekesz előhívása, kimenet engedélyezése,
2. kimeneten megjelenő feszültségérték visszaolvasása, kimenet letiltása,
3. kiolvasott érték alapján a tápegység azonosítása, az egyes memóriarekesz előhívása.



21. ábra: TENMA tápegység azonosítás (LabVIEW kódrészlet)

Ezzel a módszerrel mindaddig megoldható a tápegységek megkülönböztetése, amíg különböző értékek szerepelnek a memóriarekeszekben. Ez a kódmodul egy teszt specifikus függvény, tehát a hardver absztrakciós réteg megkerülésével direkt hívja meg a LabVIEW primitíveket.

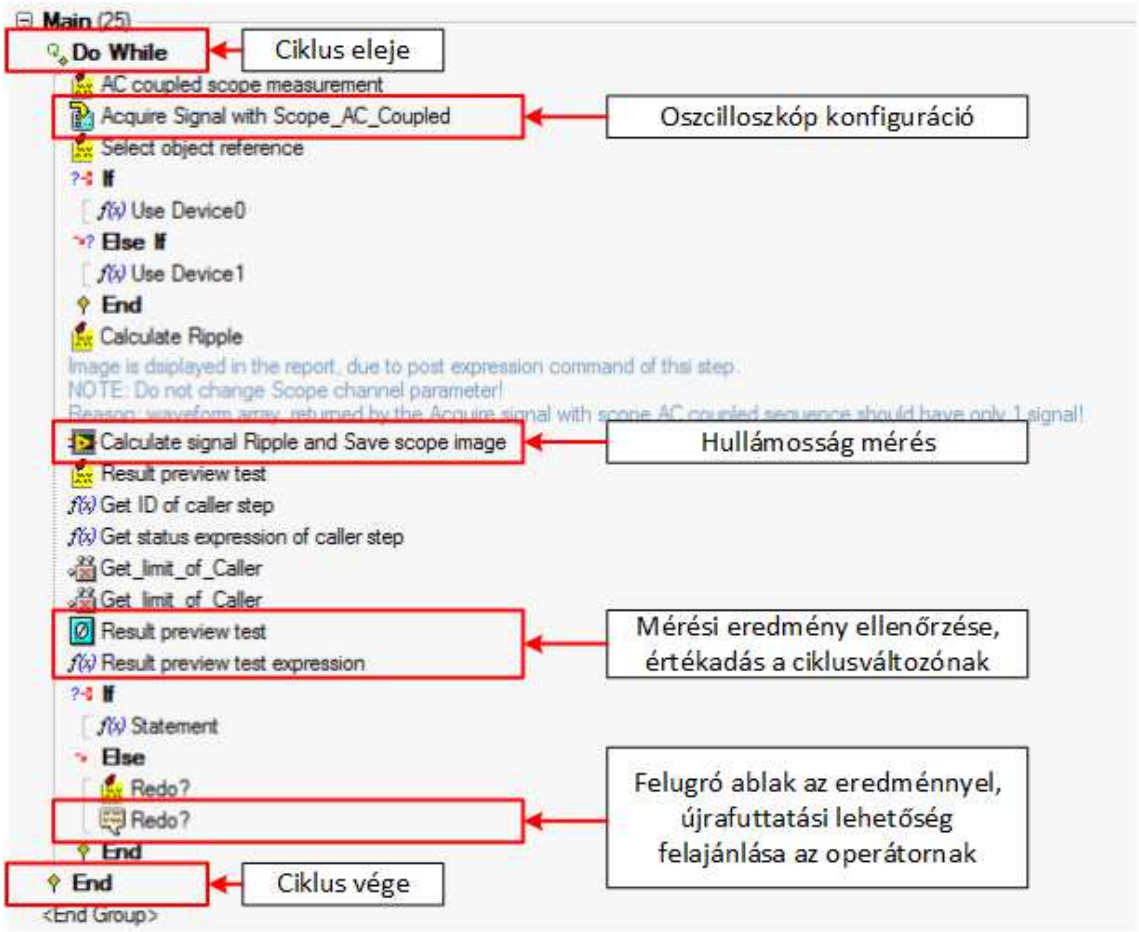
A nem megbízható működésre más megoldást nem találtam, mint minden egyes parancskiadás után egy lekérdezés futtatása, hogy sikeres volt-e a művelet. Ha ez nem teljesül, akkor a szoftver újrapróbálkozik.

3.3.3 Táptesztek operátori hibáinak kiszűrése

A táptesztek (felfutási idő, feszültségszint, zajosság mérés) végrehajtásához operátori beavatkozás szükséges, mivel az oszcilloszkóp egyik mérőfejét a bemenő tápvonalra, a másikat pedig a vizsgált tápvonalhoz tartozó tesztpontra kell helyezni. A mérőfejek földpontjait a legközelebbi földhöz érdemes csatlakoztatni, a minél kisebb zajjal terhelt mérések érdekében. Amennyiben ez például tüskesorra csíptetéssel nem oldható meg, abban az eseten rugós mérőfejet kell alkalmazni. A rugót egy közeli földpontnak kell nyomni, amely akár lehet egy közeli áramköri elem lábkivezetése is. A mérés lefutása alatt egyik mérőfej sem mozdulhat el.

Mindkét módszer magában hordozza a hiba lehetőségét akár a mérőfej, akár a föld összeköttetés elmozdulásában. Ennek a problémának az orvoslására módosítottam a

tápméréssel kapcsolatos szekvenciákat. Mielőtt a szoftver továbblépne a következő mérésre, összehasonlítja a limitekkel az aktuális mért értéket (22. ábra). Ha a mérés eredménye nem esik bele a tartományba, akkor egy felugró ablak segítségével a tesztszoftver megjeleníti az operátor számára a határ- és a mért értékeket. Ekkor a mérést végző személy kiértékelheti az eredményt és eldöntheti, hogy megismételi-e a mérést.

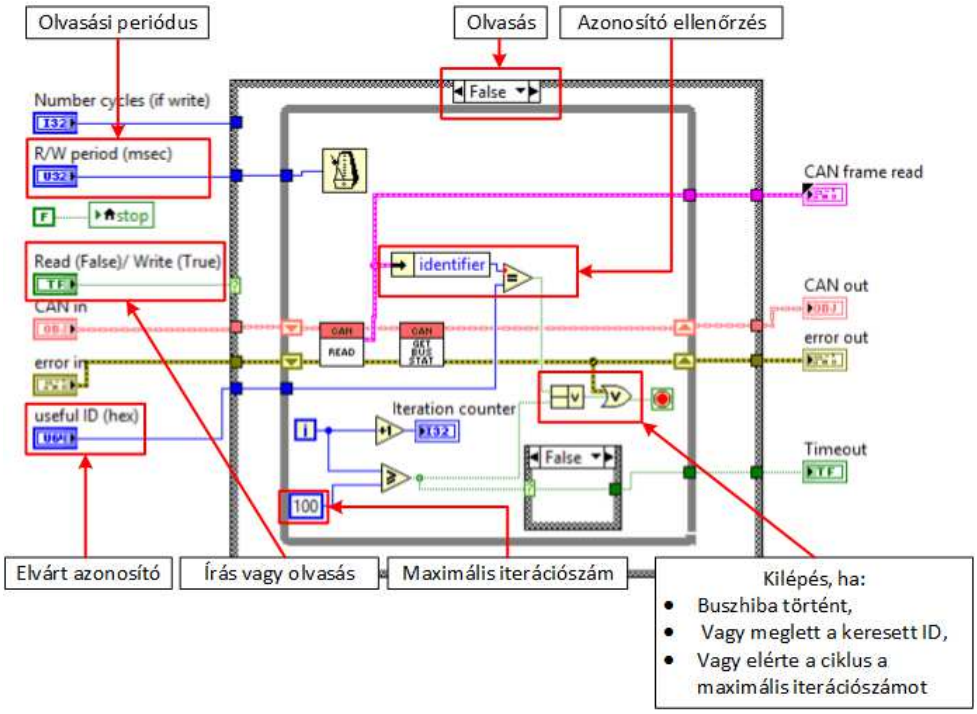


22. ábra: Zajosság mérés, példa az operátori hiba kiszűrésére

3.3.4 CAN kommunikáció kiegészítése

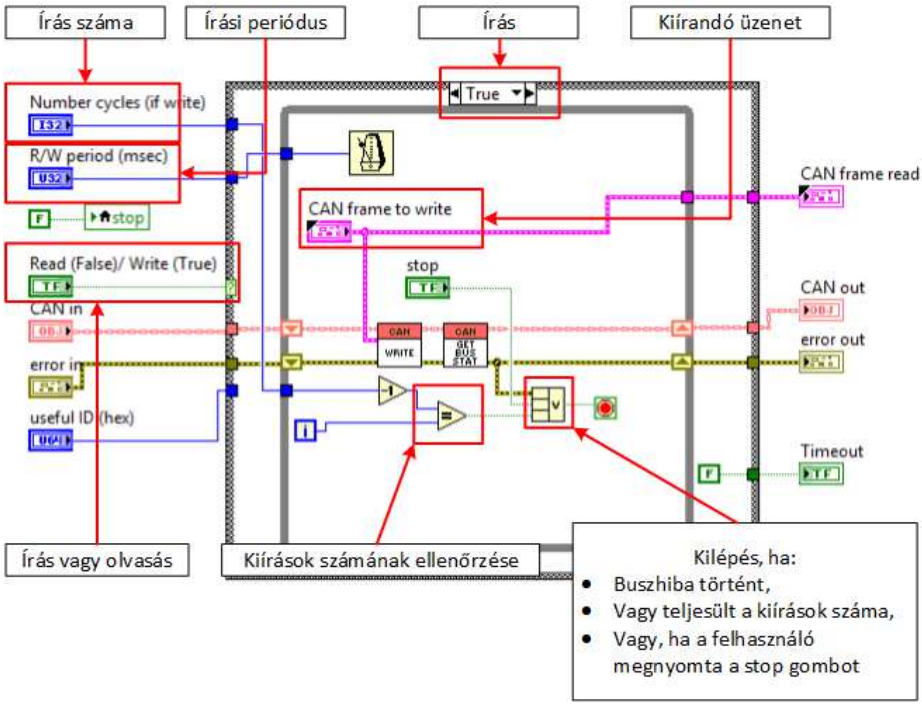
A töltésvezérlő áramkörrel a tesztrendszer CAN buszon, kérés-válasz jellegű üzenetváltásokkal kommunikál, ehhez mindkét félnek ismernie kell az alkalmazott azonosítókat. A tesztrendszerben logikusnak láttam kialakítani egy olyan metódust, amely tudja, hogy milyen ID-val rendelkező üzenetet kell fogadnia. A szoftvermodul (23. ábra) adott periódusonként megvizsgálja érkezett-e újabb üzenet az alkalmazott interfészen. Ha igen, akkor ellenőrzi az azonosítóját is, hogy az elvárt üzenet érkezett-e be. Ha igen, akkor kilép és visszatér az üzenettel a függvény. Ha nem érkezett be a

keresett ID-val rendelkező üzenet, akkor folytatja az adott periódusonként megismételt lekérdezést, egészen 100 lekérdezésig, ezután időtúllépés hibával visszatér a függvény.



23. ábra: CAN olvasás adott azonosítóig

A függvényt kiegészítettem úgy, hogy periodikus írása is alkalmas legyen (24. ábra):



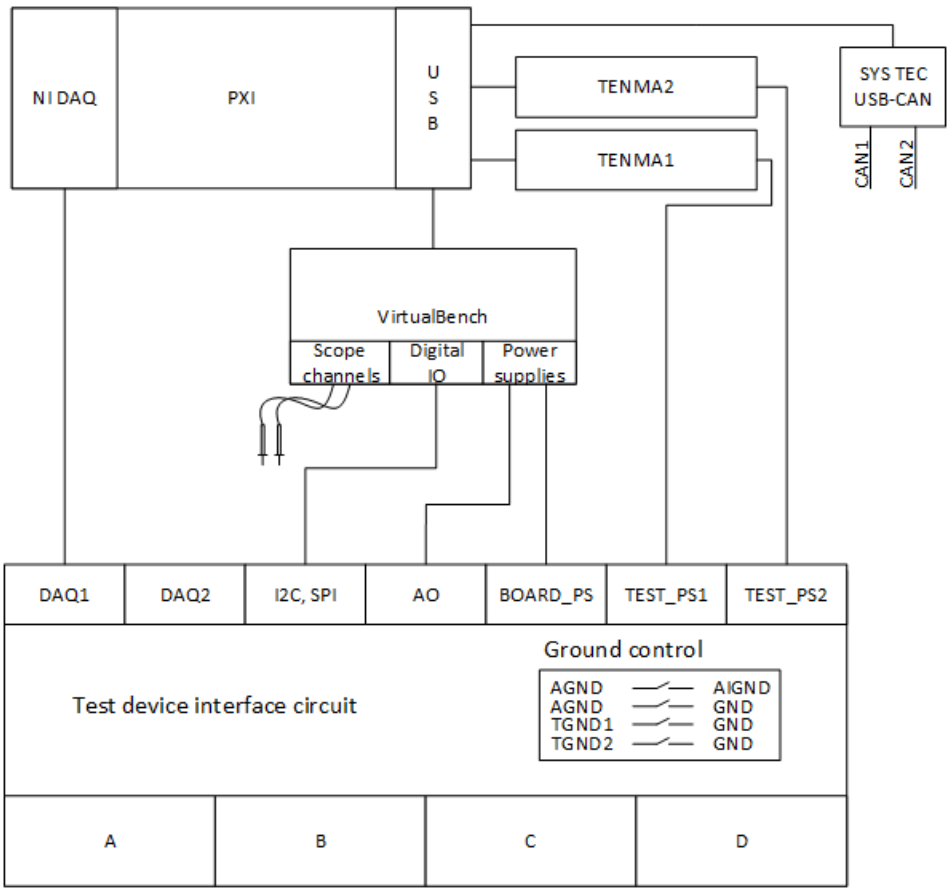
24. ábra: Periodikus CAN írás

4 Tesztrendszer felépítése

Ez a fejezet bemutatja, milyen infrastruktúra állt rendelkezésemre és ennek felhasználásával milyen tesztelési rendszert alakítottam ki, amelyet a töltésvezérlő tesztelésénél alkalmaztam.

A bemutatott eszközökből álló tesztrendszer blokkvázlatát a 25. ábra szemlélteti. Az eszközök bekötése a korábbi megfontolások alapján történt (3.3. fejezet).

A manuális tesztek alatt egy tápvonal vizsgálata során digitális multiméterrel mértem a jel feszültségszintjét, felfutását és zajosságát pedig oszcilloszkóppal. A tesztek gyorsításához a félautomata tesztelések során a multimétert kihagytam, helyette oszcilloszkóppal mértem a feszültszintet is. Így megtakarítható az az idő, amíg az operátor az oszcilloszkóp és a digitális multiméter mérőfeje között váltogat. Azért tehettem meg ezt a módosítást, mert a DMM-mel mért jelek stacionáriusak. A mérés során hosszabb ideig (300ms) mintavételeztem a jelet, majd ezeket a mintákat átlagoltam.

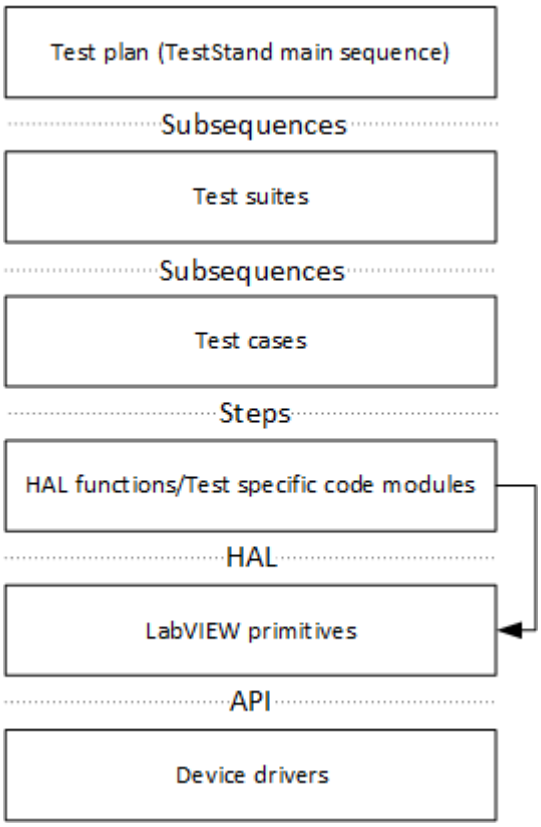


25. ábra: Tesztrendszer felépítése

Az ábra nem teljes, ahogy korábban bemutattam (3.1.6. fejezet) további eszközök csatlakoztatására is van lehetőség (például: BNC vagy banándugó csatlakozós oszcilloszkópok, digitális multiméterek, függvénygenerátorok stb.), de ezeket nem alkalmaztam a töltésvezérlő tesztelésekor.

4.1 Tesztmetódusok kialakítása

A 3.2.2. fejezetben bemutattam a meglévő tesztszoftvert. A töltésvezérlő teszteléséhez a meglévő szoftver komponensek módosítása, kiegészítése nélkülözhetetlen volt. A következő felépítést dolgoztam ki az átláthatóság és könnyű kezelhetőség érdekében (26. ábra):



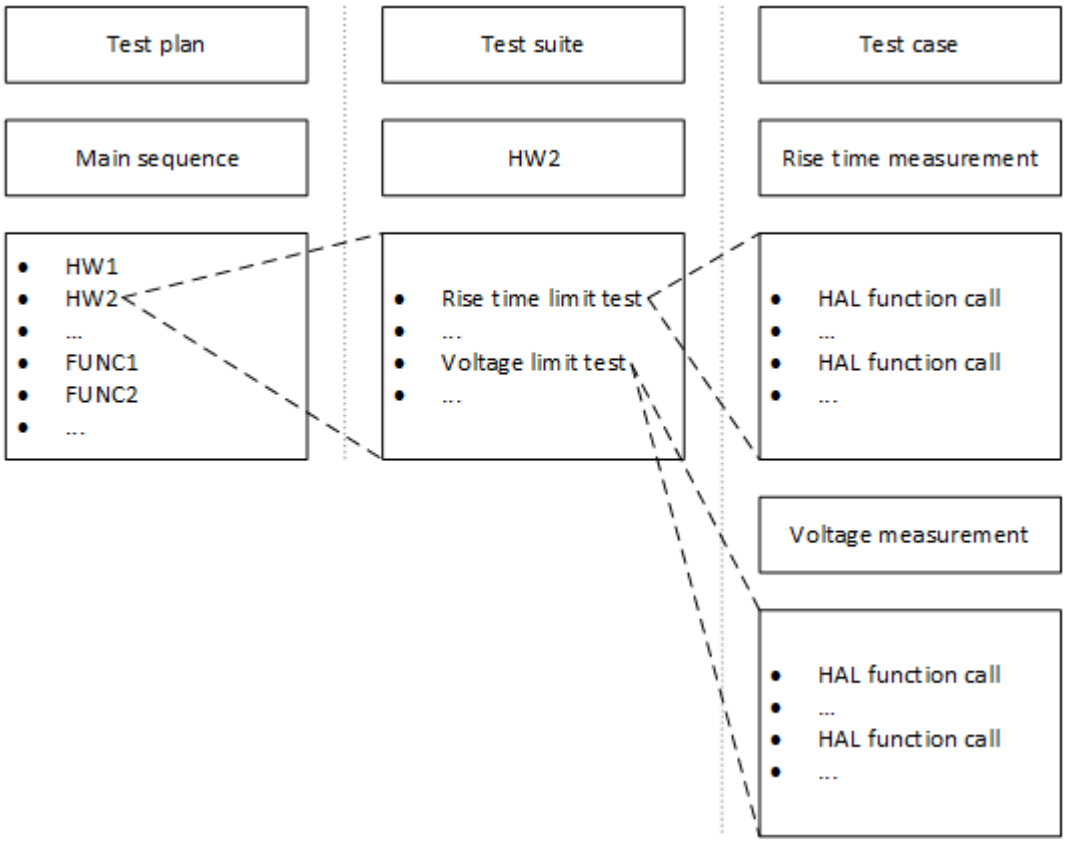
26. ábra: Tesztmetódus felépítése

Legfelül egy főszekvencia áll, amely valójában egy tesztelési terv. Ezen a szinten lehet a tervet konfigurálni: itt lehet megadni, hogy milyen egységeit szeretnénk tesztelni az eszköznek (például: táp, ADC, GPIO tesztek), de ennél alacsonyabb szinten (például elfogadási kritérium módosítása) nem lehet módosítani a komponenseket.

A tesztelési terv alszekvenciái, azaz lépések a főszekvencián belül, egy-egy tesztkészletet valósítanak meg. Egy tesztkészlet lehet például egy tápvonal vizsgálata. A

tesztkészletek lépései a tesztesetek, a tápegység teszt példánál maradva a feszültség, felfutási idő és zajosság mérés.

A tesztkészleteken belül van lehetőség az elfogadási kritériumok, tehát például a feszültségmérés limitértékeinek módosítására. A tesztesetek már tartalmaznak olyan modulokat, amelyek a hardver absztrakciós réteg metódusait hívják. Ezen a szinten szükséges konfigurálni a hívásokat, annak függvényében, hogy milyen hardverkomponensek állnak rendelkezésre.



27. ábra: Tesztszoftver példa

A 27. ábra szemlélteti a tesztszoftver felépítését. A tesztelési terv (Test plan) tartalmaz hardveres (HW1, HW2 stb.) és funkcionális (FUNC1, FUNC2 stb.) teszteket. Ezek valójában tesztkészletek (Test suite), amelyek tesztesetekből (Test case) állnak. Például a HW2-es tesztkészlet áll egy felfutási idő vizsgálatból (Rise time limit test) és egy feszültségmérésből (Voltage limit test), mindkét tesztesetet HAL függvényhívások alkotják.

5 Töltésvezérlő kártya teszthardvere

Első lépésben a követelményeket gyűjtöttem össze a töltésvezérlőről (5.1. fejezet), majd ezek alapján megterveztem a tesztáramkört (5.2. fejezet).

5.1 Követelmények

A **követelményanalízis** során rendelkezésemre álltak a fejlesztői dokumentációk, a manuális mérések eredményei, valamint maga a töltésvezérlő áramkör. A követelmények felvételéhez és az elfogadási kritériumok meghatározásához a kézi méréseket vettem alapul.

A **töltésvezérlő kártya tesztkövetelményeit** a 2. táblázat foglalja össze. A referencia- és tápvonalak vizsgálatához a korábban említett megfontolások alapján elegendő egy oszcilloszkóp használata. A műszer mindkét csatornájára szükség van, mivel a bemenő tápvonalhoz képest is vizsgálni szeretnénk a jel felfutását, hasonlóan a kézi mérésekhez.

A **fogyasztásmérés** kivitelezhető a tesztelendő áramkörrel sorosan bekötött DMM-es árammméréssel, vagy soros sönt ellenállás feszültségének mérésével, de legegyszerűbben programozható tápegységgel, amely vissza tudja mérni a kimeneti feszültséget, valamint a fogyasztó által felvett áramot (a rendelkezésre álló TENMA tápegységek képesek erre).

A tesztek során szükség van egy **kommunikációs interfészre**, a funkcionális tesztek elvégzéséhez. Az interfészen keresztül a tesztrendszer adatlekérdezési és vezérlési utasításokat képes adni. Például lekérdezhető, hogy a processzor mekkora feszültséget mér az analóg bemeneteken vagy megadható, hogy az adott PWM kimenet kitöltési tényezője 70% legyen. CAN és UART kommunikációs interfészek állnak rendelkezésre a töltésvezérlő processzoron. A töltőoszlopban CAN interfész van alkalmazva, ezért ezt kell tesztelni, a soros interfész debug és programozási célt szolgál. Ezek alapján a **CAN interfészen** keresztüli vezérlés mellett döntöttem.

A referencia- és tápvonalak **rövidzár vizsgálata** egyértelmű, ha az adott vonal nem az elvárt értéket veszi fel, akkor hibás, nem kell a többi vonalat figyelni mellette. Ezzel ellentétben az **ADC-, digitális I/O és PWM vonalak** áthallás vagy rövidzár vizsgálata szükséges, és a következőképpen valósítható meg:

1. Egy vonal vezérlése, a többi alapállapotban marad.
2. Értékek kiolvasása az összes vonalon.
3. Kiolvasott értékek összehasonlítása a vezérlési parancsban megadott értékekkel.
4. Az első három lépés megismétlése minden egyes kivezetésre.

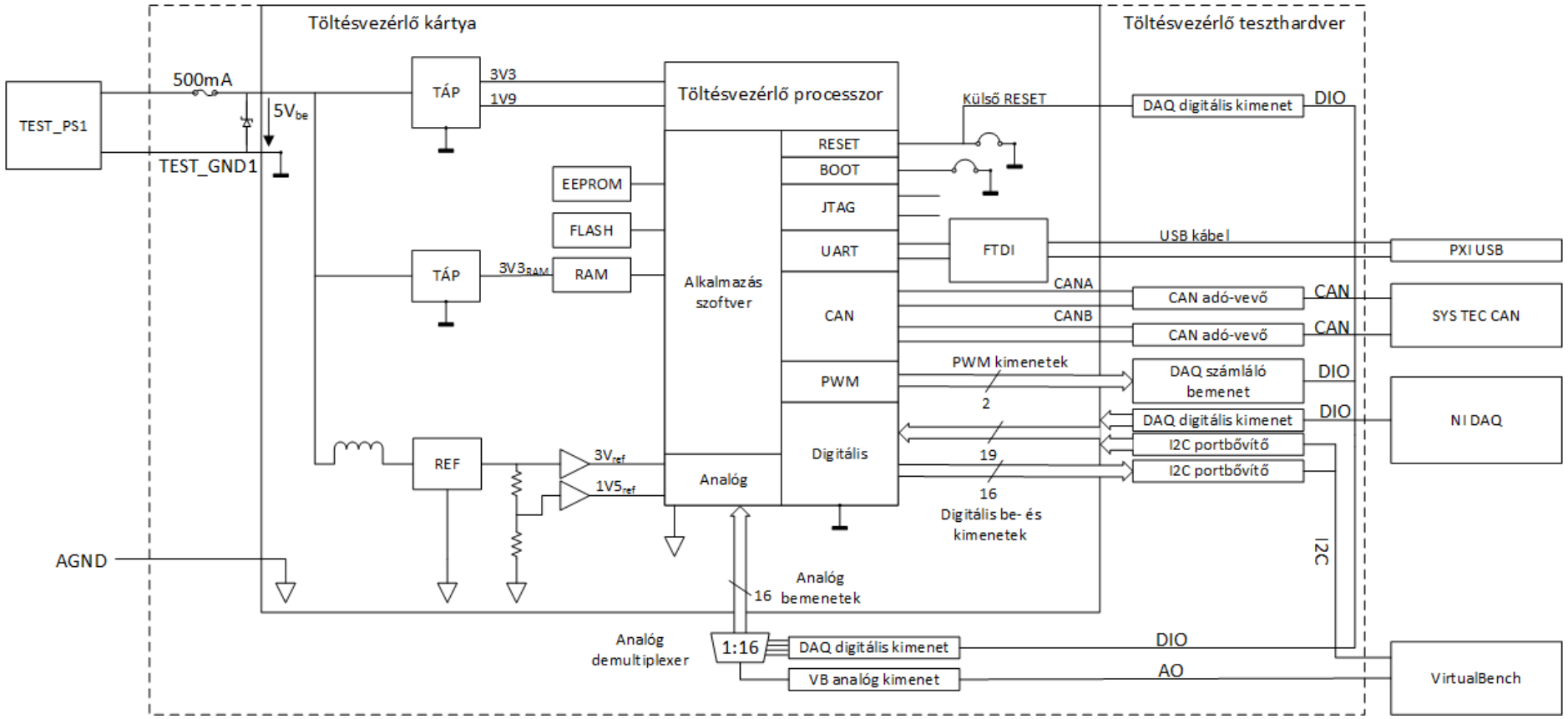
Tesztelendő egység	**Darabszám**	**Mérési feladatok**	**Szükséges eszközök**
Tápvonalak (3,3V, 3V, 1,9V)	3	érték, felfutás, zajosság, rövidzár	oszcilloszkóp
Referenciafeszültségek (3V, 1,5V)	2	érték, felfutás, zajosság, rövidzár	oszcilloszkóp
Áramkör fogyasztása	1	áramfelvétel	programozható tápegység
AD átalakítók	16	feszültség érték, áthallás/rövidzár	analóg jelforrás, CAN interfész
Digitális bemenetek	19	logikai érték, áthallás/rövidzár	digitális kimenet, CAN interfész
Digitális kimenetek	16	logikai érték, áthallás/rövidzár	digitális bemenet, CAN interfész
PWM kimenet	2	frekvencia, kiöltési tényező áthallás/rövidzár	időzítő/számláló bemenet, CAN interfész
CAN interfész	2	rövidzár, kommunikáció	CAN interfész

2. táblázat: Töltésvezérlő kártya tesztelés

A töltésvezérlő kártya **áramfelvételét** megmértem, a mért értékek 270mA körül szóródtak. Ez az érték akkor mérhető, ha fut a szoftver. Bekapcsolás során nagyobb áramfelvételt mértem, de ez tranziens jellegű volt. A tranziens áramfelvétel értékei 300mA +/- 10mA-es tartományban helyezkedtek el, de a vizsgálathoz alkalmazott multiméter átlagolta az értéket, ezért ennél nagyobb áramfelvételre lehet számítani. A TENMA tápegységek egészen 5A-ig terhelhetők, így ez nem okoz problémát.

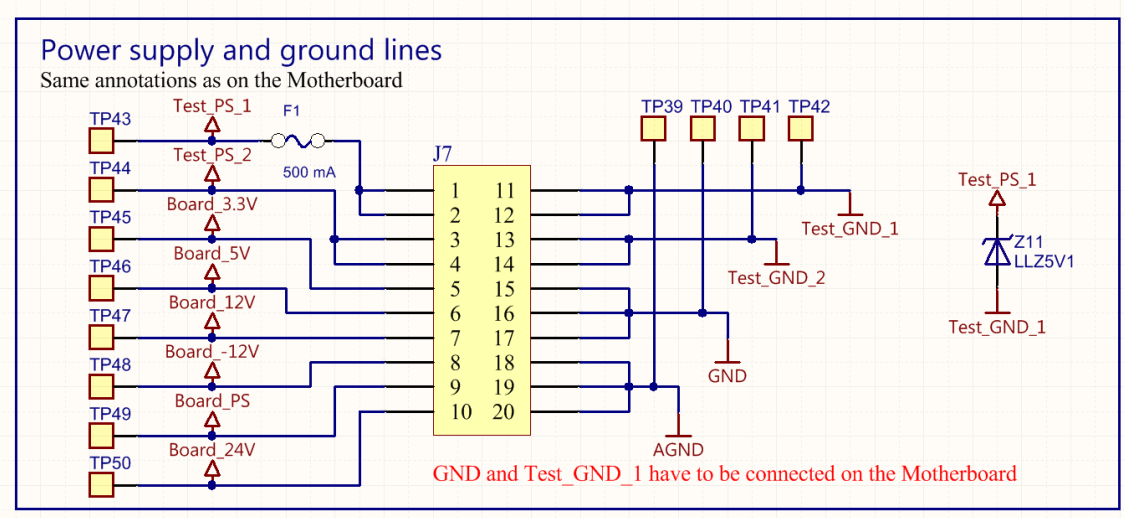
5.2 Töltésvezérlő kártya tesztáramköre

A felsorolt követelmények alapján megterveztem a töltésvezérlő teszthardverét, amelynek blokkvázlatát a 28. ábra mutatja.



28. ábra: Töltésvezérlő teszthardver blokkvázlata

A **tesztelt áramkör tápellátását** az 1-es számú TENMA tápegység végzi (29. ábra: Test_PS_1). Az áramfelvétel korlátozására egy 500mA-es, gyorskioldó olvadó biztosítót (F1) helyeztem el a bemeneti tápvonalon. Amennyiben rövidzár van a tápvonalakon, például a töltésvezérlő ültetési hibából adódóan, ez az alkatrész képes megvédeni a processzort a meghibásodástól. Szintén a tápvonalon elhelyezett 5,1V-os Zener dióda (Z11) védi meg az áramkört az esetleges túl magas feszültségtől (6V a megengedett maximális érték). A töltésvezérlő kártya digitális földje a TENMA tápegység földkivezetésére (Test_GND_1), az analóg föld pedig a tesztműszer interfészáramkör analóg földvonalára (AGND) van csatlakoztatva. A tesztkártya digitális áramköri elemei az alaplap digitális földjéhez (GND) kapcsolódnak. A tesztelő áramkör tápellátása független a tesztelt áramkör tápellátásától, erre a pontos áramfelvétel mérés miatt van szükség.

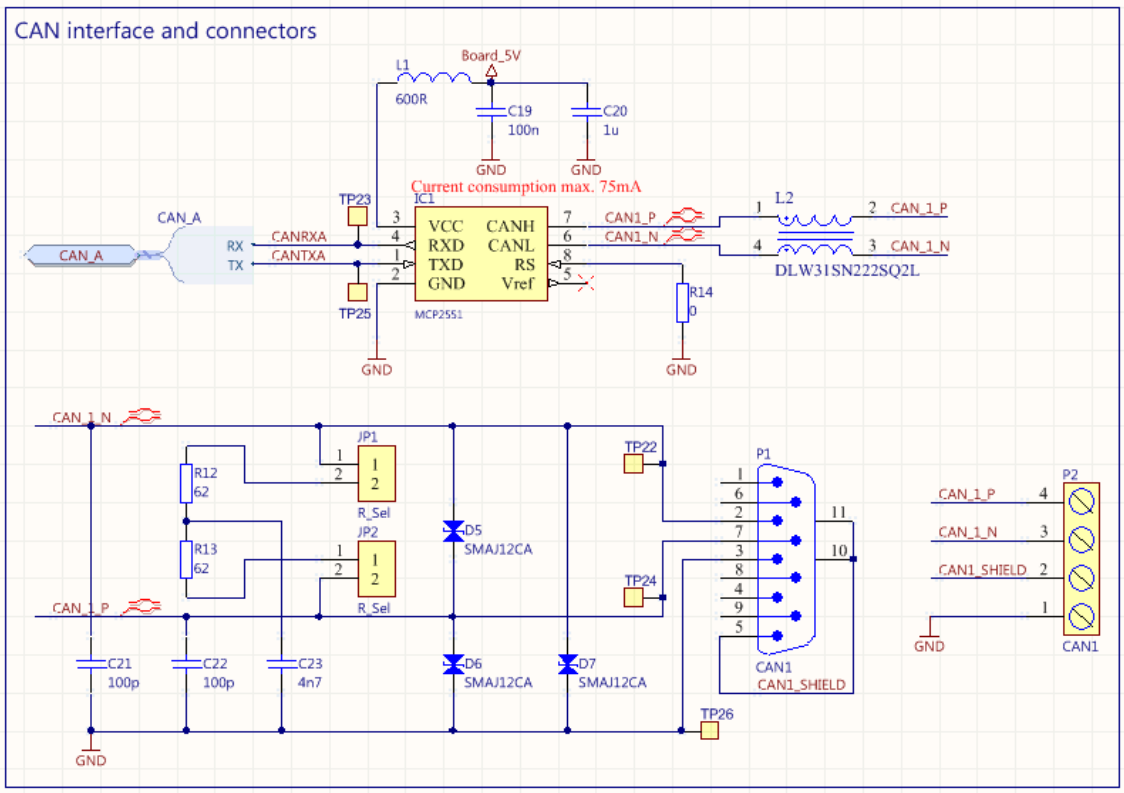


29. ábra: Táp- és földvonalak bekötése az alapi csatlakozón

Az **FTDI chip USB kimenete** közvetlenül a PXI ipari PC egyik USB portjára csatlakozik, ehhez nem volt szükség kiegészítő áramkörre.

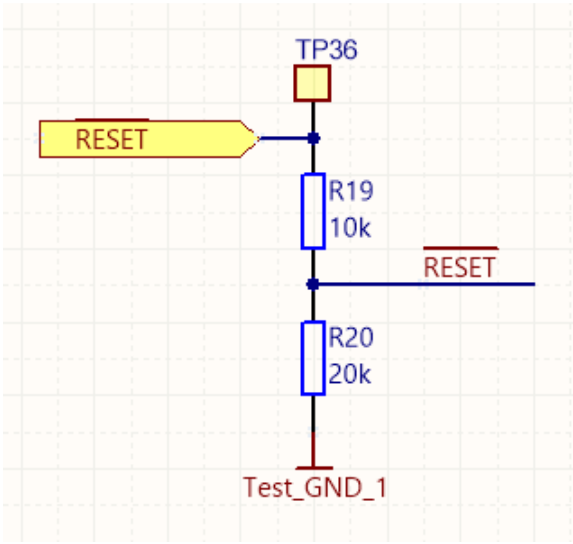
A töltésvezérlő áramkörön a **CAN vonalak** RX-TX formában érhetők el. Azért, hogy kommunikálni tudjon a processzor a SYSTEC USB-CAN átalakítón keresztül a tesztrendszerrel, a teszthardverre egy-egy CAN adó-vevő áramkört terveztem a két interfészhez (30. ábra). Microchip által gyártott MCP2551 CAN transciever-t [28] választottam. Ennek oka, hogy a processzor tényleges alkalmazásával egyező megoldást szerettem volna, mivel ez az IC található meg a töltésvezérlő perifériakártyáján is. A

közös módusú tekercs (L2) és a TVS diódák (D5, D6, D7) zavarelnyomás céljából kerültek az áramkörbe. A CAN vonalak D-Sub és 4-pólusú csatlakozón is elérhetők.



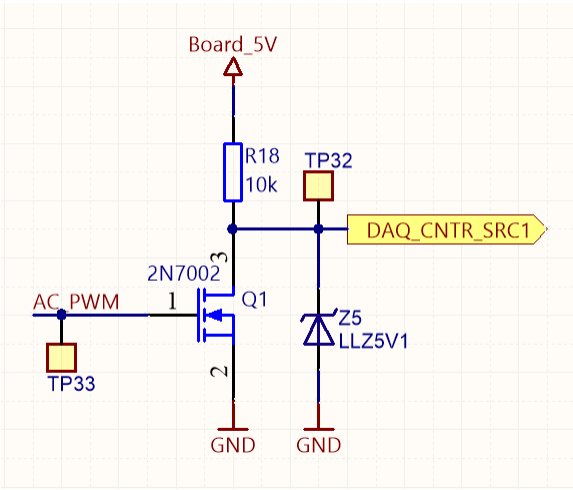
30. ábra: CAN adó-vevő áramkör (egy interfészre)

A processzor **külső RESET** vonalát egy NI DAQ digitális kimenet vezérli. A DAQ kártya digitális vonalai 5V-os, a töltésvezérlőé pedig 3,3V-os logikai magas szintet használnak. Ahhoz, hogy a két eszköz digitális vonalai összeköthetők legyenek, szintillesztést kellett végeznem a jelen, mert a DSP (töltésvezérlő processzor) legfeljebb 3,6V-ot tolerál a bemenetein. A szintillesztést ellenállásosztóval oldottam meg. Digitális multiméter segítségével lemértem, hogy legalább 120uA áram szükséges ahhoz, hogy a processzor digitális bemenetét 0V-os szintre állítsam, ezt az ellenállások méretezésekor figyelembe kellett vennem. A megvalósításban 10 és 20kΩ-os ellenállásokat használtam (31. ábra, R19 és R20), ekkor az ellenállásokon folyó áram 5V/30kΩ = 166uA lesz. Logikai magas jelszint esetén 5V*20kΩ/30kΩ = 3,33V fog a processzor bemenetére kerülni.



31. ábra: Külső RESET, szintillesztési példa

A DAQ kártya két számláló/időzítő be- vagy kimenettel rendelkezik, amelyeken képes élváltásokat és kitöltési tényezőt számolni, frekvenciát mérni, PWM jelet generálni stb. A számláló/időzítő funkcióval rendelkező vonalak beletartoznak a 24 digitális be- és kimenetbe, szoftveresen konfigurálható, hogy milyen feladatot lássanak el. A töltésvezérlő processzor **PWM kimeneteit** szintillesztettem a DAQ kártya számláló bemeneteihez (32. ábra):

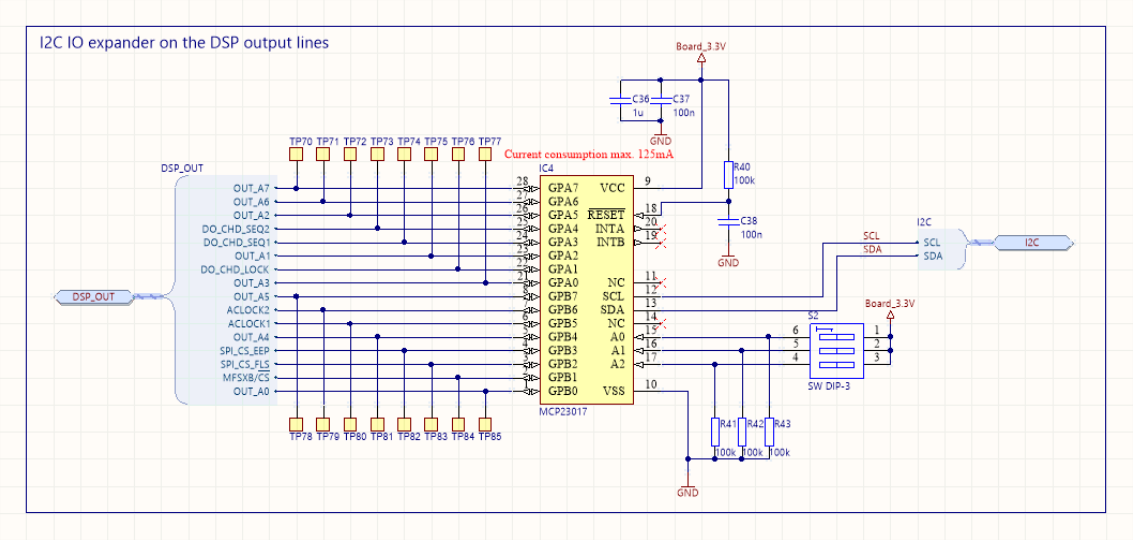


32. ábra: PWM kimenetek szintillesztése

Az áramkörben egy N-csatornás FET található, amely nyitott állapotában logikai alacsony szintet (0V) kapcsol a számláló bemenetre, zárt állapotban pedig a bemenet egy 10kΩ-os ellenálláson keresztül 5V-ra, logikai magas szintre van felhúzva. A vezérlést a FET Gate bemenetére kapcsolt 3,3V-os PWM jel adja, amely a kimeneten negáltan fog

megjelenni 5V-os jelszintekkel. Ezt a problémát szoftveresen orvosoltam. A Zener dióda a digitális vonal védelmét biztosítja magas feszültségek ellen.

A töltésvezérlőnek **16 digitális kimenete** és **19 digitális bemenete** van. Az összes jelutat egy DAQ eszköz digitális IO vonalaival nem lehet lefedni, ezért I2C buszon vezérelhető portbővítőket alkalmaztam (MCP23017 [25]). A Microchip által gyártott IC két 8 bites portot tartalmaz, tehát 16 vonal vezérlésére vagy olvasására képes. Három címbeállító vonallal rendelkezik, így egy I2C buszon összesen 8 chip kezelhető függetlenül. Mind a bemeneti, mind a kimeneti jelvonalakhoz illesztettem egy-egy portbővítő IC-t (illesztési példa: 33. ábra), melyek címbeállítását egy-egy kapcsolóval (33. ábra: S2) lehet megadni. A fennmaradó 3 bemenetet NI DAQ digitális kimenetekre csatlakoztattam, ellenállásosztóval megvalósított szintillesztésen keresztül. Az ellenállások méretezését a korábban említett megfontolások alapján végeztem.

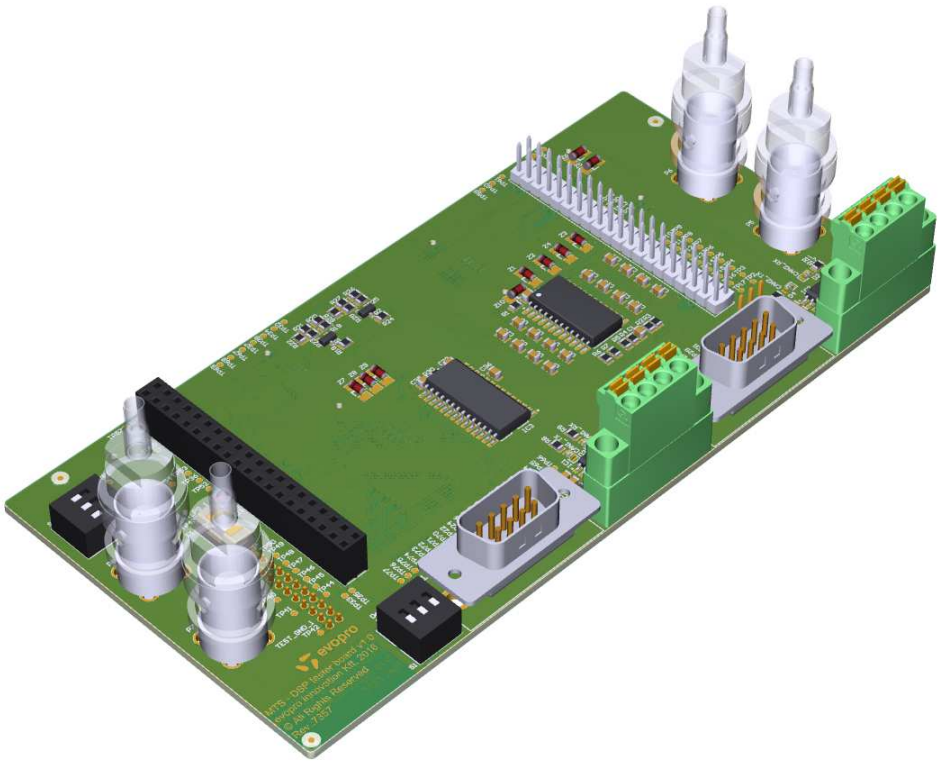


33. ábra: Töltésvezérlő processzor kimenetek, portbővítő példa

A processzor **analóg bemeneteihez** analóg demultiplexeres vezérlést valósítottam meg (Függelék, 76. ábra). Az analóg jelet a VirtualBench 6V-os tápvonala biztosítja, amelyet szükség esetén a függvénygenerátorával is lehet helyettesíteni. Minden jelvonal egy-egy ellenálláson keresztül analóg földre van kötve, így csak azon van 0V-tól különböző érték, amelyre demultiplexer bemenete van kapcsolva. A multiplexer vezérlését a DAQ digitális vonalaival oldottam meg. Választásom egy Texas Instruments által gyártott alacsony soros ellenállású (70Ω) 1:16-os multi/demultiplexerre esett (CD74HCT4067 [29]). A töltésvezérlő az analóg bemenetein legfeljebb 3V-t vár el, ezért egy Zener diódás védelmet helyeztem el a multiplexer bemenetén.

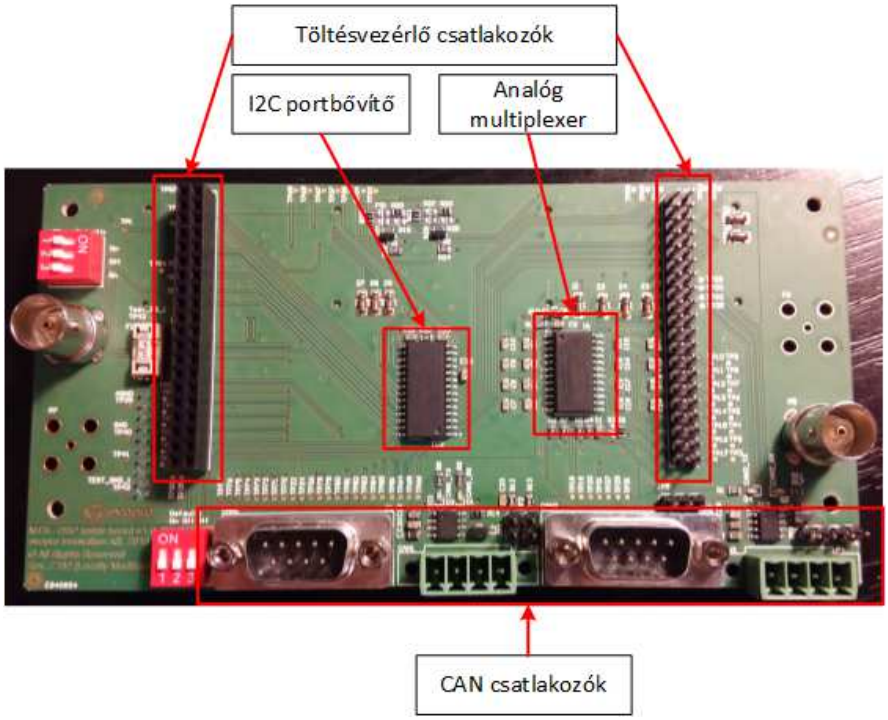
A feltétkártya fontos feladata a **műszervédelem**. Minden digitális vonalat, amely a tesztrendszer felől vagy felé megy Zener diódás védelemmel láttam el. Így, ha valamilyen hiba folytán a megengedett jelszintnél magasabb feszültség kerül ezekre a digitális vonalakra, akkor a tesztelő műszerek nem károsodnak. Az analóg jelek védelmét az alaplap biztosítja, tehát ezzel nem kellett foglalkoznom.

Az kapcsolási és a nyomtatott áramköri rajzokat az Altium Designer [30] tervezőprogramban készítettem. A 34. ábra mutatja a töltésvezérlő teszthardver tervezőprogram által generált 3D modelljét.



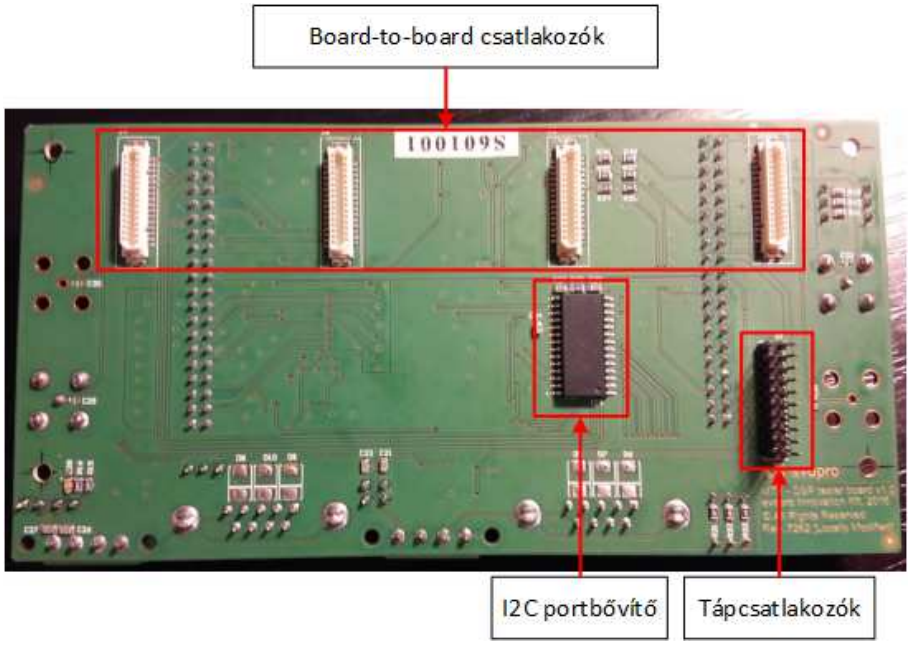
34. ábra: Töltésvezélő teszthardver 3D ábra

A 35. ábra mutatja a **megvalósított áramkört**. A hüvely- és tüskesorra csatlakozik a töltésvezérlő áramkör. A huzalozáshoz az alsó és a felső réteg elegendő volt, nem volt szükség belső rétegek kialakítására. A felső oldalon helyeztem el az analóg demultiplexert és az egyik IO bővítőt, valamint minden egyes mérendő jelhez egy-egy tesztpontot. A kapcsolók a portbővítők címbeállítását szolgálják, a D-Sub és a 4-pólusú sorcsatlakozón pedig a CAN vonalak érhetők el. A lezáró ellenállásokat a jumper-ek segítségével lehet bekötni a rendszerbe. A BNC kivezetéseken az alaplapra BNC vagy banándugós csatlakozón bekötött eszközök érhetőek el.



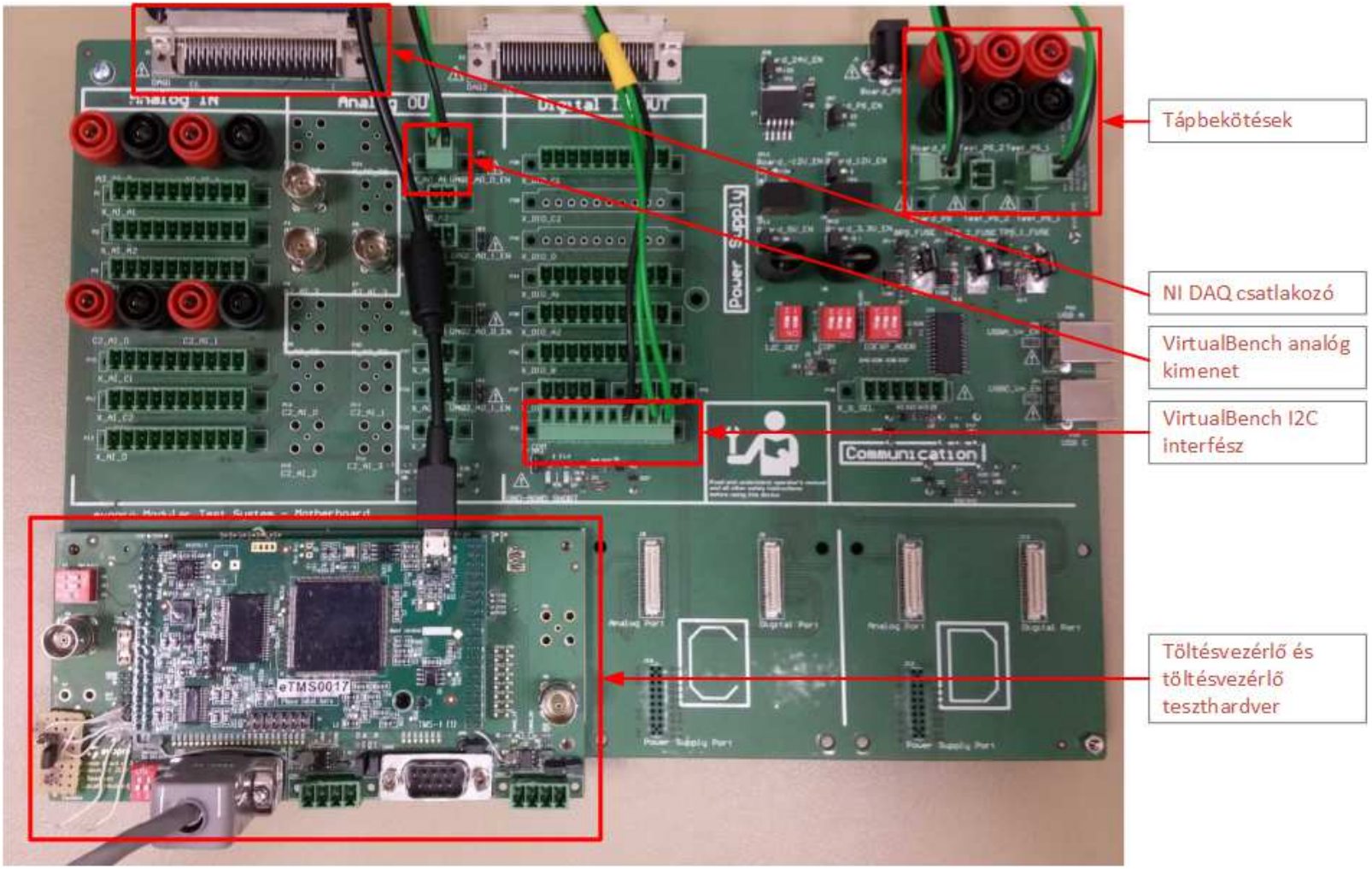
35. ábra: Töltésvezérlő tesztáramkör felülről

Az alsó oldalon (36. ábra) találhatók a fehér board-to-board csatlakozók, a tápvonalak biztosító tüskesor és a másik IO bővítő.



36. ábra: Töltésvezérlő tesztáramkör alulról

Az összeállított tesztrendszert a 37. ábra mutatja. A PCB bal felső részén található az NI DAQ csatlakozó, alatta a VirtualBench tápegységének 6V-os kimenete lett bekötve, mely ebben a tesztesetben analóg kimenetként funkcionál.

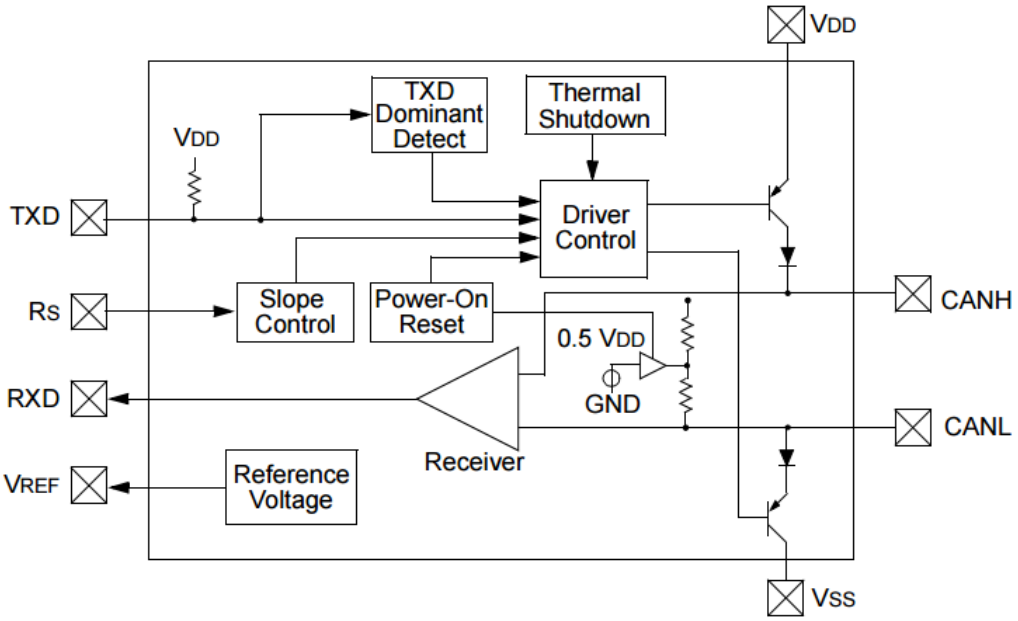


37. ábra: Összeállított tesztrendszer

A bal alsó sarokban található a töltésvezérlő, alatta az általam tervezett és felélesztett / beüzemelt tesztáramkör. A feltétkártya, az A és a B helyet is elfoglalja, mert a DAQ két időzítő/számláló bemenete külön-külön csatlakoztatási helyre esik a kártyahelyek kiosztása alapján. A töltésvezérlő fölött látható a VirtualBench I2C interfészének és az analóg kimenetének bekötése. A jobb felső sarokban láthatók a tápcsatlakozók. Mivel a töltésvezérlő csak egy bemenő táppal rendelkezik, így csak az egyik TENMA és az alaplapot ellátó 25V-os VirtualBench tápegység lett csatlakoztatva a mérési elrendezésben.

5.3 Tervezési tapasztalatok

A töltésvezérlő tesztáramkörén próbapaneles kiegészítésre volt szükség, mert a CAN adó-vevő belső felépítése következtében a TX vonal tétlen állapotában a tápfeszültségére van húzva egy ellenálláson keresztül (38. ábra):



38. ábra: MCP2551 blokkvázlat [28]

Az IC 5V-os tápot kap, így ez kerül a processzor CAN RX bemenetére. Ha emellett a processzor tápegysége le van kapcsolva, akkor ezen a tápvonalon 1,5V feszültség mérhető, az elvárt 0V helyett. Ennek oka az, hogy a töltésvezérlő, a bemenetén megjelenő feszültséget a bemenet védődiódáján keresztül a töltésvezérlő kártya tápvonalára csatolja, ahonnan a feszültség kikerül a táp IC bemenetére. A tápvonalon

megjelenő feszültség, az áramkör módosítása nélkül, meghamisíthatja a felfutás méréseket, mert a tápvonalakon található kondenzátorokat feltöltött állapotban tartja.

A módosítást követően a CAN adó-vevők a kiegészítő tesztpanelen keresztül kapják a tápellátást: egy jumper-rel megadható, hogy 5V vagy föld legyen a tápvonalaikra kapcsolva (37. ábra bal oldalán látható).

39. ábra: Javított töltésvezérlő teszthardver

A CAN vonalak nem kerültek szintillesztésre (5V-ról 3.3V-ra). Ez a működés során nem okozott problémát, a processzor kivezetésein található védődiódák elvezették a túlfeszültséget a tápvonalra, de egy következő verzióban érdemes volna javítani.

Az NI DAQ kártya digitális bemenetei TTL-kompatibilisek. Ez azt jelenti, hogy nagyjából 2V-os jelszinthez már logikai magas értéket rendelnek. Ebből kifolyólag a processzor PWM kimeneteinek szintillesztését nem lett volna szükséges elvégezni.

Még egy módosítást hajtottam végre, a felfutási idő mérések könnyebb elvégzéséhez. Ezeknél a méréseknél az oszcilloszkóp egyik csatornája a bemenő tápvonalat méri, míg a másik az aktuálisan vizsgáltat. Az operátor számára könnyebb, ha nem kell két mérőponton is egy-egy mérőfejet tartania, ezért egy-egy vezetéket forrasztottam a bemenő táp- és földvonal mérőpontjaira, így azokra az oszcilloszkóp mérőfeje felcsíptethető (39. ábra, bal oldalon látható fehér, szabad végű vezetékek).

6 Töltésvezérlő kártya tesztszoftvere

A töltőoszlop szoftvere alapesetben nem támogatja a funkcionális tesztek elvégzését, ezért egy új, a töltésvezérlőben futó programot kellett készítenem. Ezt a C nyelven megírt szoftvert a 6.1-es, a PC oldali tesztszoftvert pedig a 6.2-es fejezetben mutatom be.

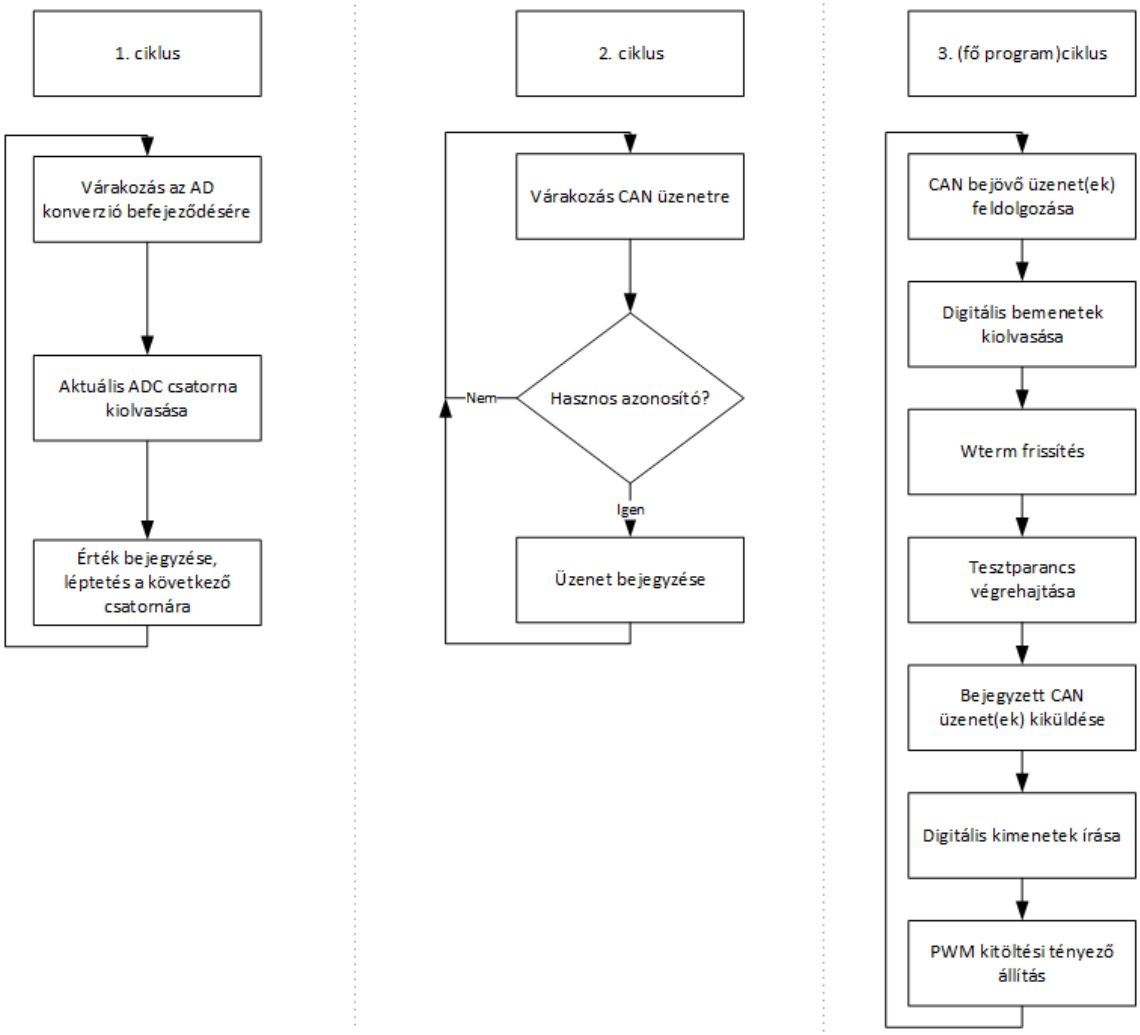
6.1 Processzor oldali tesztszoftver

A processzoron futó szoftvert alapvetően három különálló, párhuzamosan futó ciklusra lehet bontani (40. ábra).

Az **első ciklusban** az analóg bemenetek olvasása történik. A 16 csatorna multiplexálva érhető el, ezért minden egyes iterációban csak egy értéket olvas ki a processzor, amelyet elment egy tömbbe. A tömb 16 elemű, mindegyik csatornának a legutoljára kiolvasott értékét tárolja.

A **második ciklusban** történik a beérkező CAN üzenetek kezelése. A ciklus valójában egy megszakítási rutin, amely, ha hasznos ID-val rendelkező üzenetet olvas be, akkor azt bejegyezi egy tömbbe.

A **harmadik ciklus** nevezhető fő programciklusnak, ebben hajtja végre a szoftver a tesztelési feladatokat. A bejegyzett üzenetek feldolgozása után kiolvassa a digitális bemenetek aktuális állapotát, majd frissíti a Wterm-nek szolgáltatott adatokat. A tesztparancs végrehajtása során az adott perifériához tartozó változókat írja és olvassa, valamint bejegyezi egy tömbbe a kiküldendő CAN üzeneteket. Az utolsó három lépésben megtörténik az üzenetek kiküldése, valamint a digitális és PWM kimenetek értékeinek kiírása.



40. ábra: Processzor oldali tesztszoftver működése

Az első ciklus felel az ütemezésért. 500ms-ot vesz igénybe az összes AD csatorna kiolvasása egyesével. Ahogy ezekkel végzett, meghívja a fő programciklust. Ez azt is jelenti, a főciklusnak 500ms-on belül végeznie kell a futásával. A CAN üzenetekért felelős programrész függetlenül működik a másik két egységtől.

A programkód, amelyet készítettem, csak alkalmazásszinten különbözik a töltésvezérlő szoftvertől. Az UART interfészt megőriztem debug célra, a PC oldali tesztszoftverrel CAN-en keresztül kommunikál a program. A kommunikáció kérés-válasz jelleggel működik: a tesztrendszer kiad egy parancsot (REQ), a töltésvezérlő ezt nyugtázza vagy adatokat küld (ANS), az utasítástól függően. A következő táblázat foglalja össze az általam kidolgozott protokoll üzeneteit (3. táblázat):

Teszt	Típus	ID	Adatszó1	Adatszó2	Adatszó3	Adatszó4
CANA	REQ	0x50	x	x	x	x
	ANS	0x50	x+1	x+1	x+1	x+1
CANB	REQ	0x51	x	x	x	x
	ANS	0x51	x+1	x+1	x+1	x+1
ADC	REQ	0x100	-	-	-	-
	ANS1	0x101	CH0	CH1	CH2	CH3
	ANS2	0x102	CH4	CH5	CH6	CH7
	ANS3	0x103	CH8	CH9	CH10	CH11
	ANS4	0x104	CH12	CH13	CH14	CH15
DIO	REQ	0x200	irány	érték	0x0000	0x0000
	ANS	0x201	irány	érték	érték	0xFFFF
PWM	REQ	0x300	kivezetés	kitöltési tényező	0x0000	0x0000
	ANS	0x301	kivezetés	kitöltési tényező	0xFFFF	0xFFFF

3. táblázat: CAN üzenetek

Az üzeneteket ID alapján egy-egy adatfeldolgozó függvényhez rendeltem. Az összerendelést egy leírótömbbel valósítottam meg (Can_RX_table). A tesztszoftver a működése során a CANB interfészén várja a parancsokat, kivéve a CANA interfészt tesztelő függvény, amely a CANA interfészhez van rendelve. A következőképpen néz ki az azonosító- függvény összerendelés:

```
can_rxtx_entry Can_RX_table[]=
{
    //{Interface, ID, DLC, 0, timeout, function}
    {&ECana_Receive, MSG_CANA_TEST, 8, 0, EVABRD1_UPDATE_RATE * CAN_TIMEOUT_FACTOR,
&CANA_test},
    {&ECanb_Receive, MSG_CANB_TEST, 8, 0, EVABRD1_UPDATE_RATE * CAN_TIMEOUT_FACTOR,
&CANB_test},
    {&ECanb_Receive, MSG_AI_READ_REQ, 8, 0, EVABRD1_UPDATE_RATE *
CAN_TIMEOUT_FACTOR, &process_AI_request},
    {&ECanb_Receive, MSG_DIO_RW_REQ, 8, 0, EVABRD1_UPDATE_RATE *
CAN_TIMEOUT_FACTOR, &process_DIO_request},
    {&ECanb_Receive, MSG_PWM_CTRL_REQ, 8, 0, EVABRD1_UPDATE_RATE *
CAN_TIMEOUT_FACTOR, &process_PWM_request},
};
```

A **CAN interfészek** tesztje az adatszavak értékét növeli meg, majd visszaküldi az üzenetet megegyező ID-val. Az alábbi kódrészlet a CANA tesztet mutatja:

```
//CANA test
void CANA_test( struct MBOX *input ){
    if( NULL != input ){
        CANA_TEST_data->word.w0 = input->MDL.word.LOW_WORD+1;
        CANA_TEST_data->word.w1 = input->MDL.word.LOW_WORD+1;
        CANA_TEST_data->word.w2 = input->MDL.word.LOW_WORD+1;
        CANA_TEST_data->word.w3 = input->MDL.word.LOW_WORD+1;
        ECana_Send(&CANA_TEST_msg);
    }
}
```

Az **ADC** tesztekhez rendelt üzenet tartalma nem rendelkezik funkcióval, de az azonosítója jelzi a processzor számára, hogy az analóg bemenetek értékeit kéri a tesztrendszer. A töltésvezérlő 16 bites AD átalakítókkal rendelkezik, így a 16 csatornája négy üzenetben fér el. Mivel minden üzenet különböző csatorna értékeit tartalmazza, ezért azok azonosító alapján is meg vannak különböztetve. Az ADC aktuális értékeit egy tömb tartalmazza. A tömbben nyers adatokat tárol a tesztszoftver, a feszültségérték kiszámítását a PC-oldali szoftver végzi.

A töltésvezérlő **19 digitális be- és 16 digitális** kimenettel rendelkezik. Az IO tesztek során a kérés első adatszava határozza meg, hogy be- vagy kimenetre vonatkozik az üzenet. A szoftver csak kimenet esetén veszi figyelembe a második adatmezőben található értéket. Az „érték” minden egyes bitjéhez egy-egy kimeneti vonal van rendelve, minden egyes bit értéke a kimenet logikai szintjét határozza meg (0: alacsony, 1: magas). A válaszüzenetben kimenet esetén az irány és az érték változatlan marad, a maradék mezőket konstansokkal, hexadecimális FF értékkel tölti fel a szoftver. A bemenetek egy 19 bites számon ábrázolhatók a kimenethez hasonló módon, mivel egy adatmező 16 bit hosszúságú, ezért kettőt is elfoglal a válaszüzenetben. A digital_in tömb tartalmazza az aktuális értékeit a bemeneteknek, a digital_out tömbben pedig a kimenetre legutóbb kiírt értékek szerepelnek. Az alábbiak a kezelőfüggvények:

```
void send_DI_msg() {

    DIO_TEST_data->word.w0 = 0x00;
    DIO_TEST_data->word.w1 = digital_in[0];
    DIO_TEST_data->word.w2 = digital_in[1];
    DIO_TEST_data ->word.w3 = 0xFF;

    ECanb_Send(&DIO_TEST_msg);
}

void set_DO_pin (Uint16 value){

    digital_out[0] = value;
    DIO_TEST_data ->word.w0 = 0x01;
    DIO_TEST_data ->word.w1 = value;
    DIO_TEST_data ->word.w2 = 0xFF;
    DIO_TEST_data ->word.w3 = 0xFF;

    ECanb_Send(&DIO_TEST_msg);
}
```

A **PWM** tesztekben a kitöltési tényező változtatására van lehetőség. A töltésvezérlő két PWM kimenete 1kHz frekvencián működik a töltőoszlopban, ezt nem tettem változtathatóvá a tesztszoftverben. Az első adatmező adja meg, hogy melyik

csatorna módosítását kéri a tesztrendszer, a második pedig a kívánt kitöltési tényezőt határozza meg. A kezelőfüggvényt a következőképpen valósítottam meg:

```
void process_PWM_request( struct MBOX *input ){
    Uint16 pin = input->MDL.word.LOW_WORD;
    Uint16 duty_cycle = input->MDL.word.HI_WORD;
    //values should be in percentage
    if ( duty_cycle > 99 ) {
        duty_cycle = 99;
    }
    else if ( duty_cycle < 1 ) {
        duty_cycle = 1;
    }
    switch (pin){
        case 0: //DC: 0
            duty_cycle_percentage[1] = duty_cycle;
            break;
        case 1: //AC: 1
            duty_cycle_percentage[0] = duty_cycle;
            break;
        default: //AC: 1
            duty_cycle_percentage[1] = duty_cycle;
    }
    PWM_TEST_data->word.w0 = pin;
    PWM_TEST_data->word.w1 = duty_cycle;
    PWM_TEST_data->word.w2 = 0xFF;
    PWM_TEST_data->word.w3 = 0xFF;

    ECanb_Send(&PWM_TEST_msg);
}
```

A perifériák írására vagy olvasására lehetőség van a Wterm alkalmazás segítségével is, amennyiben ez szükséges, de a tesztrendszer szempontjából nem ez a preferált interfész, hanem a CAN-es vezérlés.

6.2 PC-oldali tesztszoftver

A PC-oldali tesztszoftver a 3.2.1-es fejezetben bemutatott TestStand tesztelési keretrendszerben fut. A szoftver szekvencia hívásokból áll. A szekvenciákban, legyen az tesztelési terv, tesztkészlet vagy teszteset, szükség volt a megfelelő működéshez a TestStand programozási környezet által nyújtott függvényekre is. A szoftverben számos felugró ablak segíti az operátor munkáját, például ezek mondják meg hova kell helyezni az oszcilloszkóp mérőfejét, vagy visszajelzést adnak, hogy sikeres volt-e az adott egység vizsgálata stb. Szinkronizálást (pl.: késleltetés), vezérlést (pl.: for, while ciklus, if-else szerkezetek) elősegítő függvényeket is alkalmaztam. Minden függvényhívás egy-egy lépésként (step) jelenik meg a programban. A lépések paraméterezhetők, számos beállítás érhető el, például előfeltétel, több szálon futó tesztek esetén szálak közti szinkronizáció, naplózási beállítások stb.

A főszekvencia a következő lépésekből áll:

1. Inicializálás,
2. Tápegység tesztek,
3. Tesztszoftver letöltés,
4. Funkcionális tesztek,
5. Végleges szoftver letöltés és validáció,
6. Teszt lezárása, kiértékelése.

Minden egyes lépés további alszekvenciákra bontható. A következő fejezetek ezeket fogják bemutatni. Ha az inicializálás során valamelyik lépés hibára fut (pl.: nem elérhető a VirtualBench), a szoftver azonnal a teszt lezárására ugrik, egyébként a tesztek végigfutnak és a TestStand által készített jegyzőkönyv alapján meghatározható az esetleges hiba.

6.2.1 Inicializálás

A lépés magába foglalja a **hardver eszközök beállítását** és a **limitértékek betöltését** egy szöveges fájlból. A TestStand lehetőséget ad minden egyes limit importálására vagy exportálására, így egy fájlon keresztül lehet módosítani az összes elfogadási kritériumot, nem szükséges kikeresni az adott tesztet a szekvencián belül. A hardverkonfiguráció szintén további lépésekre bontható. Egy szöveges fájlban tárolom, hogy milyen hardverelemek állnak rendelkezésre a tesztrendszerben, ezt tölti be először a rendszer. A fájl a hardver absztrakciós réteg beállításait adja meg. A következő lépésben az operátornak lehetősége van lefuttatni egy **oszcilloszkóp kalibrációt** a digitális multiméterhez viszonyítva (az opciót a TestStand minden indulásakor felajánlja, de folyamatos tesztek során nem), amelyet azért kellett megvalósítanom, mert a feszültségmérések pontatlanok voltak.

A **kalibráció** a következő lépésekből áll (az első lépés kivételével mindegyiket a szoftver végzi):

1. Egy-egy digitális multiméter és oszcilloszkóp csatorna tápvonalra kötése (a tápvonalnak nem szabad bekötve lennie a tesztrendszerbe),
2. Kalibráció (ciklikusan az oszcilloszkóp összes mérési tartományára végrehajtja a szoftver):

a. Oszcilloszkóp mérési tartomány beállítása,
b. A tápfeszültség beállítása az oszcilloszkóp aktuális mérési tartományának 10%-ra (ha ez például 1V, akkor a táp 0,1V-t kapcsol a kimenetére),
c. Feszültségmérés oszcilloszkóppal és multiméterrel, a mért értékeket nevezzük x1 illetve y1-nek,
d. A tápfeszültség beállítása az oszcilloszkóp aktuális mérési tartományának 90%-ra (ha ez például 1V, akkor a táp 0,9V-t kapcsol a kimenetére),
e. Feszültségmérés oszcilloszkóppal és DMM-mel, a mért értékeket nevezzük x2 illetve y2-nek,
f. Kalibrációs paraméterek kiszámítása egyenes illesztéssel a négy mérési pont alapján,

3. Kalibrációs paraméterek ellenőrzése, majd mentése fájlba.

A következő egyenletrendszer írható fel a mért értékekre:

$$y1 = x1 * m + b$$
$$y2 = x2 * m + b$$

Ezekből levezethető a következő két összefüggés adódik, amelyek megadják a kalibrációs paramétereket kiszámítását:

$$m = \frac{y1 - y2}{x1 - x2}$$
$$b = y1 - x1 * m$$

Ha a szoftver helyes értékeket számolt ki (m értéke 0,5 és 1,5 között van és nem NaN (Not a Number), b értéke nem NaN), akkor a paramétereket a mérési tartományhoz párosítva egy szöveges fájlban tárolja. Amennyiben ez nem teljesül, akkor az operátornak lehetősége van újraindítani a kalibrációs folyamatot. Ha fájlba mentés megtörtént, akkor a paramétereket a tesztszoftver betölti a tesztállomás változók közé. Amelyik tesztesetben **feszültségmérés** történik, a tesztszoftver a mérési tartomány függvényében korrigálja a mért értéket a kalibrációs értékekkel.

A kalibrációs paraméterek betöltését követően a **program konfigurálja a hardver eszközöket**, mint például a tápegységeket, CAN interfészt, digitális IO-kat, analóg kimenetet stb. Az eszközök „befoglalása” is ezzel a lépéssel valósul meg, amely következtében csak a tesztszoftver tudja ezeket kezelni mindaddig, amíg be nem fejeződik a tesztek futtatása, azaz nem „engedi el” őket a program.

A táptesztek előtt a tesztrendszernek be kell állítania a föld rétegek összekapcsolását:

- A **töltésvezérlő** kártyán a **digitális** és **analóg** föld egy ponton (egy 0Ω-s ellenállással) össze van kötve. A töltésvezérlő digitális földjét a TENMA tápegységtől (**TEST_GND1**), az analóg földjét pedig az alaplap analóg földjétől (**AGND**) kapja.
- A töltésvezérlő **tesztkártya áramkörei** az alaplapot is tápláló VirtualBench tápvonalról működnek, tehát a földpontjuk az alaplapi digitális föld (**GND**).
- Az **NI DAQ** kártya digitális vonalaihoz tartozó föld szintén **GND**-hez kapcsolódik.
- A **tesztrendszer analóg kimenetének** (a VirtualBench 6V-os tápegységének) a földje az **AGND**-re csatlakozik.
- Amennyiben BNC- vagy banándugós csatlakozón **további analóg merőműszerek** vannak bekötve a rendszerbe, azok külön földvonallal rendelkeznek (**AIGND**).

A töltésvezérlő tesztelésének szempontjából csak a **GND – TEST_GND1** összekötését kell megtenni, ennek oka, hogy:

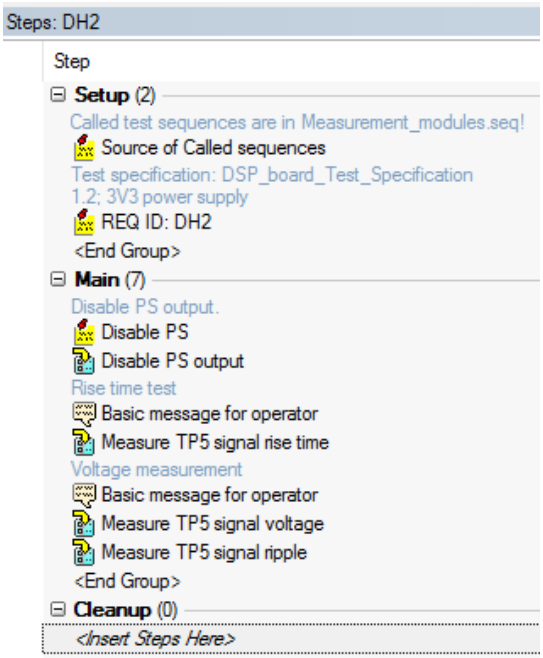
- A GND-AGND összekötés teljesül a töltésvezérlő kártyán,
- az NI DAQ kártya és a tesztkártya digitális földjei közös földre csatlakoznak,
- a VirtualBench analóg kimenetének földje a töltésvezérlő analóg kimenetével összeköttetésben áll,
- valamint a tesztek elvégzéséhez nincs szükség további analóg műszerekre, így azok nincsenek a rendszerbe csatlakoztatva.

6.2.2 Tápegység tesztek

Hat tápegység tesztlépést (szekvenciát) állítottam össze a szoftverben (DH1-6). A tesztszekvenciák programozásakor törekedtem a hatékony tesztelés megvalósítására. Odafigyeltem arra, hogy a lehető legkevesebb tápegység kapcsolásra legyen szükség a mérések során. Az első szekvenciában implementáltam a bemenő tápvonal feszültségének ellenőrzését, valamint a töltésvezérlő kártya áramfelvételének mérését. A másik öt lépés a táp- és referenciavonalak vizsgálatát (felfutási idő, feszültségszint, zajosság) végzi. Ugyanazzal a tesztkészlettel dolgoznak, csak a paraméterezésben és az operátornak adott utasításban különböznek.

A tesztkészlet a következő lépésekből áll össze (a TestStand-ben a szekvencia felépítését a 41. ábra mutatja):

1. Tesztelt eszközt áramellátását végző tápegység lekapcsolása (Disable PS output),
2. Felfutási idő mérés (Measure TP5 signal rise time) a TP5-ös mérőponton,
3. Feszültségmérés (Measure TP5 signal voltage),
4. Zajosság mérés (Measure TP5 signal ripple).



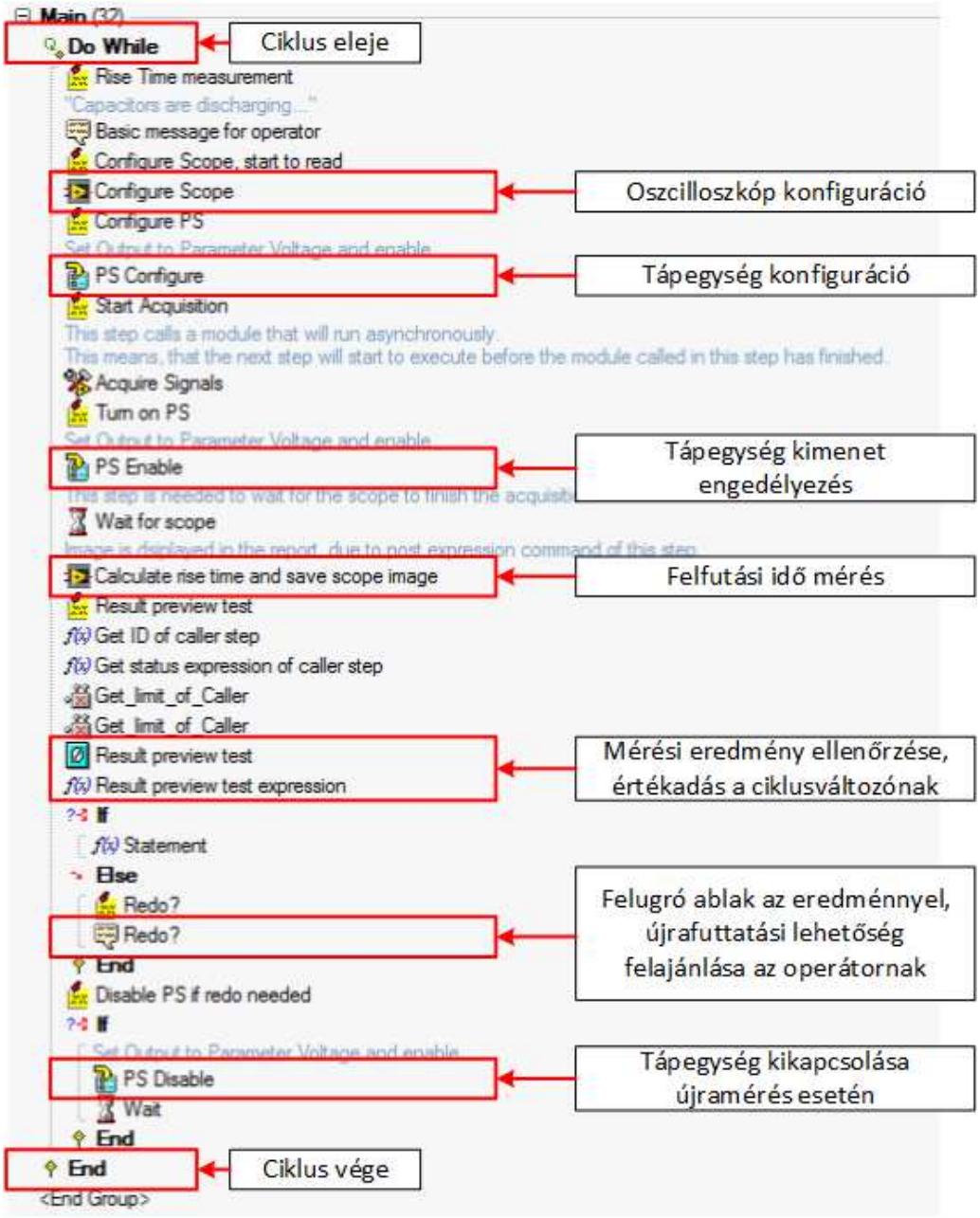
41. ábra: DH2 tápteszt (TestStand kódrészlet)

Minden tesztlépés a működéséhez elvárt módon, a tesztek végzése előtt beállítja a tápegységet, amely a tesztelt eszközt táplálja. A felfutásokat bekapcsolás során érdemes

mérni, ezért kell az első lépést minden esetben meghívni (Disable PS output). A mérések teszteset hívások.

Egy felfutási idő mérés tesztesete a következő lépésekből áll (42. ábra):

1. Tápegység lekapcsolása (első futás esetén ezt a lépést külön meg kell tenni a teszteset meghívása előtt, ahogyan a 41. ábra is mutatja)
2. Várakozás a tápvonalon található kondenzátorok kisülésére (erre azért van szükség, mert befolyásolhatják a mért időt),
3. Oszcilloszkóp konfigurálás (mindkét csatorna mintavételezése, mintavételi frekvencia, DC csatolás, mérési tartomány beállítása),
4. Tápegység bekapcsolása,
5. Felfutási idő mérése és oszcilloszkóp kép mentése a mért jelalakokról,
6. Visszajelzés az operátornak a mérés eredményéről, ha az nem felelt meg az elfogadási kritériumnak. Ekkor az operátor eldöntheti, hogy újrakezdi az első lépéstől (tápegység kikapcsolása) a mérést, vagy elfogadja a hibás eredményt és tovább lép a következő mérésre.

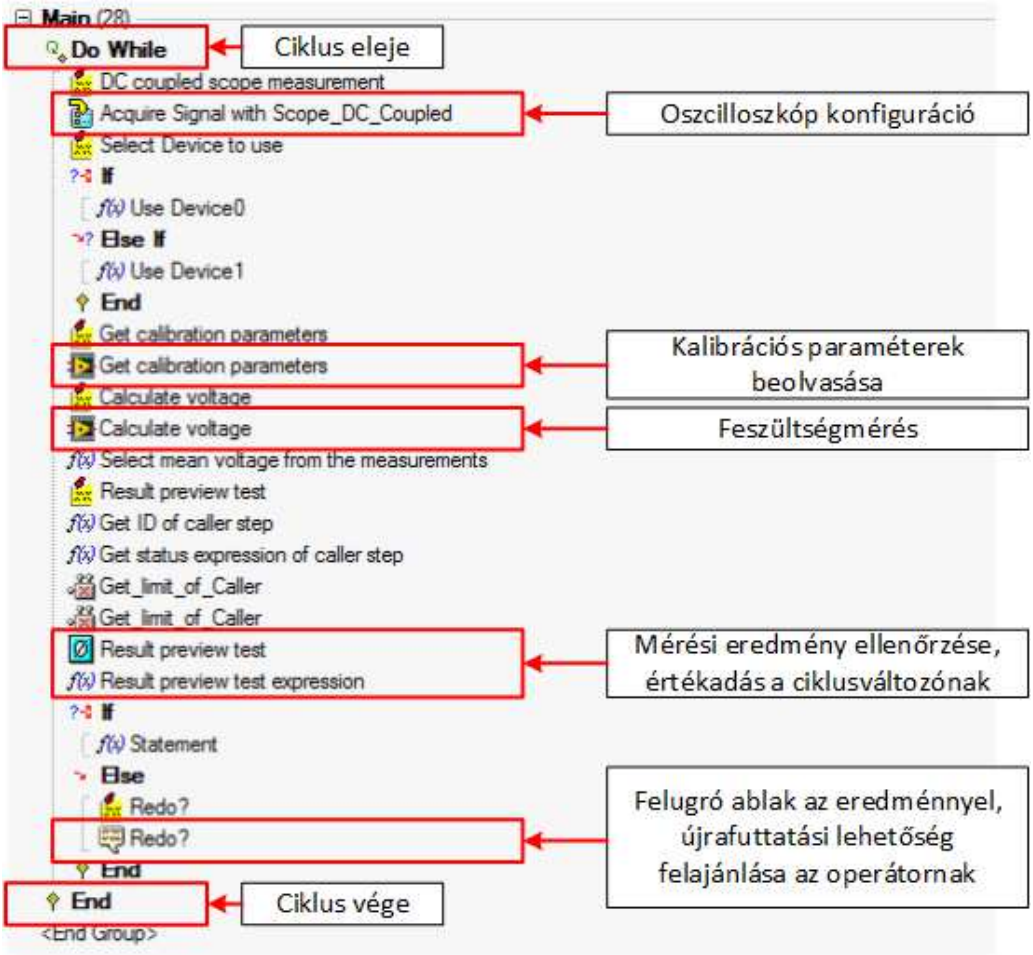


42. ábra: Felfutási idő mérés (TestStand kódrészlet)

A mérés eredményétől függetlenül a szoftver végrehajtja kép készítést. Ez hiba esetén a hibaok megkeresését segíti elő. A lépésekből látható (42. ábra), hogy a tápegység nem kerül kikapcsolásra a teszteset végeztével, ezért a feszültségmérésnél ezzel már nem kell foglalkozni. A feszültségmérés a következő lépésekből áll (43. ábra):

1. Oszcilloszkóp konfigurálás (egy csatorna mintavételezése, mintavételezési idő, DC csatolás, mérési tartomány beállítása),
2. Kalibrációs paraméterek betöltése a mérési tartomány alapján,

3. Mérés, a mért értékek átlagolása, majd a kalibrációs paraméterekkel korrigált feszültségérték kiszámítása,
4. Visszajelzés az operátornak a mérés eredményéről, ha az nem felelt meg az elfogadási kritériumnak. Ekkor az operátor eldöntheti, hogy újrakezdi az első lépéstől (oszcilloszkóp konfiguráció) a mérést vagy elfogadja a hibás eredményt és tovább lép a következő mérésre.

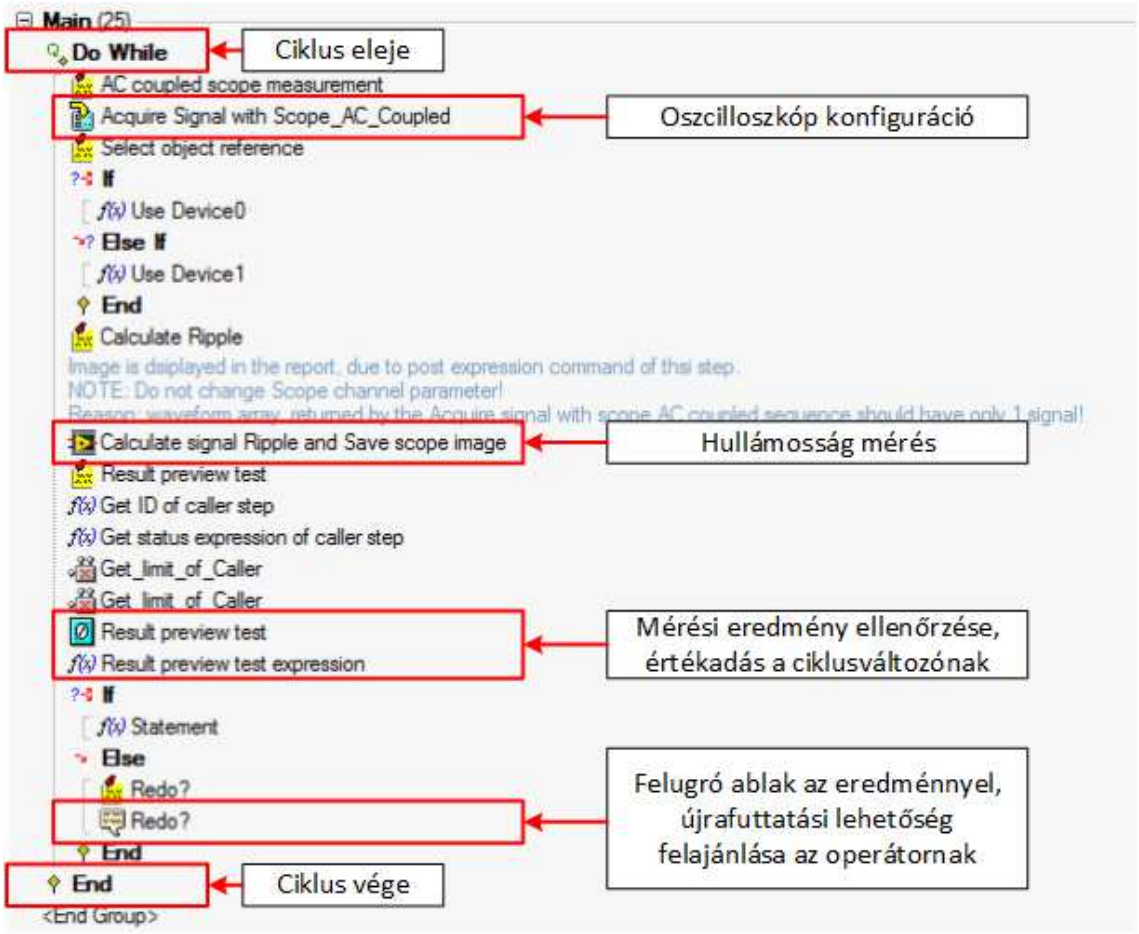


43. ábra: Feszültségmérés (TestStand kódrészlet)

A zajosság mérése hasonlóan alakul a feszültségméréshez (44. ábra):

1. Oszcilloszkóp konfigurálás (egy csatorna mintavételezése, mintavételi frekvencia, AC csatolás, mérési tartomány beállítása),
2. Oszcilloszkópkép készítése a jelről,
3. Mért értékekből a legnagyobb csúcstól-csúcsig tartó érték kiszámítása,
4. Visszajelzés az operátornak a mérés eredményéről, ha az nem felelt meg az elfogadási kritériumnak. Ekkor az operátor eldöntheti, hogy újrakezdi

az első lépéstől (oszcilloszkóp konfiguráció) a mérést vagy elfogadja a hibás eredményt és tovább lép a következő mérésre.



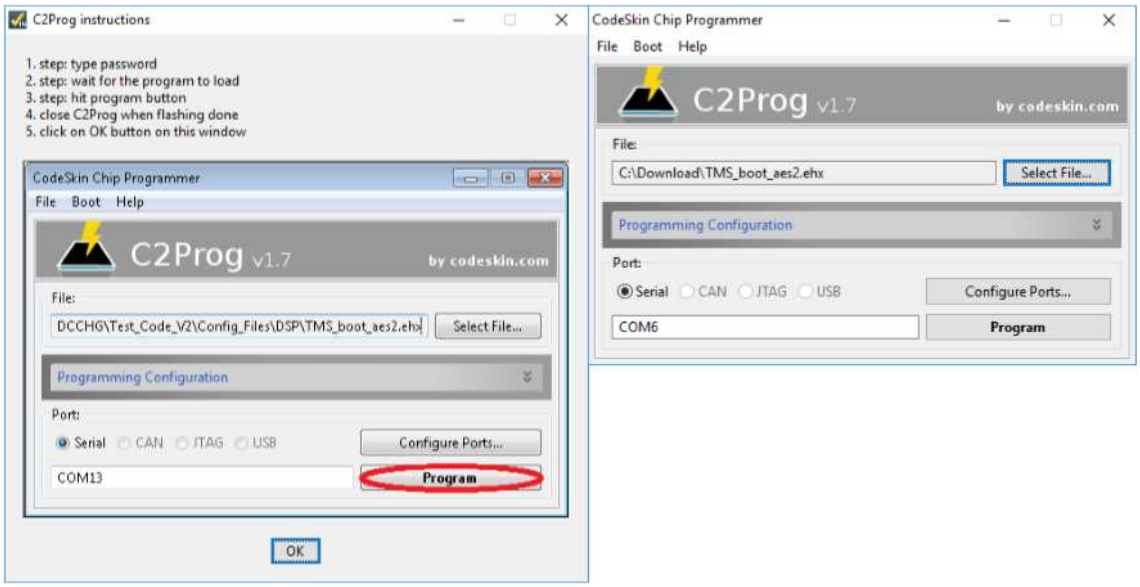
44. ábra: Zajosság mérés (TestStand kódrészlet)

6.2.3 Tesztszoftver letöltés

A töltésvezérlő kártyán levő hardverelemeket a funkcionális tesztek előtt fel kell programozni. Először az FTDI chip programozását kell elvégezni FT Prog-gal, majd a C2Prog segítségével bootloader-t kell letölteni a belső FLASH memóriába. Végül ezek után lehet a Wterm alkalmazással a külső FLASH-be a tesztszoftvert is letölteni. A lépések sorrendje nem változtatható meg.

Az FT Prog [10] nem rendelkezik LabVIEW-s API-val, de parancssorból is kezelhető. A TestStand tetszőleges parancssori hívást végre tud hajtani, így az **FTDI chip programozását** közvetlenül a tesztelési környezetből oldottam meg. Az IC-t XML formátumú leírófájl letöltésével lehet konfigurálni. A gyári beállítások felülírására a következtében az operációs rendszer az FTDI chipet mindig ugyanahhoz a soros porthoz fogja hozzárendelni, amennyiben csak egy csatlakozik a PC-hez.

A töltésvezérlő működéséhez szükséges **bootloader letöltése** a C2Prog alkalmazással lehetséges. A C2Prog [11] rendelkezik parancssori- vagy LabVIEW-s interfésszel, melyek lehetővé teszik a program automatizált futtatását, de ezek licensz kötelesek. Egy ilyen licensz 495 USD áron érhető el. A projekt költségkeretének tervezése során ezt az összeget nem allokálták, ezért más módszerrel kellett megoldanom a C2Prog automatizált futtatását. A TestStand képes futtatható fájlok hívására is, tehát a C2Prog grafikus kezelői felületét el tudja indítani (45. ábra). A programmal többféle Texas Instruments processzorra, többféle interfészen (JTAG, USB, soros port) keresztül lehet szoftvert letölteni (hex kiterjesztésű fájlt vár a felhasználótól). Induláskor mindig a legutóbbi konfigurációt tölti be, de a konfigurációt el is mentheti. Ez tartalmazza a letöltendő fájlt, a processzor típusát, a processzor külső oszcillátorának frekvenciáját, valamint programozáshoz alkalmazott interfészt. A szoftverletöltéshez a TestStand elindítja a C2Prog grafikus kezelői felületét, az operátor pedig felugró ablakban kapja a meg munkautasítást (45. ábra):

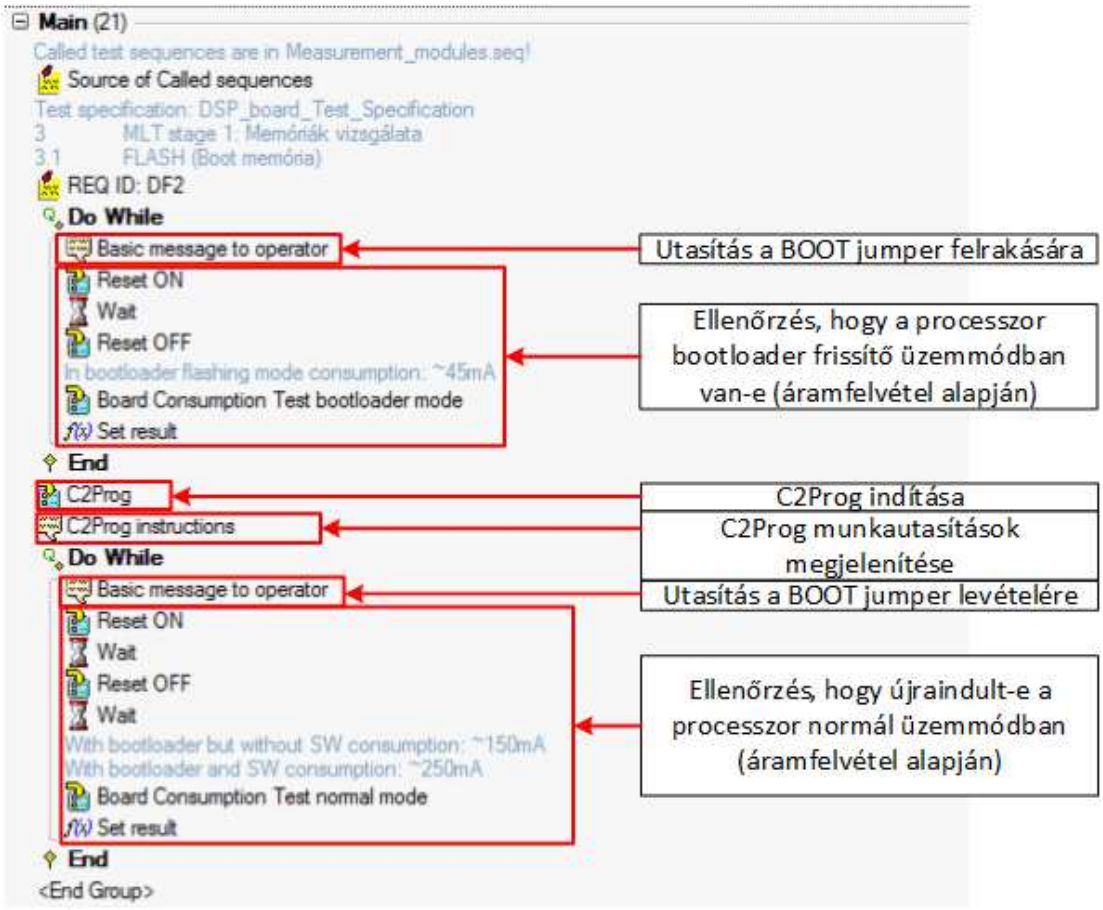


45. ábra: Munkautasítás és a C2Prog grafikus kezelői felülete

A bootloader programozása a következő lépésekből áll (46. ábra):

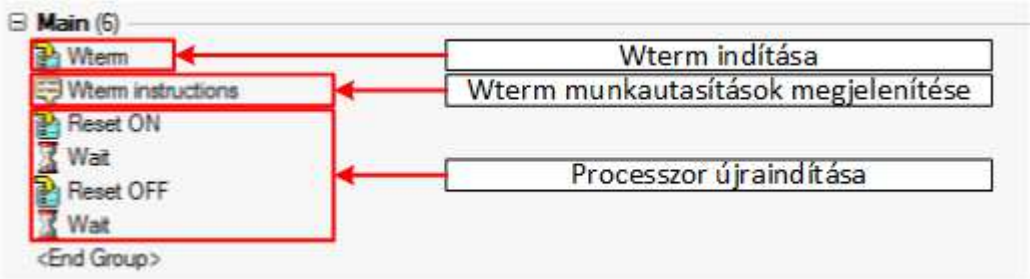
1. Az operátor utasítást kap a BOOT jumper felrakására,
2. A tesztrendszer reset állapotba viszi a processzort, majd elindítja, ekkor az bootloader frissítő üzemmódban indul el, ha ez nem teljesül, akkor visszaugrik az előző lépésre (utasítás a BOOT jumper felrakására),
3. A TestStand elindítja a C2Prog-ot,

4. Az operátor a Program gomb megnyomásával letölti a bootloader-t, majd bezárja a programot (amíg a program nem záródik be, a TestStand várakozni fog ebben a tesztlépésben),
5. Az operátor utasítást kap a BOOT jumper levételére,
6. A tesztrendszer reset állapotba viszi a processzort, majd elindítja, ekkor az normál üzemmódban indul el, ha ez nem teljesül, akkor visszaugrik az előző lépésre (utasítás a BOOT jumper levétele).



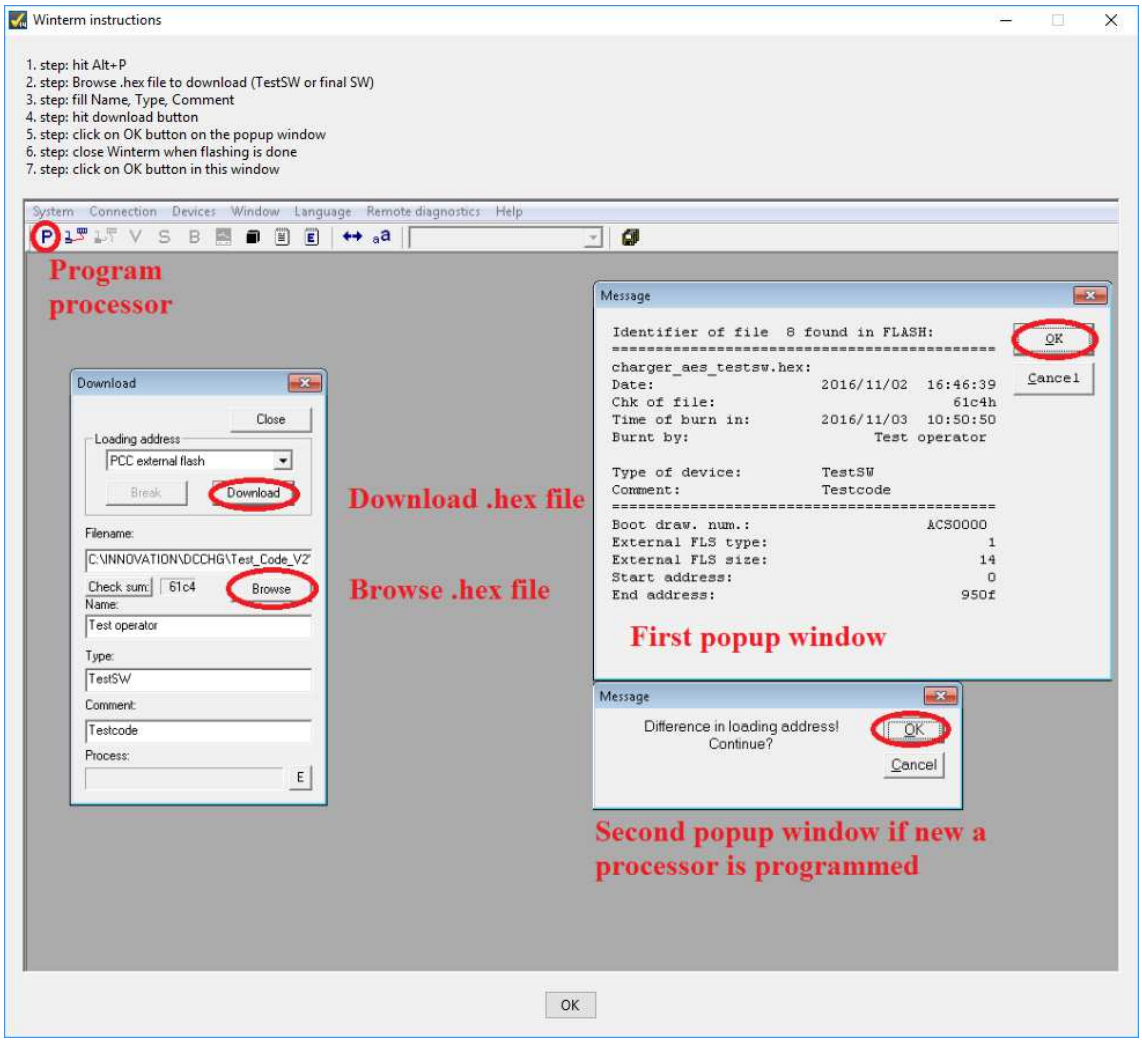
46. ábra: Bootloader letöltés (TestStand kódrészlet)

Az processzoron futó alkalmazás letöltéséhez a Wterm szükséges. Az alkalmazás nem rendelkezik se parancssori, se LabVIEW-s interfésszel, ezért a C2Prog-hoz hasonló megoldást választottam. A TestStand elindítja a programot, amelyet a szoftverletöltés után az operátor fog bezárni. Ha az operátor bezárta az utasításablakot, akkor tesztszoftver újraindítja a processzort.



47. ábra: Processzor programozása Wterm segítségével (TestStand kódrészlet)

Az operátornak csak a szoftvert tartalmazó hex fájlt kell betallóznia (48. ábra, Browse gomb), majd meg kell nyomnia a letöltés (Download) gombot. A Wterm a sikeres programozás után automatikusan újraindítja a processzort, amelyen ekkor már a tesztszoftver fog futni.



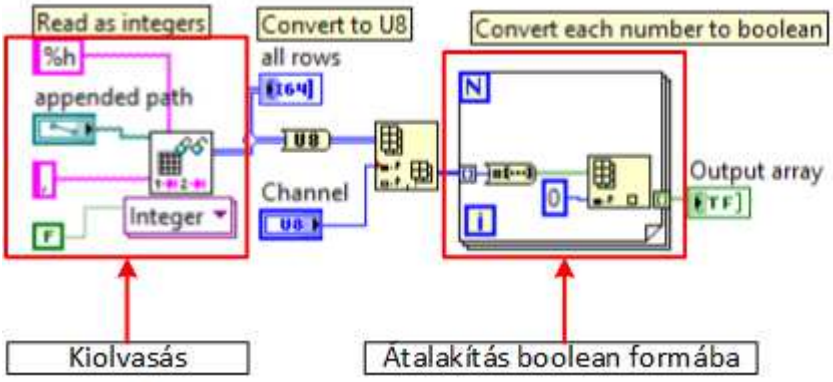
48. ábra: Wterm munkautasításutasítások

6.2.4 Funkcionális tesztek

A funkcionális tesztek során az ADC, CAN, GPIO és PWM vonalak vizsgálatát végzi el a tesztrendszer. Ezek közül először a CAN tesztelésének kell lefutnia, a többi ellenőrzés sorrendje tetszőleges. Azért szükséges a CAN tesztet elsőként elvégezni, mert a többi teszt során ezen az interfészen kommunikál a tesztrendszer a processzorral.

A **CAN tesztelés** biztosítja azt, hogy nincs ültetési hiba a CAN RX-TX kivezetéseken, valamint visszajelzést ad arról, hogy a tesztszoftver fut-e a processzoron. Ahogy a 6.1-es fejezetben bemutattam, a processzor eggyel megnöveli a beérkező üzenet adatszavainak értékét, majd visszaküldi az üzenetet változatlan azonosítóval. A teszt mindkét interfészen elvégzésre kerül.

A töltésvezérlő teszthardver könnyebb huzalozásának érdekében nem sorba kötöttem be az analóg demultiplexerre a processzor **analóg bemeneteit**, hanem törekedtem a legegyszerűbb kialakításra. Ebből kifolyólag például az 1-es számú AD csatorna nem az 1-es számú demultiplexer vonalra van bekötve. Azonban azért, hogy mégis sorban tudjon lefutni a teszt az egyes bemeneteken, létrehoztam egy leírótáblát. A leírótábla megadja, hogy a demultiplexer vezérlővonalainak milyen értéket kell adni az egyes AD csatornák kijelöléséhez. A leírótáblát egy szöveges fájlban tárolom, minden sor sorszáma adja meg, hogy melyik csatornához tartozik az adott konfiguráció. Készítettem egy kódmodult a kiolvasásra:

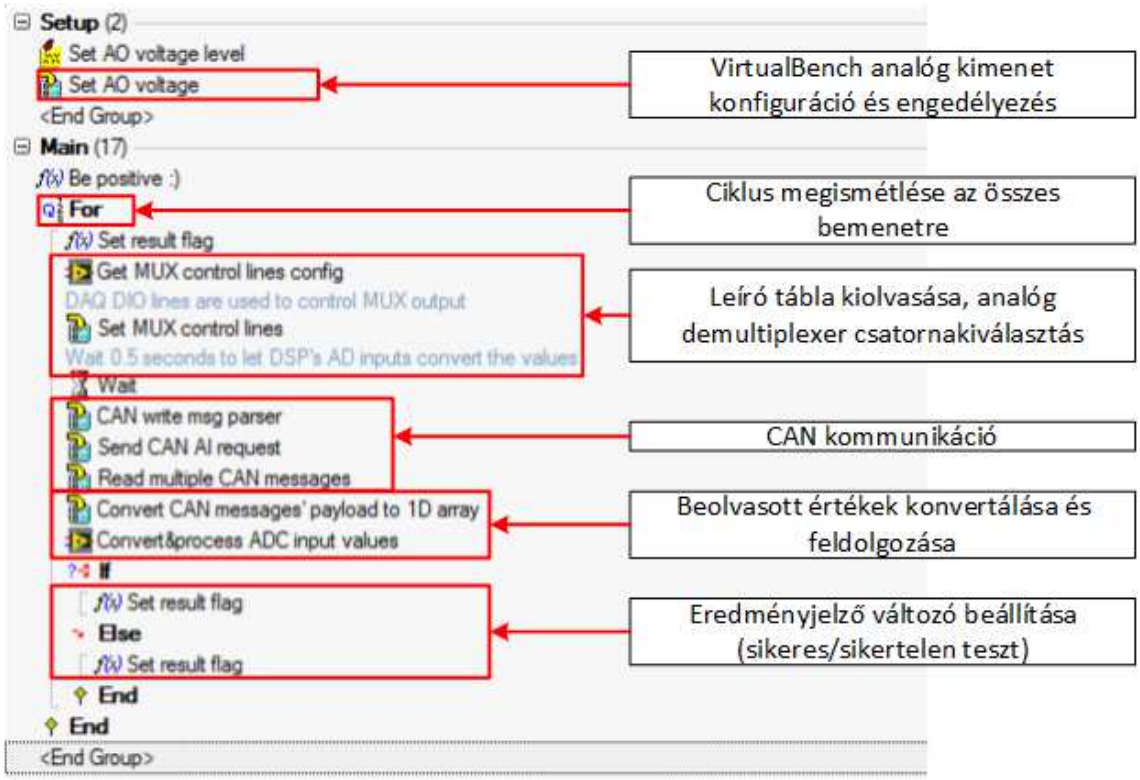


49. ábra: Leírótábla olvasása (LabVIEW kódrészlet)

A függvény kiolvassa a vesszővel elválasztott értékeket, majd Boolean formátumra alakítja (ilyen adattípussal várja a vezérlést a tesztrendszer a digitális kimenetein).

A tesztszoftver a következő lépéseket végzi el az AD átalakító tesztek során (50. ábra):

1. 0V-tól különböző feszültségérték beállítása a tesztrendszer analóg kimenetén,
2. Leírótábla kiolvasása, vezérlővonalak beállítása,
3. Az analóg bemenetek állapotainak lekérdezése CAN interfészen keresztül,
4. Lekért adatok átkonvertálása feszültségértékekre,
5. Ellenőrzés, hogy helyes értéket olvasott-e ki a processzor, valamint, hogy nincs-e rövidzár,
6. 2-5. lépés megismétlése mind a 16 bemenetre.



50. ábra: Töltésvezérlő ADC teszt (TestStand kódrészlet)

Ha hibát talált a teszt, akkor a jegyzőkönyvbe beírja a tesztszoftver, hogy mely csatornák között van rövidzár vagy mely csatornán olvasott ki rossz értéket a processzor.

Az I2C portbővítők vonalainak és a töltésvezérlő kivezetéseinek összekötésekor a huzalozás egyszerűsítésére törekedtem, tehát az analóg csatornákhoz hasonlóan nem sorban vannak bekötve a **digitális vonalak** az IC-kre. Arra viszont odafigyeltem, hogy az

egyes be- és kimeneti vonalak külön-külön IO bővítőre csatlakozzanak. Ebből adódóan ezekhez a tesztekhez is létrehoztam egy-egy szöveges fájlban tárolt leírótáblát. A fájlban egy sor sorszáma azonosít egy vonalat. Az első oszlopban található érték megadja, hogy az adott digitális vonal a DAQ kártyára vagy az IO bővítőre van kötve (ez a processzor digitális bemeneteinek vizsgálatakor számít). A második oszlop meghatározza, hogy a tesztrendszerben hányadik számú portra, a harmadik pedig, hogy melyik vonalra csatlakozik a töltésvezérlő digitális jele. A töltésvezérlő digitális bemeneteinek tesztelése esetén, az adatok alapján kiszámolja a szoftver, hogy a tesztrendszer melyik kimenetét milyen állapotba kell kapcsolnia. A processzor digitális kimeneteinek vizsgálata során a leírótáblából kiolvasott adatok alapján, a tesztszoftver kiszámolja, hogy a tesztrendszer melyik digitális bementén várja a logikai magas szintet.

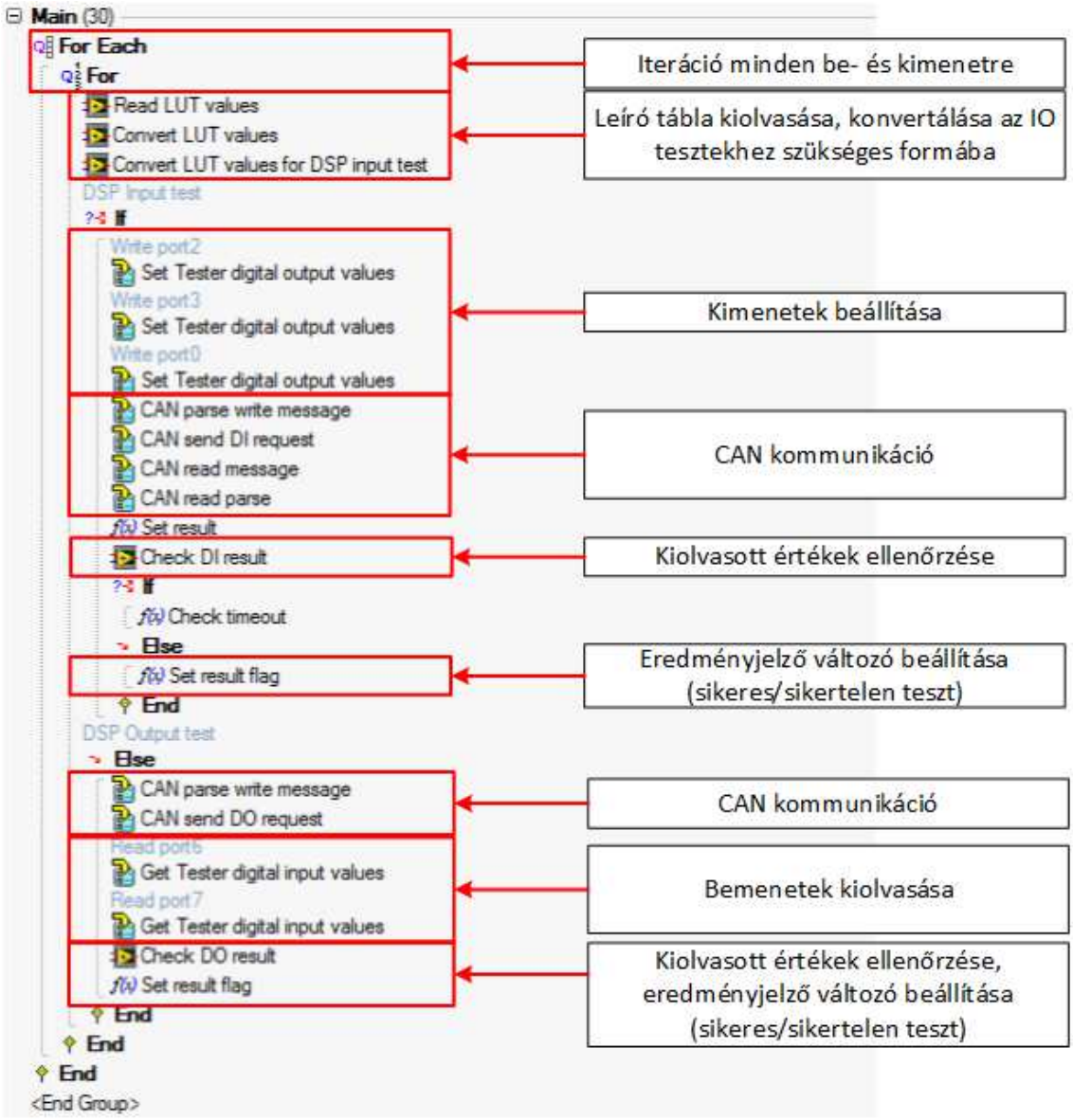
A processzor **digitális bemeneteinek** tesztje a következő lépésekből áll (51. ábra):

1. Leíró tábla kiolvasása,
2. A tesztrendszer digitális kimeneteinek beállítása: a vizsgált vonal logikai 1-re, a többi logikai 0 értékre,
3. Adatlekérés a töltésvezérlő digitális bemeneteinek értékeiről CAN interfészen keresztül,
4. Ellenőrzés, hogy a tesztrendszer által kiadott érték megegyezik-e a processzor beolvasott értékeivel,
5. 1-4. lépés megismétlése mind a 19 bemenetre.

A **digitális kimenetek** vizsgálata nagyon hasonló (51. ábra):

1. Leíró tábla kiolvasása,
2. Vezérlés küldése CAN interfészen: a vizsgált vonalat logikai 1-re, a többit logikai 0-ra állít a processzor,
3. Beállított érték visszaolvasása a tesztrendszer digitális bemenetein,
4. Ellenőrzés, hogy a beolvasott érték megegyezik-e a vezérlésben kiadott értékekkel,
5. 1-4. lépés megismétlése mind a 16 kimenetre.

Ha hibát talált a teszt, akkor a jegyzőkönyvbe belekerül, hogy mely vonalak között van rövidzár vagy mely vonalon olvasott rossz értéket a processzor.



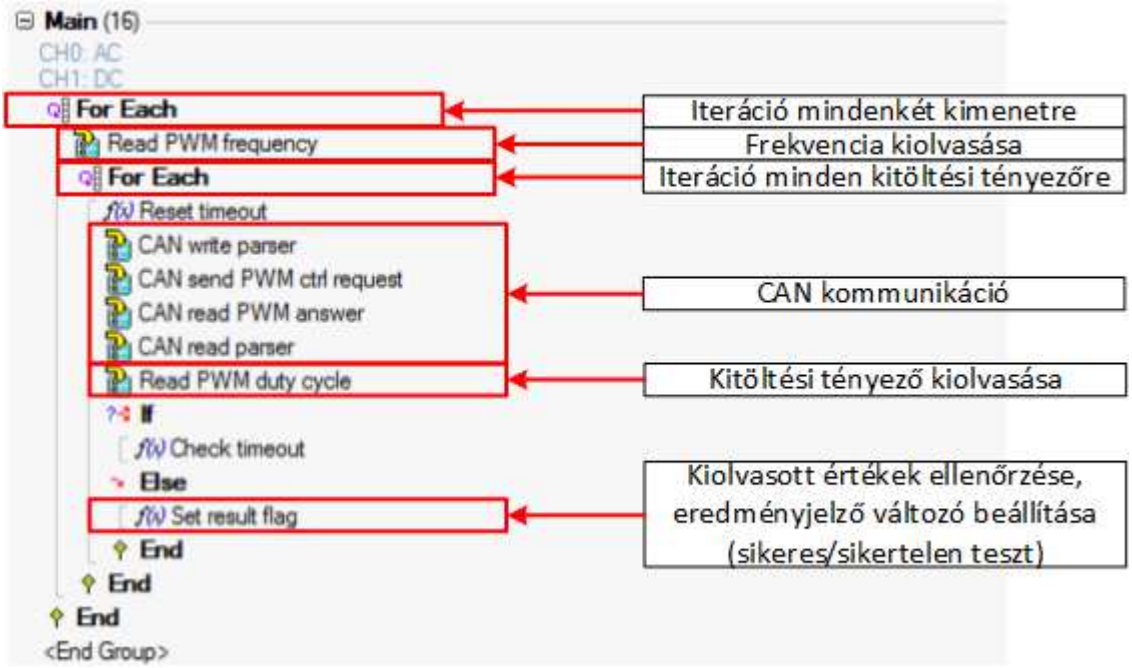
51. ábra: Töltésvezérlő digitális IO teszszekvencia (TestStand kódrészlet)

A **PWM** tesztek során a processzor állandó frekvenciájú, 1kHz-es jelet ad ki mindkét csatornáján, a kitöltési tényezőket a vezérlési utasításoknak megfelelően módosítja. A tesztrendszer 10%-tól 90%-ig minden kerek tízes (10,20,30 stb.) kitöltési tényezőt megmér (52. ábra):

1. Vezérlés küldése CAN interfészen: egyik csatorna kitöltési tényezőjének módosítása,
2. Frekvencia és kitöltési tényező mérés,

3. Ellenőrzés, hogy helyes értéket adott-e ki a processzor, valamint, hogy nincs-e rövidzár a vonalak között,
4. Kitöltési tényező növelése, ugrás az első lépésre,
5. 1-4. lépés megismétlése a másik csatornára is.

Ha hibát talált a teszt, akkor a jegyzőkönyvbe belekerül, hogy van-e rövidzár a két csatorna között vagy mely kitöltési tényezőt nem tudta helyesen kiadni az adott csatorna.



52. ábra: PWM teszt (TestStand kódrészlet)

A PWM teszt lefutásával a rendszer végzett a funkcionális teszteléssel, ha nem talált hibát a vizsgált eszközben, akkor a program tovább lép a végleges szoftver letöltésre. Más esetben a tesztek lezárása következik.

6.2.5 Végleges szoftverletöltés és validáció

A tesztszoftver programozásával megegyező módon történik a végleges alkalmazás letöltése a Wterm alkalmazás segítségével. Ha ez megtörtént, akkor a processzor a CAN1-es interfészén az oszlopvezérlőnek címzett üzeneteket kezd küldeni, a töltőoszlop állapotáról. A validáció a következő lépésekkel történik meg:

1. Az operátor a tesztrendszer 1-es CAN interfészét csatlakoztatja a töltésvezérlő CAN1-es vonalaira,

2. Várakozás egy oszlopvezérlőnek szánt üzenetre,
3. Ha az üzenet nem érkezett meg 10s-on belül, akkor a végleges szoftver letöltése nem volt sikeres.
4. Szoftverletöltés újrapróbálása,
5. 1-4. lépés ismétlése, amíg nem sikerült validálni, hogy a végleges szoftver fut a processzoron (szoftverletöltési hibák nem szoktak lenni, az újrapróbálkozás az operátori hiba kiszűrése végett került a tesztszoftverbe).

6.2.6 Tesztek lezárása

A kézi méréseket Excel táblába naplóztam, ezt viszont a TestStand közvetlenül nem képes kezelni. Ezért egy beépülő modul segítségével a mérési eredmények csv kiterjesztésű szöveges fájlba naplózza a program, amely könnyen importálható a meglévő Excel táblába.

A naplózás után a tesztszoftverben megtörténik a hardvereszközök felszabadítása, ezt követően más programok is használhatják ezeket.

7 Kézi- és félautomata mérések összehasonlítása

7.1 Hardveres tesztek

7.1.1 Mért értékek

A kézi- és félautomata mérések eredményeit ugyanarra a töltésvezérlő kártyára az 4. táblázat foglalja össze (az eredmények kiértékelését a következő fejezetben teszem meg):

Mért jellemző	Kézi mérés	Félautomata mérés
Áramfelvétel [mA]	68	256
3,3V-os tápvonal feszültség [V]	3,318	3,344
3,3V-os tápvonal felfutás [ms]	2,426	1,159
3,3V-os tápvonal zajosság [mV_{pk-pk}]	8	86,42
1,9V-os tápvonal feszültség [V]	1,908	1,94
1,9V -os tápvonal felfutás [ms]	0,76	0,552
1,9V-os tápvonal zajosság [mV_{pk-pk}]	5	90,535
3,3Vmem tápvonal feszültség [V]	3,364	3,4
3,3Vmem tápvonal felfutás [ms]	1,265	0,405
3,3Vmem tápvonal zajosság [mV_{pk-pk}]	73	152,3
3Vref tápvonal feszültség [V]	3,005	3,033
3Vref tápvonal felfutás [ms]	108	80,95
3Vref tápvonal zajosság [mV_{pk-pk}]	5	69,96
1,5Vref tápvonal feszültség [V]	1,506	1,532
1,5Vref tápvonal felfutás [ms]	104,8	88,35
1,5Vref tápvonal zajosság [mV_{pk-pk}]	4	49,38

4. táblázat: Kézi és félautomata mérések eredményei

7.1.2 Eredmények értékelése

Egy olyan kártyát teszteltem a félautomata rendszerben, amelyre a kézi mérések során már letöltöttem a szoftvert, ezért a megnövekedett **áramfelvétel** a program futását jelzi, a mérési eredmény elfogadható.

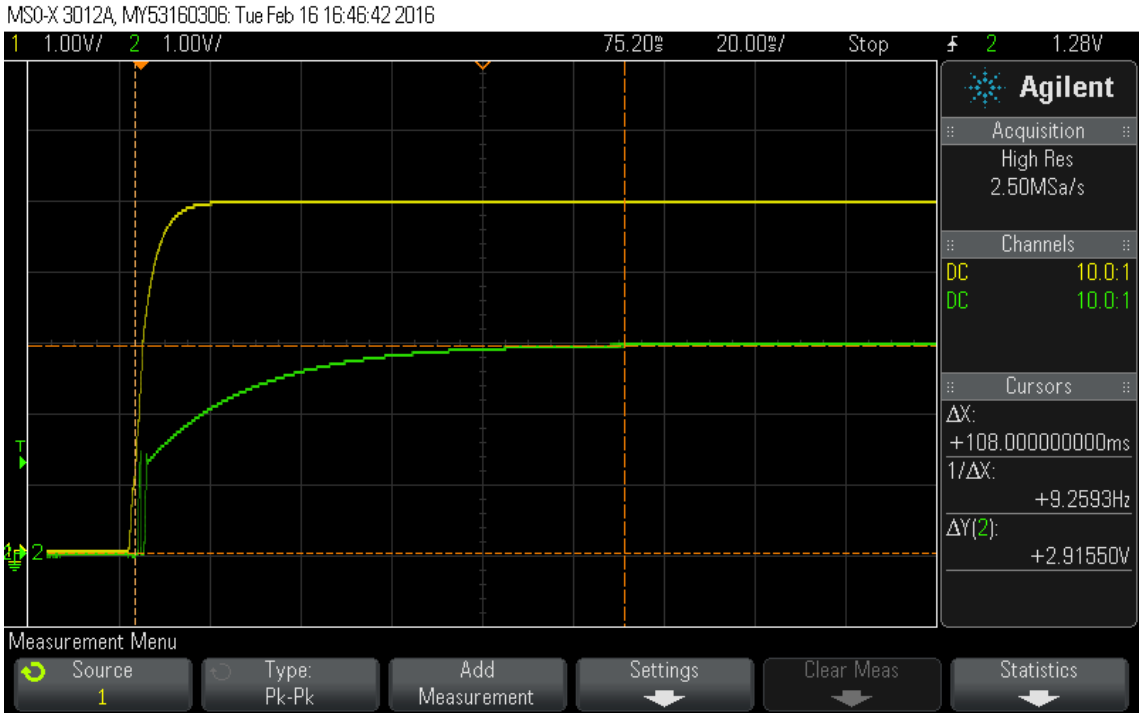
A **feszültség mérések** a táblázat alapján megfelelőek, az oszcilloszkóp pontossága az adatlapja alapján az alábbi képlet alapján számítható ki:

$$V_{hiba} = \pm 2\% * V_{mért} \pm 1\% * V_{mérési\ tartomány}$$

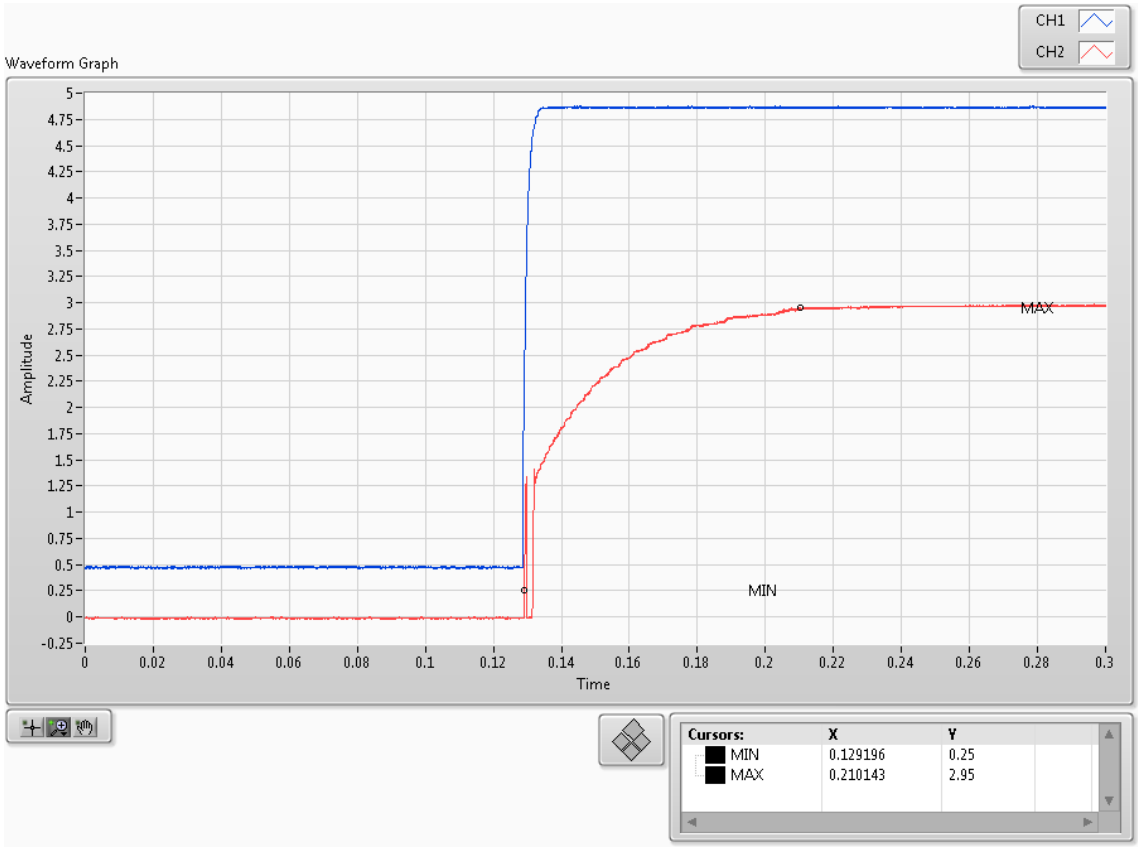
A tesztrendszerben például a VirtualBench a 3,3V-os feszültséget 4V-os mérés tartományon méri, ebből kiszámolható, hogy maximum (66+40)mV hiba léphet fel pozitív és negatív irányban egyaránt. Ezen az értéken még a kalibrációs paraméterek módosíthatnak, de ha a kézi méréseket vesszük referenciának, akkor a félautomata rendszer eredményei a hibatartományon belül tartózkodnak.

A félautomata tesztek **felfutási idő mérés**i eredményei minden esetben kisebb értéket adnak vissza. Ennek oka, hogy a kézi mérések során az oszcilloszkóp kurzoraival pontosabban lehet kijelölni az időtartományt, amelyen a felfutás megtörtént. A félautomata tesztek esetén előre megadott feszültségszintek között számolja ki a felfutás idejét. Alacsony alsó érték esetén előfordulhat, hogy túl korai időponttól kezdi el számolni az felső értékig eltelt időt, magas felső érték esetén pedig annak áll fent a veszélye, hogy túl hosszú időt fog mérni felfutásnak a szoftver. Ezek az értékek számos, nem azonos kártyán elvégzett méréssel javíthatók, a mérések során összegyűjtött statisztikai adatok alapján.

A következő két ábra mutat egy-egy példát a kézi- (53. ábra, vizsgált jel zöld színnel) és a félautomata (54. ábra, vizsgált jel piros színnel) felfutási idő mérésekre. A kézi méréssel pontosan meg tudtam adni a felfutás kezdetét és végét a kurzorok segítségével (a kurzorokat kézzel állítottam be). A félautomata esetben, a paraméterekkel megadott feszültségszintek nem tökéletes pontossággal jelölik a felfutás kezdetét és végét (kurzorok Y értékei az ábrán). A mérések alapján levonható a következtetés, hogy az általánosságban alkalmazott mérési elv, mely szerint egy jel felfutási ideje az amplitúdó 10%-tól a 90%-ig tart, teljesítésére alkalmas a tesztrendszer, de teljes felfutási idő mérése nem lehetséges.

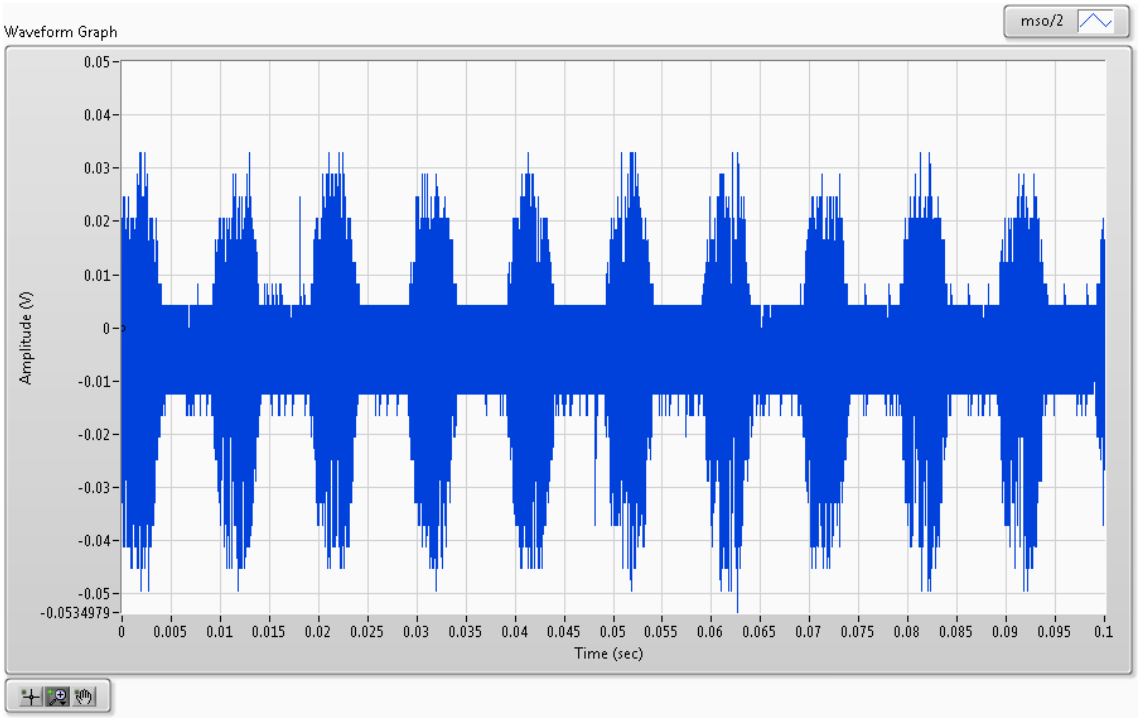


53. ábra: Kézi felfutásidő mérés (3V-os referencia)



54. ábra: Félautomata felfutásidő mérés (3V-os referencia)

A **zajosság** mérések egy nagyságrendbeli eltérést mutatnak. Ennek oka, hogy a mérésekre, a zajossághoz képest, nagy értékű külső zavar szuperponálódik a jelre a VirtualBench használata közben (55. ábra):



55. ábra: Zajosság mérés

A valód zaj az adott jelen az ábra alapján a -0,015mV-tól a 0,008mV-ig tartó sáv ($23mV_{pk-pk}$), erre „ül rá” a nagyjából 100Hz periódusú, $80mV_{pk-pk}$ értékű külső zavar. Több mérést is végeztem a zavar forrásának megtalálására:

1. Kézi méréseknél alkalmazott Agilent (Keysight) oszcilloszkóppal [31] ugyanabban a mérési összeállításban nem tapasztaltam a jelenséget,
2. Egy Tektronix oszcilloszkópos [32] mérés során szintén nem tapasztaltam a zavart,
3. Kicseréltem a VirtualBench oszcilloszkópkábelét, de a jelenség továbbra is fent állt.

A fenti tesztek alapján állítható, hogy a zavar nem a tesztrendszer áramkörein, hanem a VirtualBench-en belül szuperponálódik a zajosság mérésekre. Ennek oka lehet például az eszköz tápellátásának nem megfelelő szűrése.

A bemutatott probléma alapján megfelelően széles elfogadási kritériummal lehetséges zajosság méréseket végezni, de a tesztek után az operátornak vagy egy jelfeldolgozó algoritmusnak külön-külön ki kell értékelnie minden mérést.

7.2 Funkcionális tesztek értékelése

A funkcionális tesztek jellegre nem változtak a kézi mérésekhez képest. De az esetleges operátori hibák száma, valamint a mérések automatizáltsága miatt a tesztelésre fordított idő csökkent. Ha a funkcionális tesztek elvégzéséhez előfeltételnek tekintjük a szoftverletöltési feladatokat, akkor csak ebben a részben hibázhat a mérést végző személy, mivel a tesztek további része teljesen automata módon fut le.

7.3 Tesztelésre fordított idő

A **kézi mérések** során 13 töltésvezérlő kártyát teszteltem hardveresen és funkcionálisan, valamint programoztam fel 30 óra alatt. Egy kártyára **138 perc** esett átlagosan. Ez az idő magában foglalta az oszcilloszkóp képernyőképek mentését, a tesztek dokumentálását és kiértékelését is. A tesztek elvégzéséhez szükségem volt a töltésvezérlő működésének ismeretére.

A **félautomata** tesztrendszerrel 6 töltésvezérlőt teszteltem és programoztam fel. A tesztek **10-15 percet** vettek igénybe az egyes kártyák esetében. A szoftvert úgy írtam meg, hogy minden tesztet automatikusan kiértékelt a rendszer, valamint minden kártyához külön-külön generált egy-egy HTML jegyzőkönyvet. Ezeken kívül egy, a TestStand-be beépülő modul segítségével létrehozott egy Excel-be importáltható csv kiterjesztésű szöveges fájlt. Ez az összes elvégzett mérés eredményeit tárolja. A tesztek elvégzéséhez nincs szükség az áramkör alapos ismeretére, a tesztszoftver felugró ablakokban képekkel és szöveges üzenetekkel segíti az operátor munkáját.

A tesztrendszer fejlesztésének célja az idő- és költséghatékony tesztelés lehetőségének megteremtése a volt a sorozatgyártás számára. Ez **teljesült** azzal, hogy a tesztelésre fordított idő az **eredetihez képest 89%-kal csökkent** (138-ról 15 percre), más szóval a kézi tesztelés kilencszer annyi időt vesz igénybe, mint a félautomata. Továbbá a **teszteket szakértő bevonása nélkül is végre lehet hajtani,** mivel a tesztszoftver minden lépésről felugró ablakokban képes és szöveges utasításokat ad az operátornak.

8 Töltőoszlop emulációs környezet

A töltőoszlop emulációs környezet kialakítására még nem került sor, de tervben van, mert a teljeskörű teszteléshez szükséges. Ebben a fejezetben az erre nyújtott javaslatom fogom bemutatni.

8.1 Bevezetés

Egy összetett termék - mint például a töltőoszlop - tesztelését több szinten szükséges elvégezni [2]. Több beszállító termékéből lesz egész, ezért sokrétű vizsgálatot érdemes végezni:

1. modul szintű tesztek (hardveres, funkcionális),
2. alrendszer integrációs tesztek emulációs környezetben,
3. rendszertesztek emulációs környezetben,
4. rendszertesztek valós környezetben,
5. felhasználói tesztek.

Egyik szint kihagyása sem ajánlott. Mindegyik vizsgálat feladata az adott tesztelési összeállításban található hibák lokalizálása, tehát a magasabb integrációs szinten lévő vizsgálatok számára hibamentes, ellenőrzött termékek biztosítása. Persze így sem garantált a teljesen hibátlan működés, mert az összeköttetési pontokon új hibák jelentkezhetnek. De a javítási idő és költség kisebb lesz, mintha csak rendszertesztek lennének elvégezve, mivel könnyebb lokalizálni a hibákat.

8.2 Csatlakozó és töltési típusok

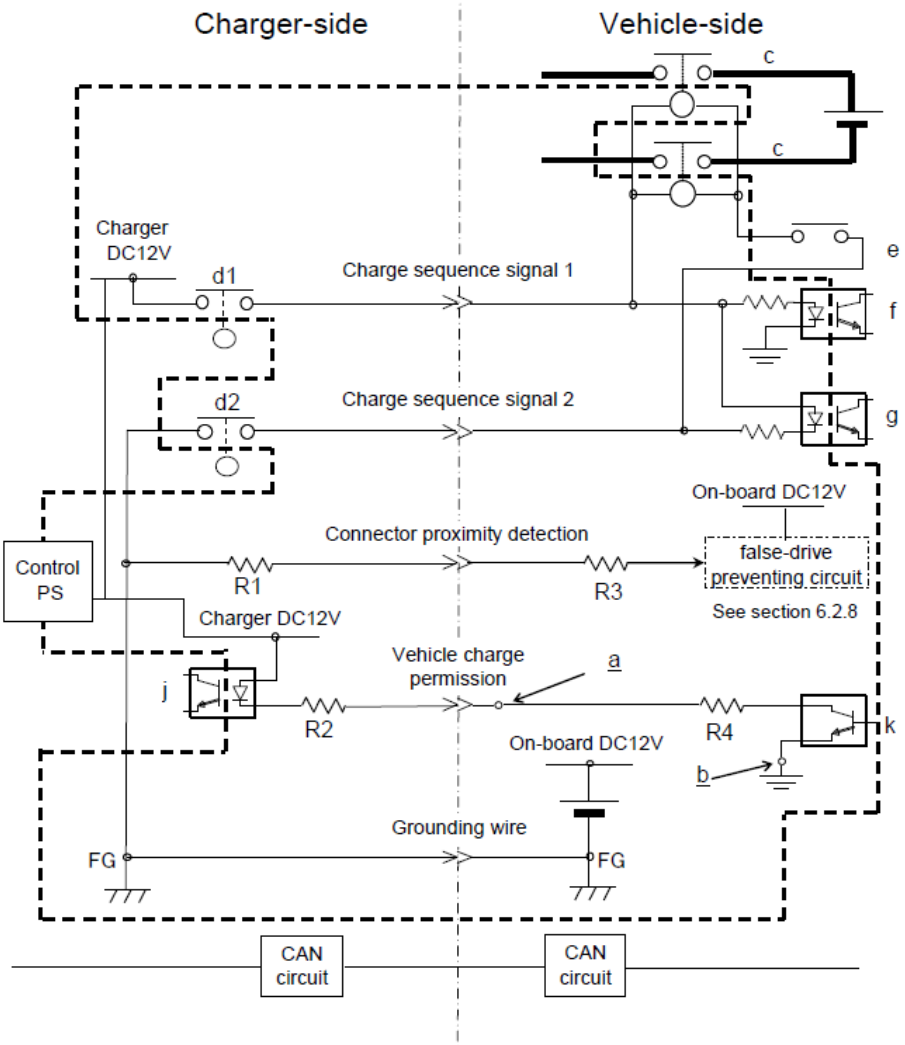
Ebben a fejezetben bemutatom milyen töltési protokollokat támogat a töltőoszlop. Az emulációs környezet kialakításához ezek ismerete elengedhetetlen.

8.2.1 CHAdeMO

A CHAdeMO [33] töltési protokollt japán ipari nagyvállalatok [34] fejlesztették ki. DC gyorstöltésre ad lehetőséget, például kis kapacitású akkumulátorokkal (15-30kWh) rendelkező járművek teljes feltöltése nagyjából 30 percet vesz igény. Célja a városi elektromos autóhasználat elterjesztése, amellyel a károsanyag kibocsátás dízel

vagy benzines autókhoz képest harmadára, hibrid járművek esetén felére csökkenthető [35].

A csatlakozó két táp- (DC + és -), négy analóg vezérlő- (Charge sequence signal 1, sequence signal 2, Connector proximity detection, Vechicle charge permission), két kommunikációs- (CAN), valamint egy földvonallal rendelkezik. Ezek láthatók a töltőoszlop és elektromos jármű kapcsolatát mutató blokkvázlaton (56. ábra):



56. ábra: CHAdeMO töltőoszlop és elektromos jármű kapcsolata [36]

A töltési folyamat három részre bontható: előkészítés, töltés és lezárás. Az előkészítési folyamatban a töltőoszlop és a jármű „megegyezik” a maximális töltési feszültségről és áramról, valamint egy biztonsági tesztet is elvégez a töltőoszlop a

balesetmentes működés érdekében. A folyamat lépései a következők (az értelmezésben az 56. ábra nyújt segítséget):

1. A töltőoszlop meghúzza a *d1*-es relét, amely következtében 12V kerül a 2-es vonalra.
2. A jármű az *f* fotokapcsolón keresztül fogadja a jelzést (értelmezése: töltés előkészítése megkezdődött), amelyre válaszul CAN üzenetben meghatározza az oszlop számára a paramétereit: töltőáramkörének feszültséglimitjét, maximális áramfelvételét, valamint az akkumulátorának kapacitását.
3. A töltőállomás miután megerősítette, hogy képes tölteni a csatlakoztatott gépjárművet, CAN buszon továbbítja a saját adatait (maximális kiadott feszültség, leadott áram).
4. A jármű ellenőrzi a kompatibilitást a fogadott adatok alapján, ha nem talált problémát, a *k* tranzisztor kinyitásával engedélyező jelet küld a töltőoszlopnak.
5. Ha az engedélyező üzenet megérkezett, abban az esetben rövid időre a töltőállomás feszültséget kapcsol a kimenetére és elvégez egy tesztet, amellyel az abnormális működés (például föld- vagy rövidzárlat) kiszűrhető.
6. Ha a teszt során a töltő nem talált hibát, akkor a *d2*-es relé meghúzásával jelzi a jármű felé, hogy készen áll a töltésre.

A töltési szakaszban mindkét fél folyamatosan ellenőrzi a működés helyességét. A töltés lépései:

1. A jármű bezárja a kapcsolót (EV contactor) a töltőáramkörének bemenetén, majd 100ms-os időközönként kiszámolja az áramot, amellyel tölthető az akkumulátora és elküldi azt az állomásnak.
2. A töltőoszlop a CAN üzenetben megadott konstans áramértéket biztosítja a járműnek.
3. Ha a monitorozás során hibás működést tapasztal a jármű, akkor a következő módszerekkel állíthatja meg a töltési folyamatot:
    a. nulla áramértéket ad meg a CAN üzenetekben,
    b. hibajelzést ad a kommunikációs buszra,

c. kikapcsolja a *k* tranzisztort, amely *NO CHARGE* analóg jelzést küld a töltőoszlop felé,

d. megszakítja az *EV contactor* kapcsolót a töltőáramkör bemenetén.

4. A töltőoszlop folyamatosan felügyeli az áram-, feszültség- és hőmérsékletértékeket az összes áramköri egységében, ha a meghatározott limiteket meghaladja bármelyik érték, akkor leállítja a töltést és hibajelzést küld a járműnek CAN buszon.

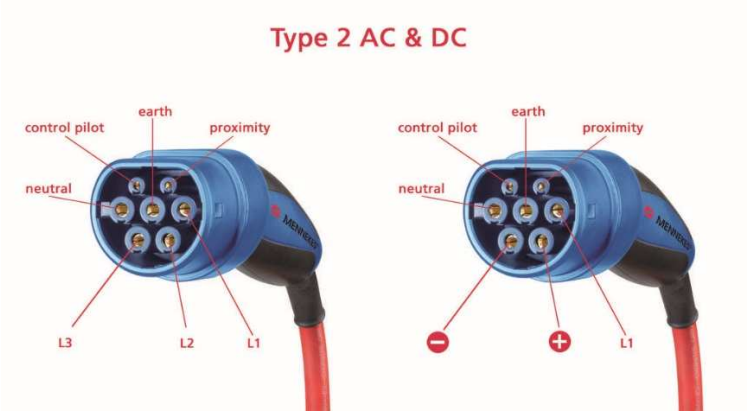
Az analóg vezérlővonalak használatát azért részesíti előnyben a protokoll, mert gyorsabban képes jelezni a problémát, mint digitális kommunikációval, így hamarabb le tud állni a töltési folyamat.

A töltési folyamat lezárása normál működés során a következőképpen alakul:

1. A jármű CAN buszon kiadja a nulla áramérték jelzést, amelynek hatására a töltőoszlop leállítja a töltést.
2. Ha a jármű megerősítette, hogy a bemenetein nem folyik már áram, akkor kinyitja az *EV contactor*-t és a *k* tranzisztor bezárásával töltést tiltó jelzést ad ki az oszlop felé.
3. A töltőállomás megerősíti, hogy nulla értékű az kimenetén az áram, majd a *d1* és *d2* relé elengedésével lezárja a töltési folyamatot.

8.2.2 AC

A MENNEKES GmbH [38] által fejlesztett Type 2 csatlakozó egyfázisú és háromfázisú AC vagy DC töltésre ad lehetőséget (a csatlakozó vezetékkiosztását az 57. ábra mutatja). Az evopro Kft. töltőoszlopában az AC típusú csatlakozó került beépítésre.



57. ábra: Type 2 csatlakozó [37]

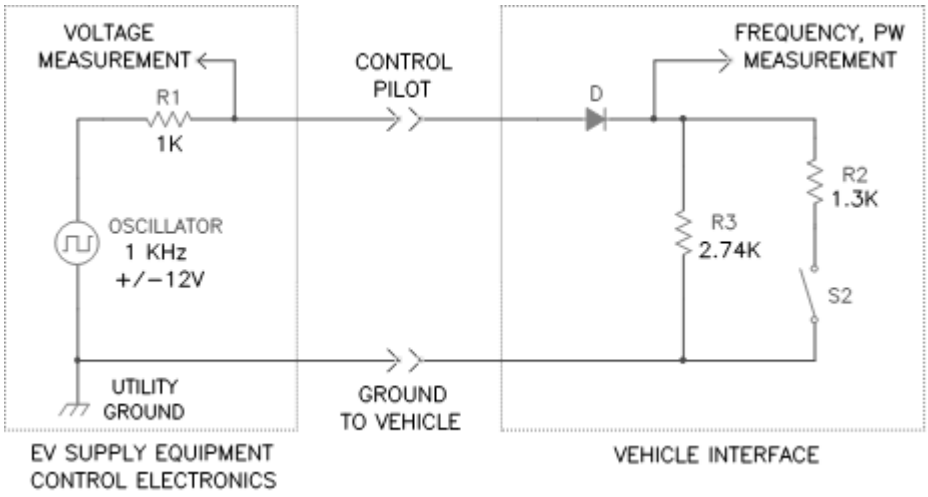
A Control Pilot (CP) egy maximum +/-12V-os PWM jel, amelynek feszültségszintjei a töltés állapotát határozzák meg. A feszültségszinteket az elektromos jármű állítja be, ellenállások kapcsolásának segítségével. A vezérlőjel +/-0,5%-os eltérést tolerál. A töltőoszlop által kiadott PWM jelek fel- és lefutási ideje nem lépheti át a 2µs-ot. A CP jel kitöltési tényezőjét a töltőoszlop képes állítani, annak függvényében, hogy mekkora az aktuális töltőárama:

- 6 és 51A között: $Kitöltési\ tényező = \frac{Áramérték}{0,6}$, például 6A-es töltőáram 10%-os kitöltési tényezőnek felel meg,
- 51 és 80A között: $Kitöltési\ tényező = \left(\frac{Áramérték}{2,5}\right) + 64$, például 75A-es töltőáram pedig 94%-os kitöltési tényezőnek felel meg.

A Proximity vonalon mérhető feszültséggel lehet meghatározni, hogy csatlakoztatva van-e az elektromos jármű a töltőoszlophoz. A különböző feszültségeket ellenállásosztással a gépjármű állítja elő, a mérhető értékek és jelentéseik a következők:

- 4,5V: a töltőkábel vége szabad,
- 3V: a jármű csatlakoztatás alatt van (ehhez a felhasználónak meg kell nyomnia egy gombot a csatlakozófejen),
- 1,5V: a jármű csatlakoztatott állapotban van.

Az 58. ábra mutatja be a Control pilot áramkör felépítését. Ha olyan jármű csatlakozik az állomáshoz, amely esetén szükség van a töltőoszlop hűtésére, az *R2*-es ellenállás értéke 270Ω az ábrán jelzett 1,3kΩ helyett.



58. ábra: Control pilot áramkör [39]

Az 5. táblázat összefoglalja a töltés egyes állapotaihoz tartozó értékeket (jelmagyarázat: CP – control pilot, EJ – elektromos jármű):

Állapot	CP Max [V]	CP Min [V]	Frekvencia [Hz]	EJ ellenállás [Ω]	Leírás
A	+12	-	0 vagy 1000	-	Nem csatlakozott
B	+9	-12	1000	2,74k	Csatlakozott
C	+6	-12	1000	882	Töltés
D	+3	-12	1000	246	Töltés hűtéssel
E	0	0	-	-	Hiba
F	-	-12	-	-	Hiba vagy ismeretlen állapot

5. táblázat: CP jel állapotai és azok jelentése [40]

A töltési protokoll lépései a következők [40] (az állapotok az 5. táblázat alapján vannak jelölve, a mért értékek a PWM jel csúcsértékei):

1. A jármű nincs csatlakoztatva, állapot: A.
2. A töltőkábel csatlakoztatásra került a járműhöz, amelynek hatására 9V-ot mér a töltőoszlop a CP vonalon. Állapotátmenet megy végbe, A-ból B-be lép a rendszer.
3. Az elektromos jármű készen áll a töltésre, ezt az *S2*-es kapcsoló bezárásával jelzi. Ekkor 3 vagy 6V-ot mér az oszlop a vezérlőjelen, állapotátmenet megy végbe. B-ből C-be vagy D-be lép át a rendszer a járműtől függően.
4. Az állomás bezárja a töltőáramkörét. A jármű meghatározza az időzítési és áramprofilt, de az áramfelvétel értéke nem lépheti át a PWM jel kitöltési tényezője által jelzett szintet, állapot: C vagy D.
5. Külső igény érkezik be a töltési teljesítmény csökkentésére. A jelzést a villamosenergia-hálózat vagy a töltőállomáson végrehajtott kézi beavatkozás következtében kapja meg a jármű. Ennek hatására a

vezérlőjel kitöltési tényezője alapján kiszámolt értékhez igazítja az áramigényét a jármű, állapot: C vagy D.

6. A töltés véget ér, a jármű kiadja a lecsatlakozási kérelmet. Ennek oka lehet például a Proximity vonal kinyitása. A jármű kinyitja az *S2* kapcsolót. Állapotátmenet megy végbe, C-ből vagy D-ből B állapotba lép át a rendszer.
7. A töltőállomás érzékeli az új állapotot, kinyitja a töltőcsatlakozót biztosító kapcsolót, állapot: B.
8. A töltőkábel lecsatlakoztatása a járműről lezárja a töltési folyamatot. Állapotátmenet megy végbe, B-ből A-ba lép át a rendszer.

8.2.3 CCS

A CCS (Combined Charging System, 59. ábra) a Type 2 csatlakozó AC típusú foglalatának DC vonalakkal kibővített változata. Az ábrán a bal oldali foglalat az USA-ban elfogadott szabványnak, míg a jobb oldali az európai előírásoknak felel meg. A foglalatba Type 2-es csatlakozót helyezve egy- és háromfázisú AC, míg Combo 2-es csatlakozóval DC töltés érhető el [42]. A töltőoszlop és az elektromos jármű között PLC (Power line communication [43]) kommunikáció folyik. A dolgozatban bemutatott töltőoszlopban a teljesítményelektronika fordítja le a töltésvezérlőtől érkező CAN üzeneteket a CCS protokollnak megfelelő vezérlésre [44].

59. ábra: CCS foglalat [42]

Az oszlop szempontjából a töltési protokoll a következő lépésekből áll:

1. Inicilizálás:
    a. elektromos jármű csatlakozik (töltőkábel csatlakoztatása),
    b. digitális kommunikáció (CAN) előkészítése,
    c. töltésvezérlési paraméterek cseréje (töltőoszlop részéről például kimeneti feszültség, jármű részéről például maximális akkumulátor feszültség),
2. Töltés előkészítése:
    a. töltőkábel „bereteszelése”,
    b. biztonsági teszt végrehajtása, esetleg előtöltés vagy töltéspróba végrehajtása,
3. Töltés:
    a. töltés indítása az inicializálás során egyeztetett paraméterekkel,
    b. folyamatos kommunikáció a járművel, állapotinformációk cseréje,
    c. kérés az energiaátvitel befejezésére (mindkét fél kiadhatja),
4. Töltés kikapcsolása:
    a. töltőáram lecsökkentése nullára,
    b. kimenő feszültség elvétele és ennek ellenőrzése,
    c. töltőkábel retesz kiengedése,
    d. digitális kommunikáció megszüntetése,
    e. töltőkábel lecsatlakoztatása a járműről.

8.3 Alrendszer integrációs tesztekhez szükséges elemek

Ebben a dolgozatban az eddig bemutatott munkám modul szintű tesztre ad példát. Egy teszt során a hardveres és funkcionális viselkedés is validálásra kerül, tehát a tesztelt eszköz hibamentes viselkedést tud biztosítani a következő szint számára.

Az alrendszer szintű integrációs teszteket egy valósághű emulációs környezetben kell végrehajtani. A tesztek célja a hatékony hibalokalizálás, így a későbbiekben a beüzemelésre fordított idő és költség csökkentése. A tesztrendszernek imitálnia kell a tesztelt elem valós környezetét, jelen esetben egy elektromos járművet kell realizálnia. A járműrealizáció két fő részből áll:

1. szabályozható fogyasztású műterhelés, amely lényegében a töltött akkumulátornak felel meg,
2. töltési protokollokat megvalósító elektronika és szoftver, amely a jármű oldali töltésvezérlést valósítja meg.

A töltőoszlopot két alrendszerre lehet bontani: teljesítmény- és vezérlőelektronikára. Ezeket külön-külön kell tesztelni az adott alrendszernek megfelelő járműrealizációval.

Az oszlopban alkalmazott töltési protokollok szabványosak, működésüket a 8.2-es fejezetben mutattam be. Az ezekhez szükséges elektromos jármű oldali és teljesítményelektronikai jeleket (mindkettő hardvertervezési feladat) az emulációs környezetnek kell előállítania. A vezérlőelektronika emulálásához terveztem egy autószimulátor áramkört. Ezt részletesen a 8.5-ös fejezetben fogom bemutatni. Továbbá a jármű töltőlogikáját (szoftverfejlesztési feladat) szintén ennek a környezetnek kell biztosítania a vezérlőelektronika alrendszer szintű integrációs tesztjeihez.

A vizsgálathoz többféle tesztelési forgatókönyvet kell kidolgozni, amelyeknek a töltés összes lehetséges állapotát (pl.: töltéselőkészítés CCS protokoll során) vagy állapotátmenetét (pl.: töltésmegszakítás a töltőkábel lecsatlakozása miatt) lefedik. A tesztelési forgatókönyvek lényegében egy tesztspecifikációt adnak ki, amely alapján már kidolgozható a tesztelési terv, majd azokból tesztkészletek stb. Az alrendszer integrációs tesztek főleg szoftveres hibák felderítésérét segítik. A hardveres problémákat az előző szint megfelelő elvégzése kiszűri, esetleg csatlakoztatási hibákra derülhet még fény.

A töltőoszlopban a teljesítményelektronika CAN-en és digitális be- és kimeneteken kommunikál a töltésvezérlővel. Az emulációs környezetnek ezeken a jeleken kívül a megfelelő fogyasztású műterhelést is biztosítania kell az alrendszer integrációs tesztekhez. A műterhelés felé követelmény, hogy AC töltés esetén akár 43kW-os, DC-nél pedig akár 50kW-os fogyasztással rendelkezzen. Az elvárt értékek a töltőoszlop maximális töltési teljesítményei. A terhelésnek szabályozható fogyasztónak

kell lennie, mivel a töltés során a jármű áramfelvétele, például a környezeti hőmérséklet vagy az akkumulátor töltési karakterisztikájának függvényében, változhat.

A témával részletesen nem foglalkozva első ötlet lehet műterhelés emulálására egy **akkumulátor** alkalmazása. Azonban ez a megoldás sok problémával jár. Az akkumulátor önmagában nem elég, szükséges a megfelelő töltőáramkör is mellé a töltőoszlopban található teljesítményelektronikán kívül. Nem megfelelő töltésfelügyelet például az akkumulátor felrobbanásához is vezethet, mivel nagy teljesítmény leadására képes a töltőoszlop. Továbbá problémát okoz az akkumulátor kapacitása és merítése. Amennyiben kis kapacitással rendelkezik, akkor gyorsan feltöltődik és nem lehet hosszú távon, például melegedés szempontjából, tesztelni a teljesítményelektronikát. Előnye a kis kapacitásnak, hogy gyorsan meríthető az akkumulátor. Nagy kapacitású akkumulátorral ugyan egy töltési folyamat részletesebben megfigyelhető, de a merítése problémát okoz. Az akkumulátorra csatlakoztatott terhelésnek olyan teljesítményfelvétellel kell rendelkeznie, hogy az időveszteség, amelyet a merítés okoz mindhárom töltési protokoll teszteléskor alkalmazva, pénzügyi szempontból is megérje. Ha ez nem teljesül, akkor másfajta műterhelés kialakítására van szükség.

A DC töltési protokollok (CHAdeMO, CCS) teszteléséhez kézenfekvő lehet műterhelésként egy, a teljesítményt az elektromos hálózatba visszatápláló, **DC-AC inverter**. Ilyen átalakítókat alkalmaznak például napelem farmoknál vagy kisebb szélturbináknál [45]. A visszatáplálás hátránya, hogy engedélyeztetni kell. Az átalakító szempontjából előnyt jelent, ha szabályozható a teljesítményfelvétele, mert könnyen szimulálható vele egy jármű töltése. Az AC-DC konverziót elvégző kiegészítő áramkörrel még AC töltéshez is alkalmazható az inverter. Számos inverter érhető el a piacon, nagyjából 1000 USD-s nagyságrendtől a 10000 USD-s nagyságrendig. A töltőoszlop tömeggyártásától függ, érdemes-e alkalmazni DC-AC átalakítót a tesztelésekhez.

8.4 Rendszertesztekhez szükséges elemek

Az **emulációs környezetben végzett rendszertesztekhez** egy teljes járműszimulátorra (nem csak műterhelésre, hanem vezérlőlogikára is) és töltőoszlopra van szükség. Ebben az esetben az emulációs környezet közvetlenül a töltőkábelre csatlakozik, nem az egyes alrendszerek be- és kimeneteihez. Ebből kifolyólag az emulációs környezetnek elegendő járműszimulátorként funkcionálnia, de mindhárom töltési protokoll számára biztosítania kell ezt.

Pénzügyi megfontolásokat figyelembe véve érdemes emulációs környezetben rendszerteszteket végezni. Feltéve, hogy egy inverterrel megoldott környezet ára 10000 USD körül mozog legdrágább esetben, egy új elektromos autó ára 25000 USD-nél kezdődik. Arról nem is beszélve, hogy három eltérő protokollal tölthető járműre van szükség, amelyet az emulációs környezet egyben képes lehet nyújtani. Bármilyen meghibásodásból kifolyólag, ha cserére vagy javításra van szükség, jobban megéri először az inverteres megoldással tesztelni töltőoszlopot.

A **valós környezetben végzett rendszertesztekre** azért van szükség, mert ekkor még felszínre kerülhetnek elektromos jármű specifikus hibák. Például nem megfelelő időzítés miatt az autó megszakítja a töltést. A rendszerteszteken általában a fejlesztők is részt vesznek. Ezt az indokolja, hogy ők tudják az esetlegesen felmerülő hibákat, problémákat kijavítani.

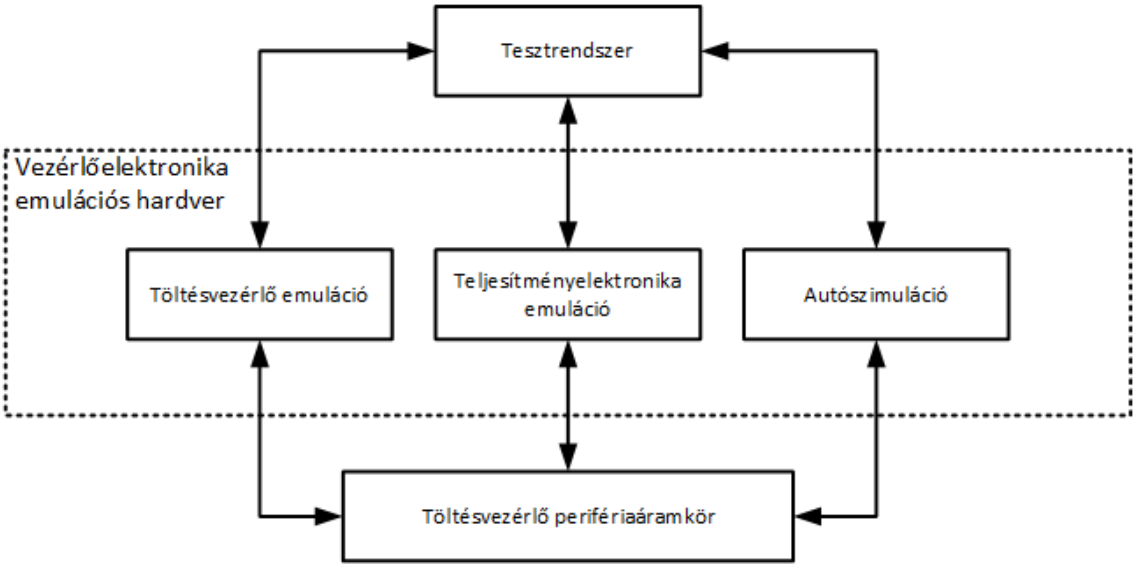
A modul és alrendszer szintű tesztek az egységek működésének validációjára törekedtek. Emellett a rendszertesztek már a **funkcionalitás** vizsgálatával is foglalkoznak. De valójában a funkcionalitás a végfelhasználói tesztek során kerül teljesen az előtérbe, mivel a felhasználó tudja, hogy mit szeretett volna. Más szemszögből látja a terméket, észlelhet hiányosságokat, kezelhetőségi problémákat stb. A végfelhasználói tesztek előnye, hogy a több, különböző jármű töltésével tovább lehet a rendszer működését finomítani.

8.5 Töltőoszlop vezérlőelektronikai alrendszerének emulációs környezete

Terveztem egy emulációs áramkört, amely a vezérlőelektronika alrendszer szintű tesztelésére alkalmas. A tesztrendszer az emulációs hardver segítségével a következő funkciókat képes megvalósítani (60. ábra):

- töltésvezérlő kártya és teljesítményelektronika emuláció,
- autószimuláció.

Mindegyik funkció egy-egy önálló egységet alkot, amelyek képesek egymástól teljesen függetlenül működni. Az emulációs áramkör egyben a töltésvezérlő perifériakártya modul szintű tesztelését is implementálja. Az áramköri tervek a töltésvezérlő teszthardveréhez hasonlóan Altium Designer-ben készültek. Legyártására még nem került sor.



60. ábra: Vezérlőelektronika emulációs hardver

8.5.1 Modul szintű tesztelés

A töltésvezérlő modul szintű tesztelését megvalósítottam, bemutattam a dolgozat 5. és 6. fejezetében. De az alrendszer teszteléshez szükség van az oszlopvezérlőnek és a töltésvezérlő perifériaáramkörének is a kártyaspecifikus tesztelésére. Ebben a fejezetben bemutatom a kiegészítőkártya hardveres és funkcionális tesztelésére alkalmas áramköri tervemet, amely egyben autószimulátor is.

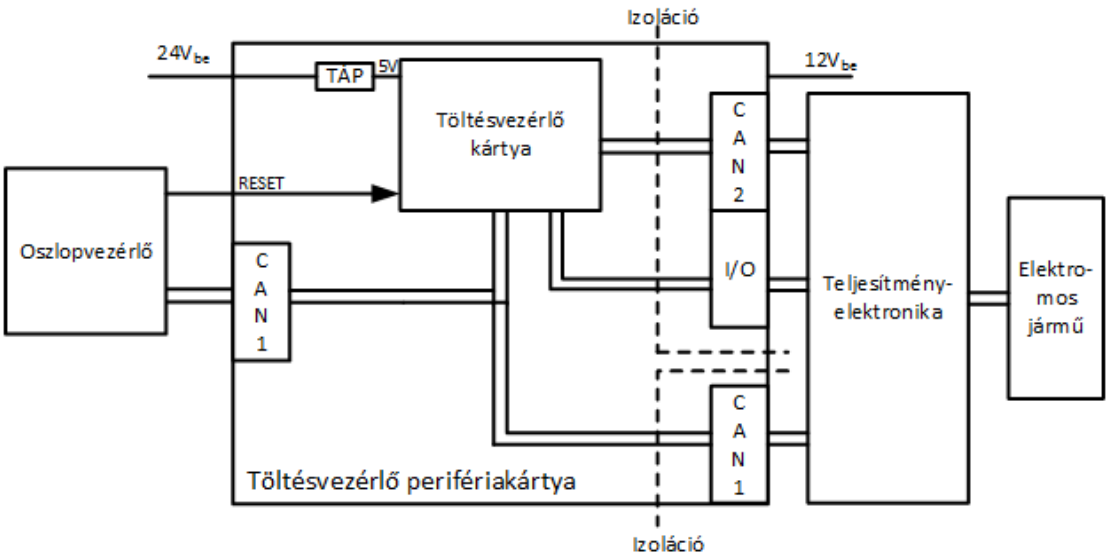
A töltésvezérlő és a **perifériakártyája** szoros kapcsolatban állnak egymással. A kiegészítő áramkör implementálja a szükséges hardver komponenseket az oszlop által biztosított töltési protokollokhoz. A modul szintű tesztelés elvégeztével lehet az alrendszer szintű tesztekre továbblépni, ahol már a két áramkör együttes működése kerül vizsgálatra.

A kiegészítő kártya **blokkvázlatát** a 61. ábra mutatja. A periféria kártyán két fő, egymástól független tápegység áramkör van, egy 12V-os és egy 24V-os oldal. Ezek egymástól galvanikusan le vannak választva, lassú jelek esetén (digitális be- és kimenetek) optocsatolóval, gyors jelek esetén (CAN) digitális izolátorral, analóg jelvonalakon pedig izoláló erősítővel. A perifériakártya a bemenő 24V-os tápvonalról állítja elő az 5V-os tápjelet a töltésvezérlő számára, egy feszültségszabályzó IC segítségével.

A töltésvezérlő az oszlopvezérlővel CAN-en kommunikál (CAN1-es interfész), valamint a külső RESET jelének vezérlését is az oszlopvezérlő áramkör végzi. A teljesítményelektronika ugyanazon a CAN buszon van, mint az oszlopvezérlő, de

galvanikusan izoláltak egymástól az adó-vevők. A CAN2-es interfészen keresztül a töltött járművel kommunikál az oszlop, CHAdeMO vagy CCS protokoll esetén.

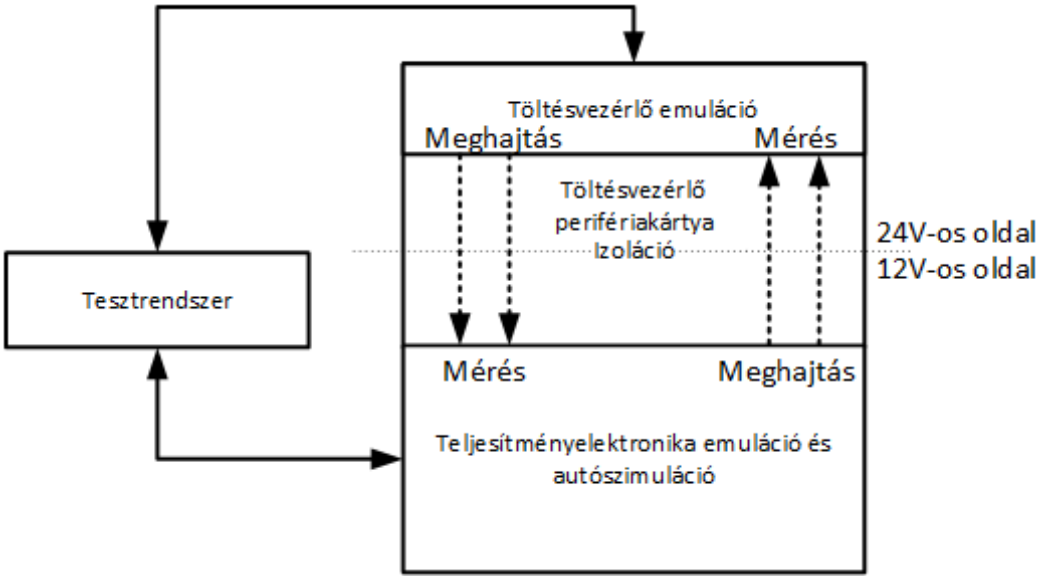
Az elektromos jármű és a teljesítményelektronika vezérlő- és kommunikációs jelei a 12V-os tápkörre csatlakoznak. A perifériakártyán található izolációs elemek szintillesztési feladatot is végeznek a töltésvezérlő jelvonalain (kimenetek esetén 3,3V-ról 12V-ra bemenetek esetén 12V-ról 3,3V-ra).



61. ábra: Töltésvezérlő áramkör perifériakártya blokkvázlat

A **modul szintű tesztek** elvégzéséhez nincs szükség a töltésvezérlő processzorra, az emuláció megvalósítja a töltésvezérlő analóg és digitális be- és kimeneteit. A tesztek során a tesztrendszer meghajtja a perifériakártya bemeneteit és a jelút másik végén visszaméri, hogy megfelelően működik-e a szintillesztés, izolálás stb. Kétféle bemenet értelmezhető a töltésvezérlő perifériaáramkörön (62. ábra):

1. teljesítményelektronika vagy elektromos jármű felől érkező jelek,
2. töltésvezérlő processzor kimenetei.



62. ábra: Modul szintű teszt töltésvezérlő emulálással

A tesztek célja, hogy az ültetési- és alkatrész hibából fakadó, problémákat kiszűrje. A teszthardver bemutatása során az emulációs funkciók bemutatására térek ki részletesen.

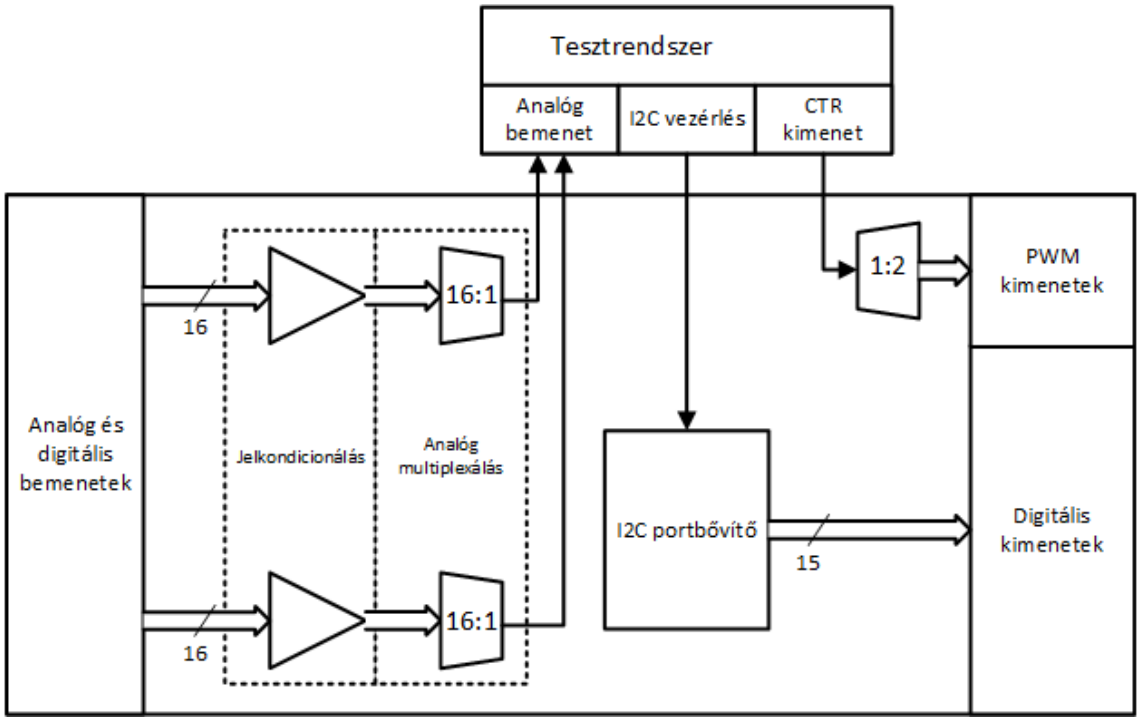
A perifériakártyának két egymástól független, izolált tápellátással rendelkező oldala van, ezek áramellátását egy-egy programozható tápegységgel oldottam meg. Mindkét oldalon további feszültségszinteket állít elő a bejövő tápvonalakról:

1. 24V-os oldal: 12V, 5V, 3,3V és analóg 3,3V,
2. 12V-os oldal: 5V, 2,5V, illetve -12V.

Ezek tesztjei a töltésvezérlő processzorhoz hasonlóan a felfutási idő, feszültségérték, zajosság mérésére kell, hogy kiterjedjenek.

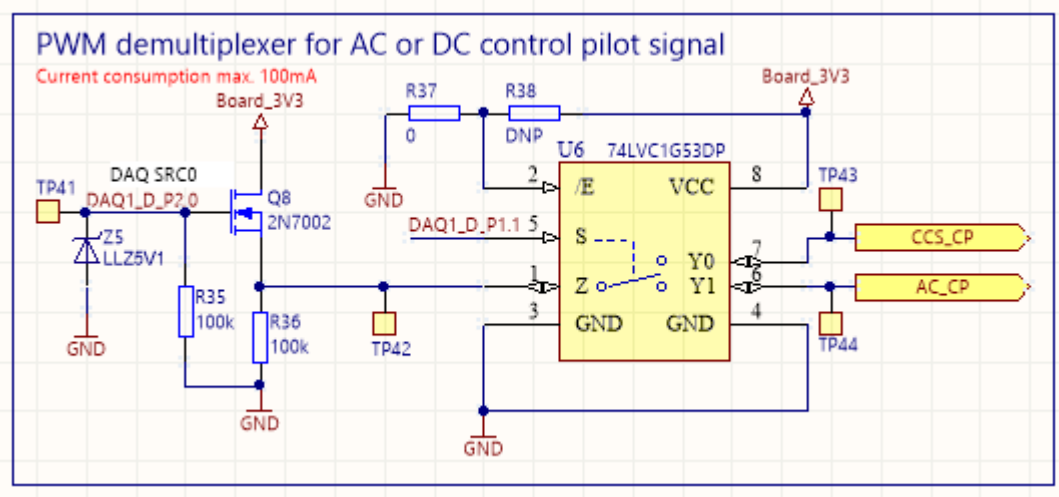
A **töltésvezérlő emuláció** során a **bemenő jelek** feszültségszintjeinek vizsgálatára van szükség, nem a logikai értékük meghatározására (63. ábra), így állapítható meg, nincs-e túl magas feszültség az egyes vonalakon, ezért a digitális bemenetek is feszültségméréssel kell tesztelni. A bemenetek egy-egy jelkondicionáló követő erősítőn keresztül két 16:1-es analóg multiplexerre vannak csatlakoztatva. A multiplexerek kimeneteit egy-egy DAQ kártya analóg bemeneten lehet visszamérni. A tesztrendszernek nincs elegendő analóg bemenete a 32 jel mérésére, ezért alkalmaztam a multiplexeres megoldást. A multiplexer vezérlését a DAQ kártya digitális vonalai valósítják meg.

Az emuláció nem állítja elő a CAN interfészek jeleit (RX-TX), hanem az operátor a tesztpontokon tudja a vonalak közti rövidzár vizsgálatot elvégezni.



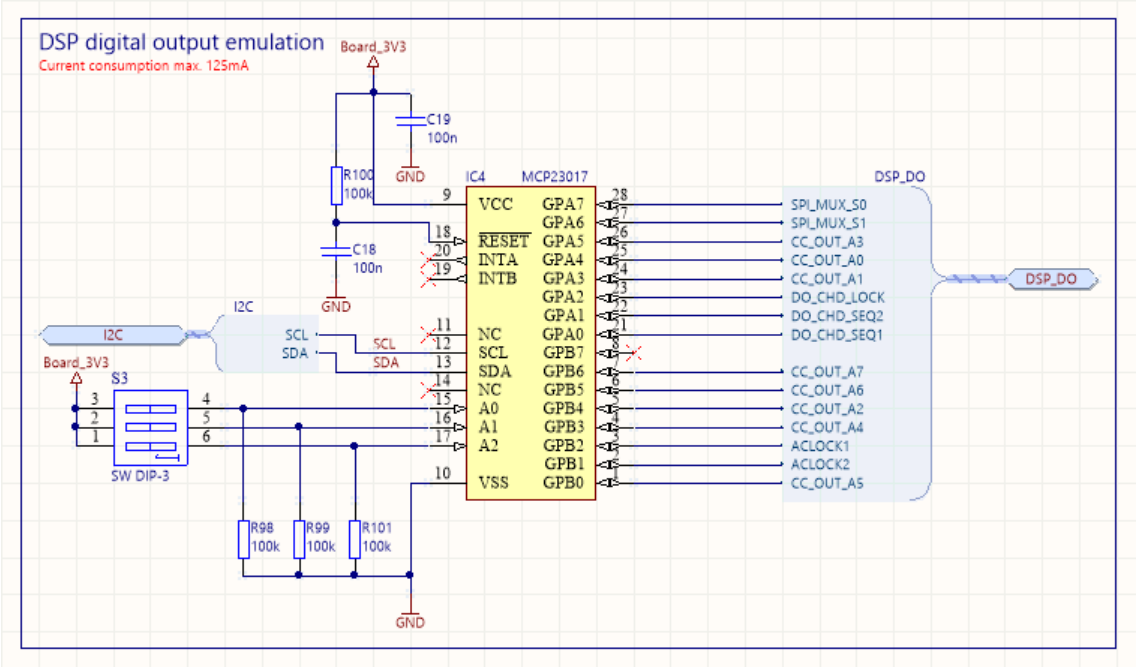
63. ábra: Töltésvezérlő kártya emuláció

A processzor **PWM kimeneteit** az NI DAQ kártya egyik időzítő/számláló modulja állítja elő. A másik időzítő/számláló végzi a PWM jelek visszamérését, ezért multi- és demultiplexálni kell a jeleket. Szintillesztésre is szükség van (5V-ról 3,3V-ra), ezt egy N-FET-es kapcsolással oldottam meg (64. ábra). A FET Drain vonala a szintillesztés kimenete. Amikor a Gate kivezetésen logika alacsony szint van, akkor a zárt állapotban van a Q8, így a kimeneten 0V van. Logikai magas szint (5V) esetén a FET kinyit és a Source-ra kötött 3,3V megjelenik a kimeneten. A FET Source vonala csatlakozik a multiplexer (U6) bemenetére, amelynek a kimenetei emulálják a töltésvezérlő PWM jeleit.



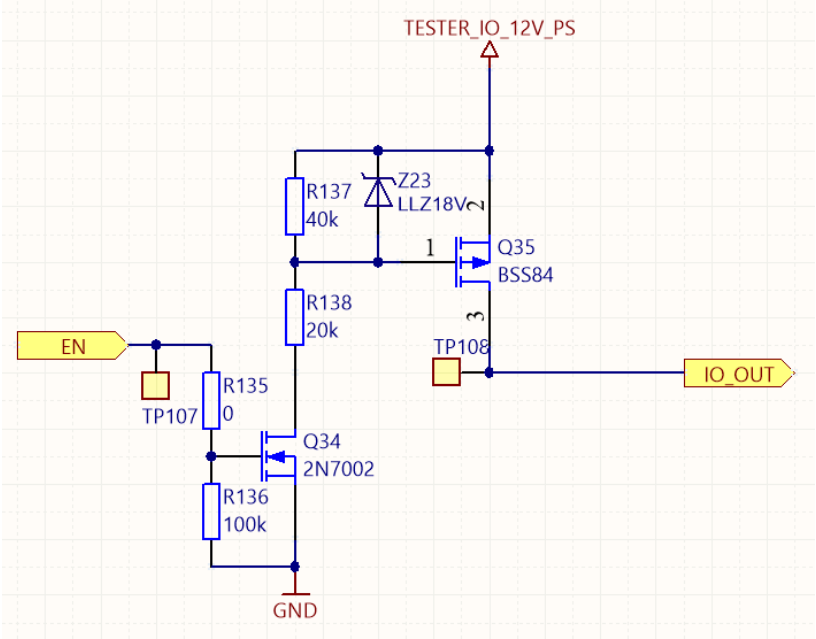
64. ábra: Töltésvezérlő PWM kimenetek emulációja

A töltésvezérlő kártya **digitális kimeneteit** egy I2C portbővítővel helyettesítettem (65. ábra). Mivel 3,3V-os jelei vannak a töltésvezérlő processzornak, ezért a portbővítő kimenetei ezzel a logikai magas jelszinttel dolgoznak.



65. ábra: Töltésvezérlő processzor digitális kimeneteinek emulációja

Nyolc **teljesítményelektronika vezérlést** végző kimenete van a töltésvezérlőnek, amelyek 12V-os jelszinttel dolgoznak. A teljesítményelektronika felől 16, szintén 12V-os jel érkezik, amelyek állapotai visszajelzéseket szolgáltatnak. A visszajelzések emulációját az alábbi áramkörrel valósítottam meg (66. ábra):



66. ábra: Teljesítményelektronika kimenet emuláció, egy vonalra

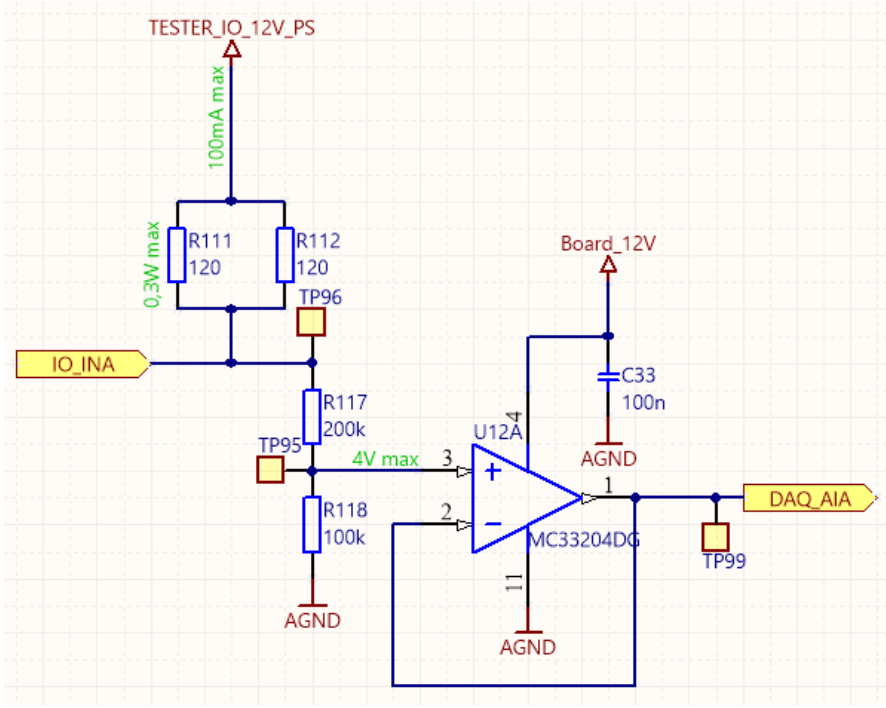
Az áramkörben egy N- (Q34) és egy P-csatornás FET (Q35) található. A Q35 Source vonalára 12V van csatlakoztatva. A működés az engedélyező jel logikai állapotától függően a következő:

1. logikai alacsony (0V): Q34 zárva van, ezért a Q35 Gate és Source vonala azonos potenciálon van, így Q35 is zárva van, ekkor a kimenet a perifériakártyán egy optocsatolón keresztül földre van kötve,
2. logikai magas (5V): Q34 nyitva van, ezért a Q35 Gate és Source vonala között 8V potenciálkülönbség lesz, így a P-FET kinyit és 12V-ot kapcsol a kimenetére.

A **teljesítményelektronikai bemenetek** működtetéséhez elvárt, hogy a töltésvezérlő perifériakártya kimenetei 100mA-es terhelés mellett is ki tudják adni a logikai jelszinteket (0, illetve 12V), tehát 12V/0,1A=120Ω terhelés szükséges. A perifériakártya kimenetei nagyimpedanciás vagy 60Ω-os ellenálláson keresztül földre húzott állapotban vannak, a vezérléstől függően (logikai magas, illetve alacsony), tehát a teszthardvernek 60Ω-os terheléssel kell rendelkeznie.

A terhelésen eső feszültség 12V*60Ω/120Ω=6V, így a teljesítménye 6V*0,1A=0,6W. Ezért a 60Ω-ot két párhuzamos 120Ω-os ellenállással valósítottam meg, mert így a 0,6W helyett 0,3W jut az egyes elemekre. Az áramkörben 1210-es típusú

ellenállásokat használtam, ezek teljesítménytűrése 0,5W. A teszthardveren az alábbi kialakítást valósítottam meg (67. ábra):



67. ábra: Teljesítményelektronika bemenet emuláció egy vonalra

Amennyiben logikai alacsony érték van a perifériakártya kimenetén, akkor 100mA áram folyik az emulációs hardver felől a perifériakártya felé. A jel állapotát az NI DAQ kártya analóg bemente méri vissza egy feszültségosztón és követő erősítőn keresztül.

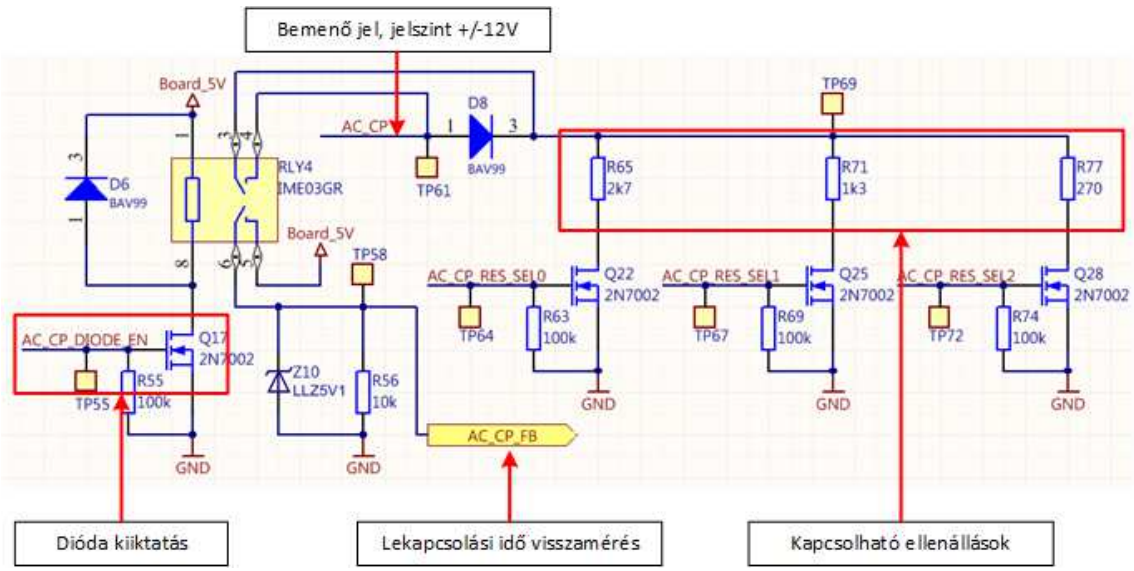
A három féle töltési protokollból csak az AC-hoz és a CHAdeMO-hoz van szükség **autószimulátor** áramkör megvalósítására. CCS protokoll estén egy teljesítményelektronikai egység állítja elő a szükséges vezérlőjeleket a töltéshez. A töltésvezérlő CAN-en kommunikál a CCS-t vezérlő egységgel, így szoftveresen megoldható a protokoll emulációja.

Az autószimulátor áramkört úgy alakítottam ki, hogy modul szintű tesztekre is képes legyen, mert a protokollok során alkalmazott vonalak validációjára szükség van az alrendszer szintű tesztelés megkezdése előtt.

Az **AC töltés** (8.2.2-es fejezet) során a jármű a **Proximity Pilot** vonalán jelzi, hogy maximálisan mekkora töltőáram felvételére képes. Az autó a jel és a földpontja közé egy ellenállást csatlakoztat, ennek értékétől függ a maximális érték. Szabvány alapján az

ellenállásértékek: 1,5kΩ (13A), 680Ω (20A), 220Ω (32A), 100Ω (63A), 50Ω (80A). Az autószimulátor mind az öt érték csatlakoztatására lehetőséget ad (Függelék, 77. ábra).

A **Control Pilot** jel a feszültségszintjével a töltés állapotát, a kitöltési tényezőjével pedig töltőáram értékét határozza meg. Az autó a jel és a földpontja közé ellenállásokat csatlakoztat párhuzamosan, amelyeknek az eredő értéke határozza meg a töltés állapotát. Az ellenállásértékek a szabvány alapján: 2,74kΩ, 882Ω, 246Ω. Az ellenállásokra egy diódán keresztül van csatlakoztatva a +/-12V-os CP jel, így a negatív tartomány feszültségértékét nem befolyásolja az ellenállások értéke. Amennyiben a töltés folyamán a jel negatív csúcsértéke leosztásra kerül (dióda nem funkcionál), akkor a szabvány szerint a töltést 100ms-on belül le kell állítani. Az általam megtervezett áramkör lehetőséget ad az ellenállások csatlakoztatására, valamint a dióda kiiktatásával a lekapcsolási idő mérésére (68. ábra):

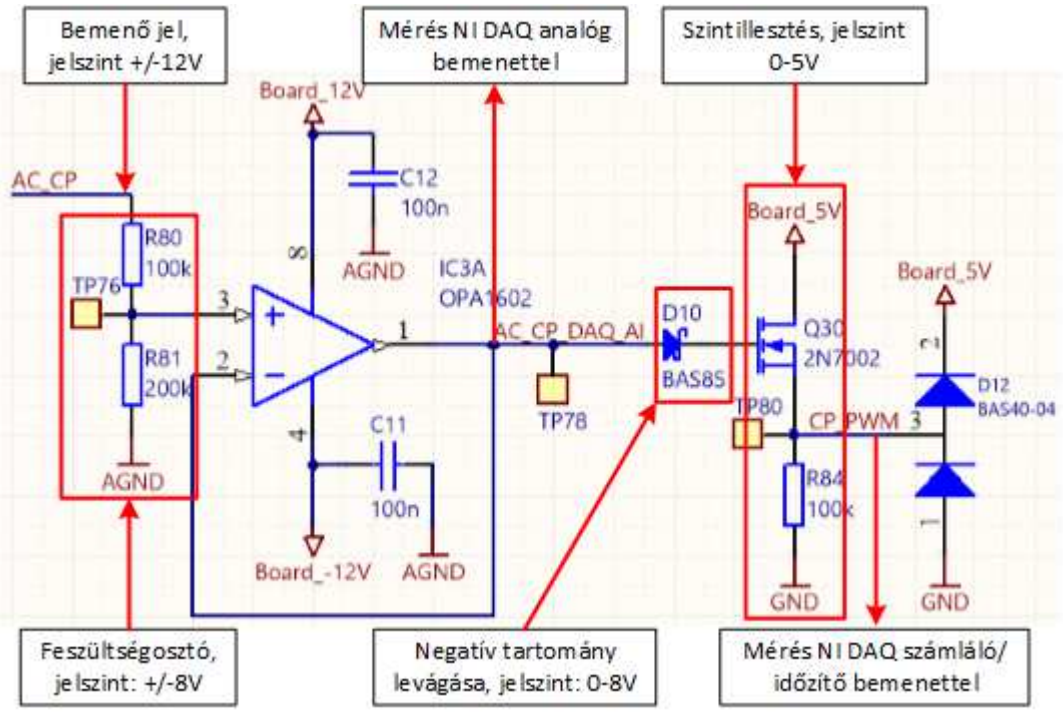


68. ábra: AC autószimulátor, CP jel

Ha a Q17-es N-FET zárt állapotában van, akkor a dióda funkcionál, egy 10kΩ-os ellenálláson keresztül pedig 0V van csatolva a lekapcsolási időt mérő digitális bemeneten. A FET nyitott állapotában a relé rövidre zárja a diódát, valamint 5V-ot kapcsol a digitális bemenetre.

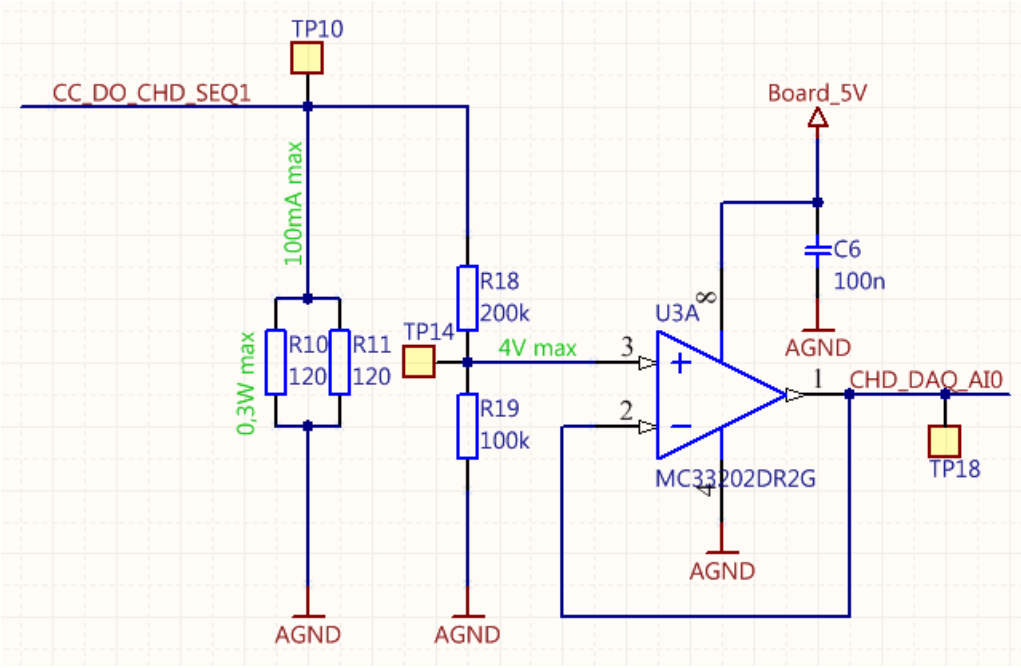
A Control Pilot mérését a következő áramkör valósítja meg (69. ábra). Egy feszültségosztón és egy követő erősítőn keresztül az NI DAQ kártya az egyik analóg bemenetén méri a jel feszültségszintjeit. A frekvenciát és a kitöltési tényezőt a számláló bemenetén képes visszamérni a DAQ kártya. Ehhez viszont először le kell vágni a negatív

tartományát a jelnek, majd egy szintillesztést is végre kell hajtani, hogy 0-5V-os tartományban mozogjon a PWM jel.

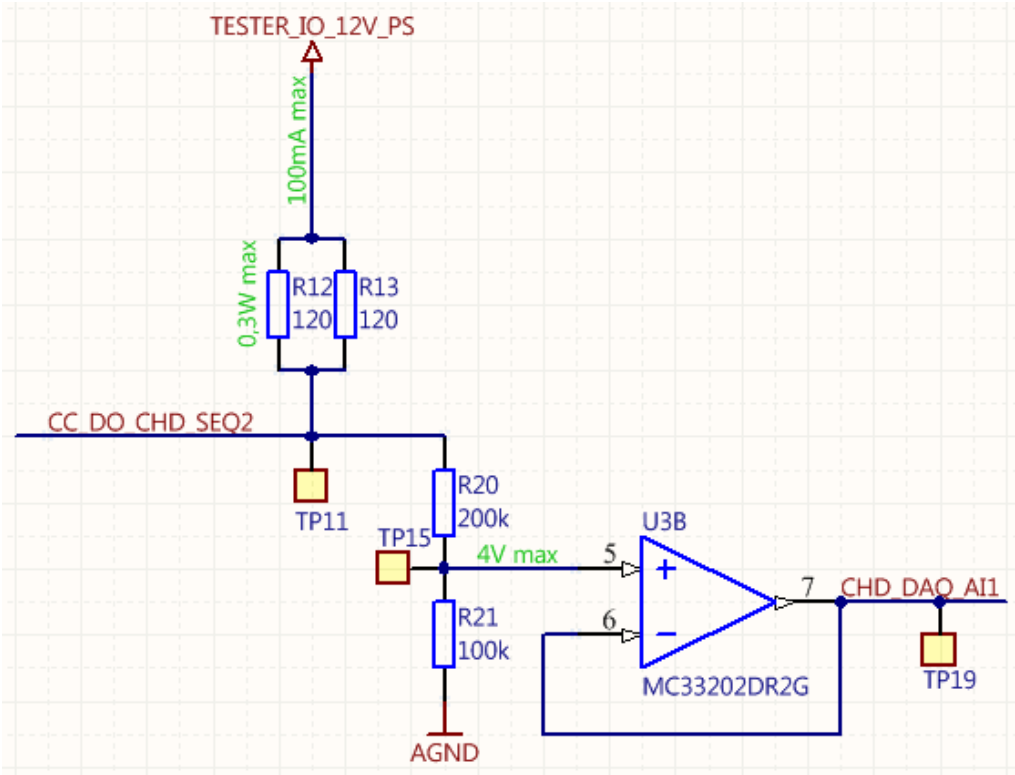


69. ábra: AC autószimulátor, CP jel mérés

A **CHAdeMO** protokoll (8.2.1-es fejezet) emulálásához az autószimulátor szempontjából a két-két be- (SEQ1, SEQ2) és kimenetre (Permission, Proximity) van szükség. A teljesítményelektronika vezérléséhez hasonlóan elvárt a töltésvezérlő perifériakártyától, hogy 100mA-es terhelés alatt is képes legyen kapcsolni a kimenetét. A **SEQ1** 12V-t, a **SEQ2** földpotenciált vár el a bemenetén a jelváltáshoz, ebből kifolyólag föld felé és 12V felől kell a terheléseket kialakítani. A periféria kártyán teljesítményelektronikai kimenetekhez hasonlóan soros 60Ω-os ellenállások találhatók, ezért szintén 60Ω-ot szükséges csatlakoztatnia a teszthardvernek. Ennek megfelelően terveztem meg a két alábbi kapcsolást (70. ábra, illetve 71. ábra):



70. ábra: CHAdeMO emuláció, SEQ1

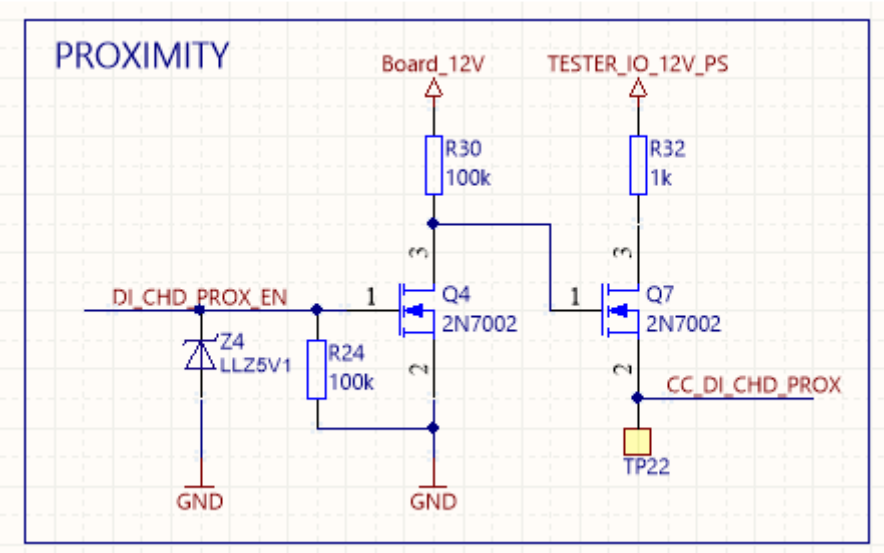


71. ábra: CHAdeMO emuláció, SEQ2

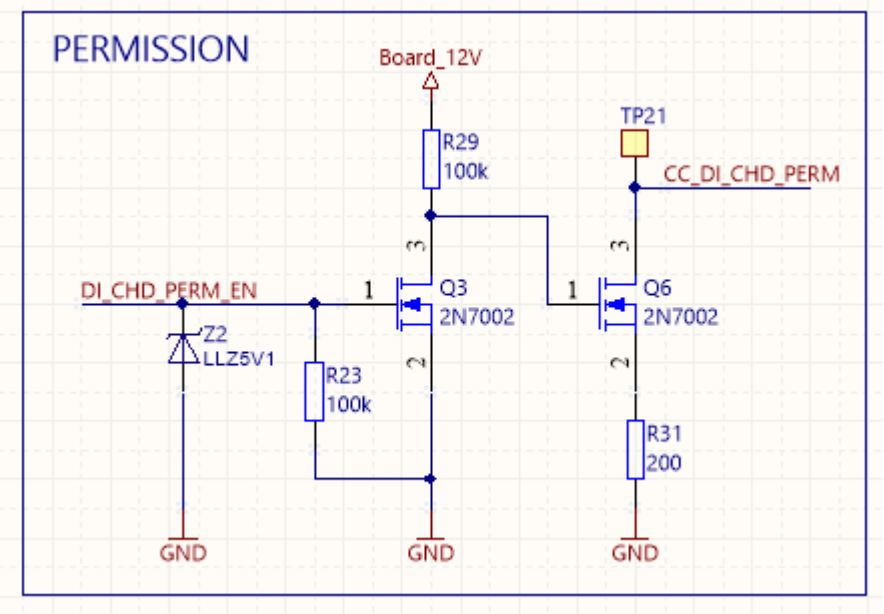
A jelszinteket az NI DAQ kártya egy-egy analóg bementén lehetséges mérni.

Az autó **Proximity** (csatlakozást jelző) kimenetén 12V-on keresztül 1kΩ kapcsolását, a **Permission** kimeneten pedig 200Ω-on keresztül földkapcsolást várja el a

töltőoszlop. Ezekhez az alábbi áramkörrészleteket terveztem meg: (72. ábra, illetve 73. ábra):



72. ábra: CHAdeMO emuláció, Proximity

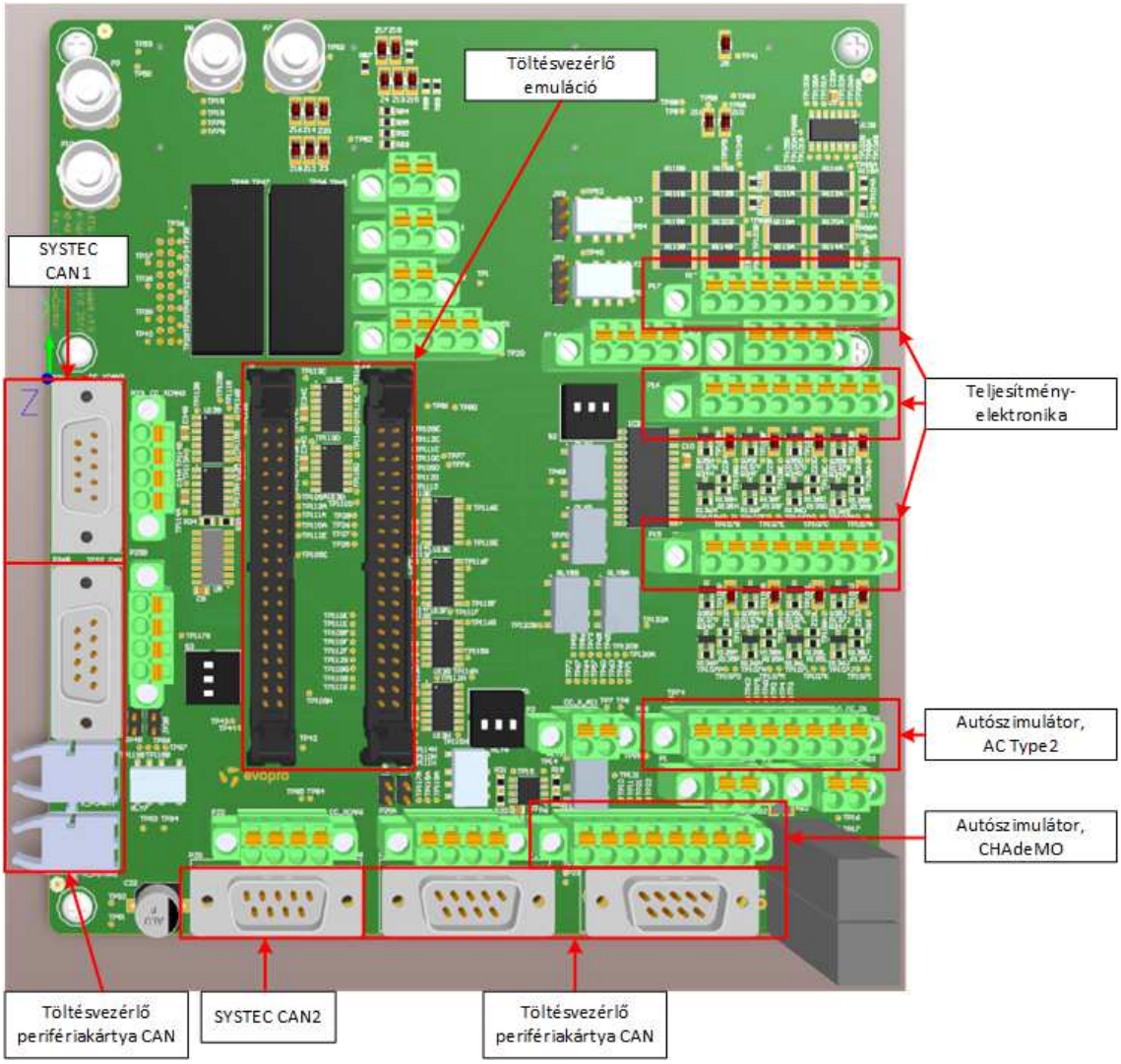


73. ábra: CHAdeMO emuláció, Permission

A kapcsolások megvalósításához N-csatornás FET-eket használtam. A tervezés során végeztem szimulációkat [46], amelyek alapján ahhoz, hogy a Q6, illetve a Q7 teljesen kinyisson, szükséges még egy fokozat, amely 12V-t kapcsol a Gate vonalaikra.

A megtervezett áramkör **3D modelljét** a 74. ábra mutatja. A tüskesorokon érhetők el a töltésvezérlő emulált jelei. Az öt D-Sub, valamint az optikai csatlakozó CAN interfészek összekötésére adnak lehetőséget. Nyolc pólusú sorcsatlakozókon érhetők el a

teljesítményelektronika és autószimulátor áramkörök. Az emulációs hardver a töltésvezérlő tesztáramkörhöz hasonlóan két feltétkártya helyet foglal el a tesztműszer interfészáramkörön.



74. ábra: Emulációs hardver 3D modell, felülnézeti kép

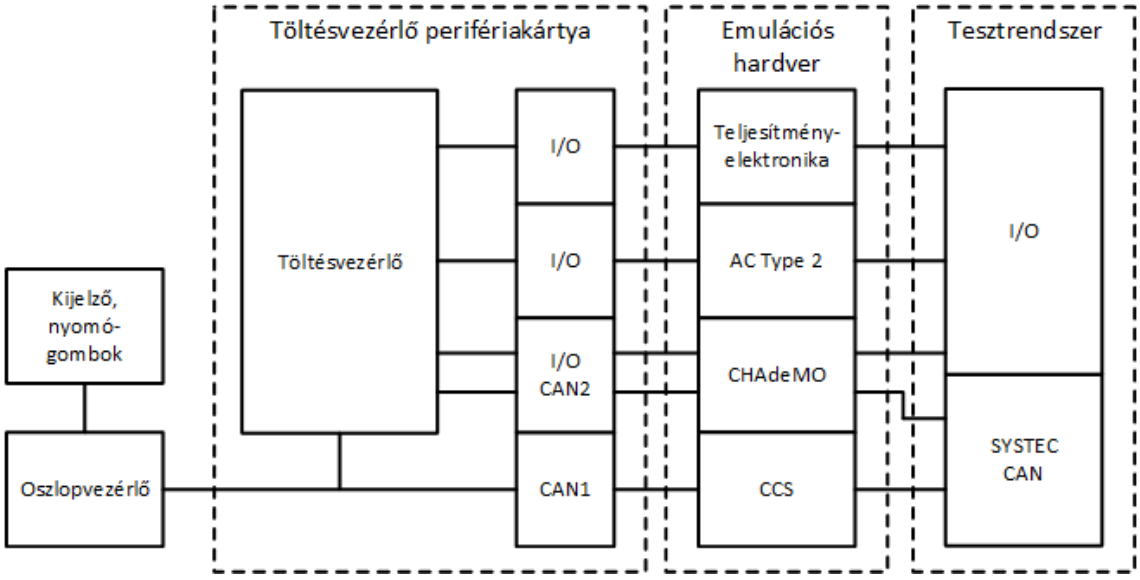
8.5.2 Alrendszer szintű tesztelés

Ez a fejezet kitekintés, egy tervet mutat be, amely alapján megvalósítható alrendszer szintű tesztelést.

A töltőoszlop vezérlőegységeinek alrendszer szintű tesztelésnek előfeltétele a következő áramkörök modul szintű tesztelése: oszlopvezérlő, töltésvezérlő és töltésvezérlő perifériakártya. Amennyiben ez az előfeltétel teljesül, akkor összeállítható a vezérlőegységek alrendszere (75. ábra):

- az oszlopvezérlő CAN1-en kommunikál a töltésvezérlővel, valamint kezeli a kijelzőt és a nyomógombokat,
- a töltésvezérlő az I/O vonalain és CAN interfészeken keresztül kapcsolódik az emulációs hardverhez,
- a tesztrendszer két CAN interfésze csatlakozik az emulációs hardveren keresztül a töltésvezérlő CAN buszaira, a be- és kimeneti vonalai pedig a teljesítményelektronikai és autószimulációs áramkörök mérését és vezérlését végzik.

Ebben az összeállításban mindhárom töltési protokoll, valamint a teljesítményelektronika is emulálható.



75. ábra: Vezérlőegységek alrendszer szintű tesztelése

A töltési protokollok leírhatók állapotdiagramokkal, ezért a töltési folyamat megvalósítható szekvenciális lépések sorozatával, amelyeket például a TestStand implementál. Amennyiben párhuzamosan futó műveletekre van szükség, azok is elvégezhetők, a tesztelési keretrendszer támogatja a több szálon futó programvégrehajtást is. Több szálon futásra szükség lehet például párhuzamos töltés emulálása alatt. Ekkor egyszerre két jármű csatlakozik a töltőoszlophoz, egyik az AC, a másik pedig valamelyik DC töltőáramkörre.

9 Összefoglalás, továbbfejlesztési lehetőségek

Diplomatervemben sikerült megvalósítanom a feladatkiírásban szereplő pontokat. Munkám során rendszerszinten kellett a tesztelés témakörével foglalkoznom. Megismerkedtem az evopro Innovation Kft.-nél rendelkezésre álló tesztelési infrastruktúrával. A rendelkezésemre álló részegységekből egy jól átlátható, egységes keretrendszert dolgoztam ki.

A rendszer működésének demonstrálására megterveztem a töltésvezérlő kártya teszthardverét, majd sikeresen integráltam a tesztszoftver elkészítésével. Ennek során elsajátítottam az áramkör tervezés alapjait, valamint elmélyítettem a LabVIEW-s és TestStand-es tudásomat.

A kidolgozott félautomata rendszerrel a tesztelésre fordított időt 89%-kal csökkentettem az eredetihez képest a töltésvezérlő áramkör esetében, valamint így a tesztelések szakértő bevonása nélkül is elvégezhetők.

Megterveztem egy autószimulátor áramkört, amely a töltőoszlop emulációs környezetének kialakítására nyújt lehetőséget. Az áramkör megtervezéséhez elengedhetetlen volt a töltési protokollok működésének megértése.

A funkcionális tesztek teljesen automatikusak, de a tápegység tesztelés operátori beavatkozást igényel. Ezt csak a teljesen automatikus teszteléssel lehet megszüntetni, amellyel a tesztelésre fordított időt tovább lehet csökkenteni. Megoldást jelenthet egy flying probe integrálása a rendszerbe, amely az oszcilloszkóp mérőfejét a tesztpontokhoz tudja érinteni. Ehhez definiálni kell egy koordináta-rendszert a vizsgált áramkörre, amelyben az egyes mérőpontok pozícióját megadva a flying probe képes kiváltani az operátori beavatkozást.

Szintén megoldást jelenthet az automatizálásra egy tűágy és egy relémátrix alkalmazása. Ekkor minden tű egy-egy tesztpontra csatlakozik a tesztelt áramkörön, a relémátrix pedig kijelöli, melyik tűpárok (mért jel és föld) jeleit mérik az oszcilloszkóp csatornái. A tűágy kialakítása igen költséges, számos mechanikai elemre is szükség van az elektronikai komponenseken kívül, csak akkor érdemes kialakítani, ha a tesztelt eszköz nagyszériás sorozatgyártásban készül.

Irodalomjegyzék 2016.12.18.

[1] John McGregor, "Test early, test often", in Journal of Object Technology, vol. 6, no. 4, May-June 2007, pp. 7-14, URL: http://www.jot.fm/issues/issue_2007_05/column1

[2] Scherer Balázs, Rendszertervezés és –integráció (VIMIMA11): Tesztelési ág, URL: https://www.mit.bme.hu/system/files/oktatas/targyak/10019/VIMIMA11_RTI_0607_Teszteles.pdf

[3] evopro Kft., URL: http://www.evopro.hu/

[4] evopro Kft. Villámtöltő, URL: http://www.evopro.hu/uploads/FlashCharger_HU_web.pdf

[5] National Instruments TestStand, URL: http://www.ni.com/teststand/

[6] evopro Villámtöltő, URL: http://www.evopro.hu/uploads/FlashCharger_HU_web.pdf

[7] Maform Kft., URL: http://www.maformdesign.com/

[8] PROCON Hajtástechnika Kft., URL: http://www.procon.hu/

[9] Toradex Ag, URL: https://www.toradex.com/

[10] FT Prog, URL: http://www.ftdichip.com/Support/Utilities.htm#FT_PROG

[11] C2Prog, URL: http://www.codeskin.com/programmer

[12] National Instruments, URL: http://www.ni.com/

[13] National Instruments LabVIEW, URL: http://www.ni.com/labview/

[14] PXISA, URL: http://pxisa.org/

[15] NI PXIe-1075, URL: http://sine.ni.com/nips/cds/view/p/lang/hu/nid/205962

[16] NI PXIe-8135, URL: http://sine.ni.com/nips/cds/view/p/lang/hu/nid/210545

[17] NI PXI Chassis, URL: http://www.ni.com/tutorial/4811/en/

[18] NI PXI-6250 DAQ mérőmodul, URL: http://sine.ni.com/nips/cds/view/p/lang/hu/nid/14122

[19] NI VirtualBench, URL: http://www.ni.com/virtualbench/

[20] SYS TEC electronic GmbH, URL: http://www.systec-electronic.com/

[21] SYS TEC USB-CANmodul2, URL: http://www.systec-electronic.com/en/products/industrial-communication/interfaces-and-gateways/can-usb-converter-usb-canmodul2

[22] TENMA tápegységek, URL: https://sigrok.org/wiki/Korad_KAxxxxP_series

[23] evopro Innovation Kft. belső használatú dokumentáció, Moduláris tesztrendszer alkalmazása

[24] SHC68-68-EPM, URL: http://sine.ni.com/nips/cds/view/p/lang/hu/nid/201628

[25] MCP23017 I/O bővítő, URL: http://ww1.microchip.com/downloads/en/DeviceDoc/21952a.pdf

[26] NI PXIe-6341, URL: http://sine.ni.com/nips/cds/view/p/lang/hu/nid/207415

[27] R&S HMC8043, URL: https://www.rohde-schwarz.com/hu/product/hmc804x-productstartpage_63493-61542.html

[28] MCP 2551 CAN transciever, URL: http://ww1.microchip.com/downloads/en/DeviceDoc/21667f.pdf

[29] CD74HCT4067 analóge multi-/demultiplexer: URL: http://www.ti.com/lit/ds/symlink/cd74hct4067.pdf

[30] Altium Designer, URL: http://www.altium.com/

[31] Agilent MSO-X 3012, URL: http://web.mit.edu/6.115/www/document/agilent_mso-x_manual.pdf

[32] Tektronix 2000, URL: http://www.tek.com/oscilloscope/tds2000-digital-storage-oscilloscope

[33] CHAdeMO Association, URL: http://www.chademo.com/

[34] Wikipedia – CHAdeMO, URL: https://en.wikipedia.org/wiki/CHAdeMO

[35] Role of CHAdeMO, URL: http://www.chademo.com/wp/role/challenge/

[36] CHAdeMO technical workshop, URL: http://www.chademo.com/wp/blog/2013/01/11/technical-workshop/

[37] MENNEKES Type 2 AC, DC csatlakozó, URL: http://www.teslaforum.dk/

[38] MENNEKES GmbH, URL: http://www.mennekes.de/

[39] Control Pilot áramkör, URL: http://standards.sae.org/j1772_199610/

[40] Pilot signal communication, URL: https://webstore.iec.ch/publication/6029

[41] CCS csatlakozó, URL: http://www.phoenixcontact-emobility.com/en/products/e-mobility-connectors/vehicle-inlets

[42] CCS specifikáció, URL: http://www.charinev.org/ccs-at-a-glance/ccs-specification/

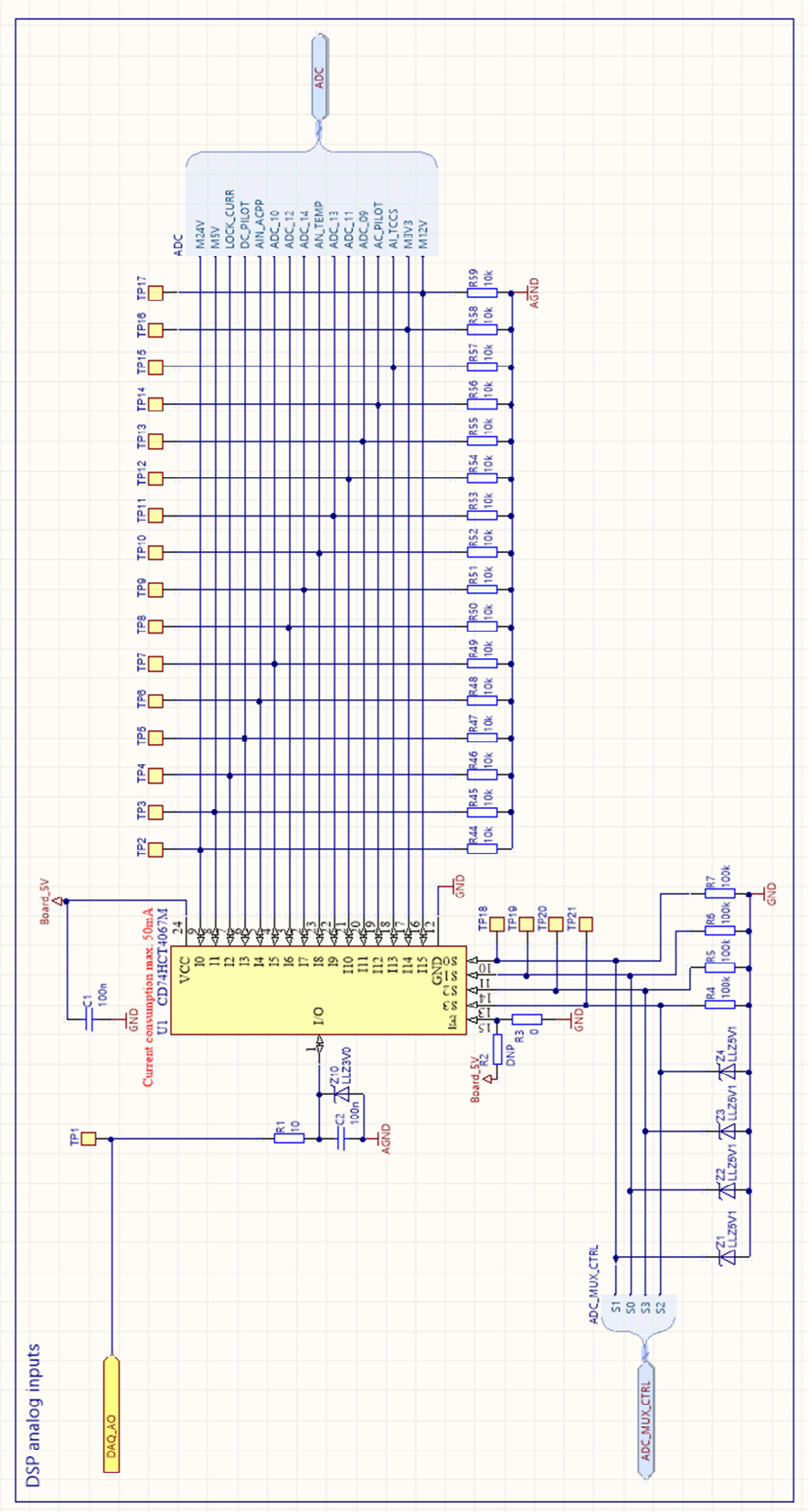
[43] Power-line communication, URL: https://en.wikipedia.org/wiki/Power-line_communication

[44] IEC 61851, URL: https://webstore.iec.ch/publication/6033

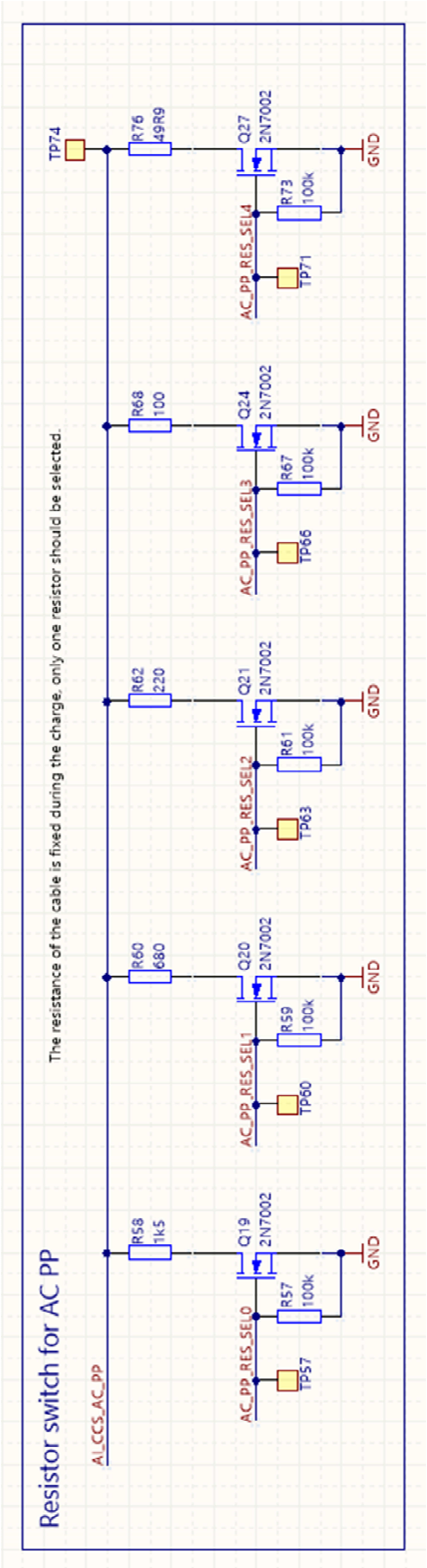
[45] Grid-tie inverter, URL: https://en.wikipedia.org/wiki/Grid-tie_inverter

[46] Falstad szimulációs szoftver, URL: http://www.falstad.com/circuit/

Függelék



76. ábra: Töltésvezérlő processzor analóg bemenetek



77. ábra: AC autószimlátor, Proximity Pilot jel