Információfeldolgozás laboratórium
Automatizált tesztelés mérés

Mérési útmutató
2016

Csak belső használatra

Scherer Balázs

Bevezetés

Az automatizált tesztrendszerek mára már széleskörűen elterjedtek, mind a gyártás utáni ellenőrzésben, mind a fejlesztés során a hosszú távú, illetve regressziós teszteknél. A tesztautomatizálás jelentős idő- és erőforrás-megtakarítást eredményez, ami tendenciává tette a lehető legtöbb helyre való bevezetését. Ezzel a folyamattal párhuzamosan természetesen megjelentek azok a környezetek is, amelyek segítik az ilyen tesztek kifejlesztését.

A mérés során célunk, hogy megismerkedjünk a tesztautomatizálás alapvető koncepciójával és problémáival, valamint betekintést nyerjünk az egyik legjobban elterjedt tesztautomatizáló szoftver az NI TestStand [1] használatába.

Az automatikus tesztek készítése viszonylag szerteágazó tudást igényel: ismerni kell a tesztelendő eszközt és a teszt céljait, ismerni kell a teszt végrehajtásához szükséges műszerek kezelését, valamint a különálló teszteket egységbe fogó tesztautomatizáló környezetet is. Ennek megfelelően a mérést bemutató dokumentáció is több részből fog állni:

1. ***A mérési feladat és környezet bemutatása***
2. ***Az alkalmazott műszerek kezelésének ismertetése***
3. ***Az NI TestStand szoftver bemutatása***
4. ***Javasolt lépések a feladat végrehajtásához***

A mérés előtt a dokumentáció elolvasása, megértése szükséges. Anélkül a mérést nem lehet időben teljesíteni! A mérés szintén feltételez egy minimális LabVIEW alapismeretet, amit elméletileg az előző félévben mindenki megszerzett. Aki hiányokat érez ezen a területen nézze át egy picit az alap LabVIEW praktikákat!

1. A mérési feladat és környezet bemutatása

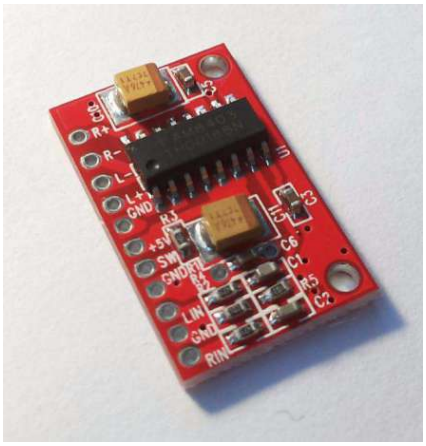
Cégünk az AnalogDeVicces úgy döntött, hogy új termékével a minőségi D osztályú audió erősítő moduljával megcélozza a világpiacot. A kor követelményeinek megfelelően a termék árusításánál az online csatornákat fogjuk előnyben részesíteni. A konkurencia legyőzését pedig attól reméljük, hogy az aliexpress, gearbest és hasonló felületek igényességükről híres vásárlói minőségszeretetére építünk, és minden egyes modulunkhoz letölthető formában elérhetővé tesszük a teszt jegyzőkönyvet. Ezáltal is biztosítva azt, hogy nálunk nem fordulhat elő – mint a konkurenciánál –, hogy a vevő nem jól működő terméket kap.

Persze a tesztelés és tesztfejlesztés költséges feladat. Annak bizonyítására, hogy mégis megéri végrehajtani ezt a feladatot, egy demó tesztrendszert vár el tőlünk a főnökünk, de ennek elkészítésére csak 8 munkaóránk van (szerencsére a hardware-es kolléga a tesztkörnyezetet már összeállította).

Magát a tesztelést a kínai gyárunkban fogja egy kolléga végezni, aki a beültetésért is felelős. Így a tesztre a következő megkötések vannak: gyorsabbnak kell lennie a modulok kézi beültetésénél, és automatikusan kell végrehajtódnia, hogy a beültetéssel párhuzamosan is végezhető legyen.

1.1. A tesztelendő modul

A tesztelendő modulunk egy PAM8304 [2] típusú D osztályú erősítőre épülő egyszerű áramkör, amely tartalmazza az erősítő minimális táphidegítését és egy bemeneti osztót/szűrőt. A modul az 1.1. ábrán látható.

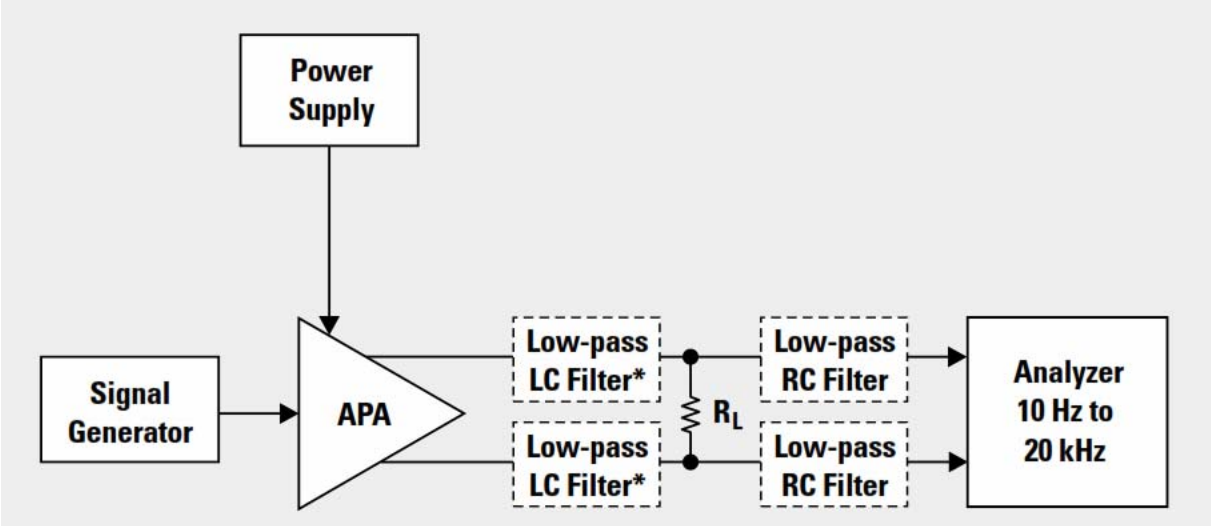


***1.1. ábra**. PAM8403 audio amplifier modul* [3]

1.2. D osztályú teljesítményerősítők mérésének elméleti összefoglalója

Ez az összefoglaló nem szándékozik részletesen bemutatni hely és idő hiányában a D osztályú erősítők mérésének elméleti alapjait. Akit ez a téma jobban érdekel, számtalan irodalom áll rendelkezésére [4], [5].

Röviden összefoglalva egy D osztályú erősítők mérésére használt tipikus elrendezés az 1.2. ábrán látható.



***1.2. ábra**. D osztályú erősítők tipikus mérési elrendezése [5]*

Az ábrán szereplő LC szűrő szerepe, hogy a kapcsoló üzemű erősítő kimenete ne kerüljön közvetlenül az ohmos R_L terhelésre, mert különben a kapcsoló frekvencián (a PAM8403 esetében ~220 kHz) leadott teljesítmény az összes hatásfokhoz kapcsolódó mérést értelmezhetetlenné teszi. A kimenő LC szűrőre a valóságban, erősen induktív hangszórók esetében nem feltétlenül van szükség, de azért sok esetben javasolják a használatukat. *(A mi esetünkben ezt az LC szűrőt (15 uH, 220 nF) az ajánlott mintakapcsolás részének vesszük, így nem kell kompenzálni a hatását a mérésnél.)*

Az analizátor előtti RC szűrő szerepe (100 ohm, 47 nF) egyszerű átlapolódás gátló szűrőként szolgálni, megakadályozva a kapcsolási frekvenciás zaj belapolódását a mérési tartományba. *Ez a szűrő nem része a valós ajánlott mintakapcsolásnak, így ha pontos mérésre törekszünk, a hatását kompenzálni kell. (A mi esetünkben gyakorlatilag elhanyagolhatjuk a kompenzációt, mert nem ez a lényeg)*

A szűrők kiválasztásához további információkat találhatunk a megfelelő szakirodalomban [5], [6].

Erősítés, Gain:

Audio erősítők egyik legtipikusabb mérése az erősítési tényező frekvencia függvényében való meghatározása. Ez a gyakorlatban egy egyszerű amplitúdókarakterisztika mérés (*a mi esetünkben annyi megjegyzés tartozik hozzá, hogy mivel a modul tartalmaz egy bemeneti osztót és AC csatoló áramkört, ezért nem a PAM8403 adatlapjában található értékeket fogjuk visszakapni a mérésnél*).

Harmonikus torzítás, THD mérés:

A harmonikus torzítás az audio erősítők jóságára vonatkozó egyik legelterjedtebb jellemző. Gyakorlatban két verzióban THD (Total Harmonic Distrosion), vagy THD+N (Total Harmonic Distorsion plus Noise) formában szokták megadni a mérések eredményeit [7]. A szakirodalom számos érvet, és ellenérvet sorakoztat fel a THD, vagy a THD+N adat használata mellett [4]. Mi a THD adat használata mellett döntünk, mert a THD+N a nevéből

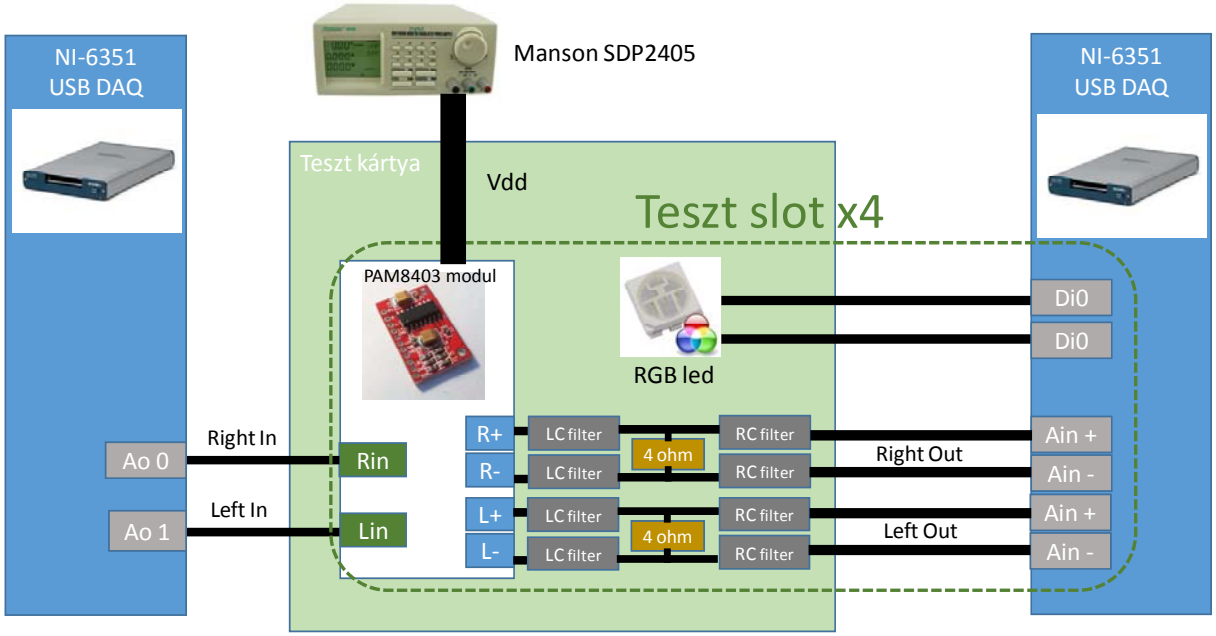
adódóan is jóval érzékenyebb a zajra, így a mi zajjal terhelt laboratóriumi környezetünkben a THD adatok megbízhatóbbak és jobban jellemzik az audio erősítőnk működését. A THD számítása körül is számos eltérés figyelhető meg mérésről mérésre, a mi esetünkben a 6. harmonikusig javasoljuk a mérést, úgy, hogy a 20 – 20.000 Hz-es tartományt vesszük figyelembe, tehát a 20 kHz felett egy szűrővel elnyomjuk a jeleket.

Hatásfok mérése:

A hasznos kimenőjel teljesítménye / az erősítő által felvett teljesítmény. Mérésére valószínűleg sajnos nem fog elég idő jutni.

1.3. A modul tesztelésére használt környezet

A PAM8403 modul tesztelésére használt rendszer blokkdiagramját az 1.3. ábrán láthatjuk. A rendszer áll egy Manson SDP2405 programozható tápegységből, egy NI USB-6351 DAQ adatgyűjtő kártyából, valamint egy teszt NYÁK-ból, amely 4 PAM8403 modul befogadására képes.

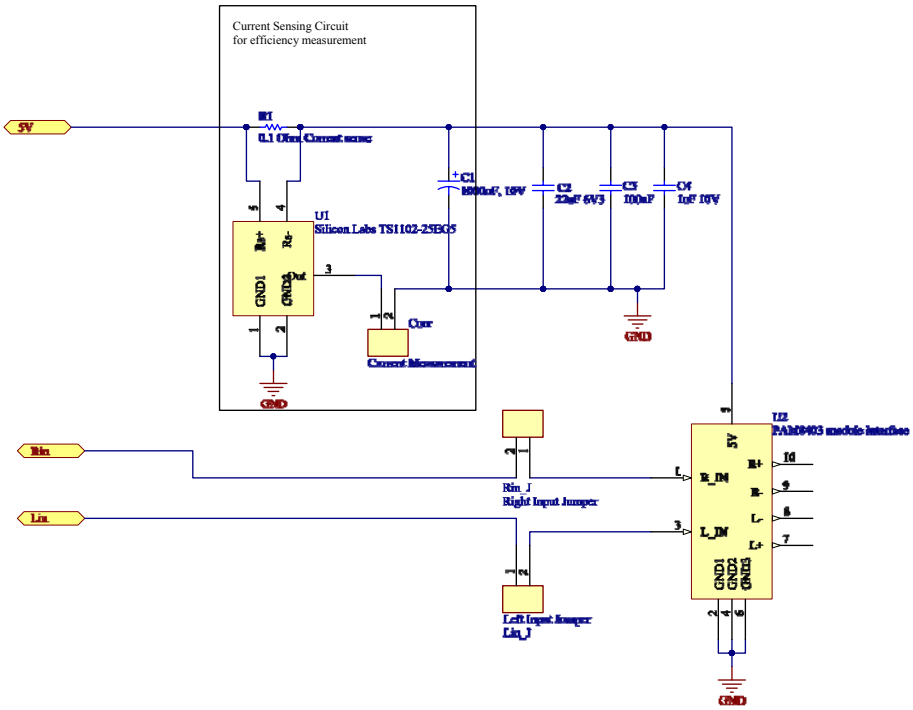


***1.3.ábra**. Teszt környezet*

1.4. Tesztkártya

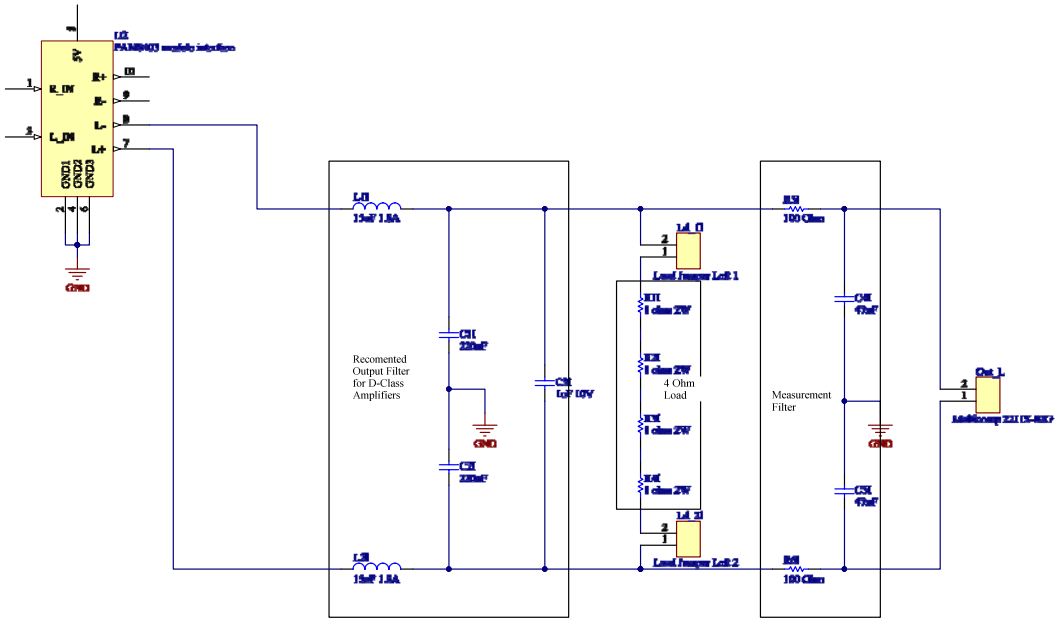
A teszt NYÁK feladata, hogy előállítsa a PAM8403 modulunk számára a tesztkörnyezetet. A teszt NYÁK 4 darab PAM8403 modul fogadására képes slotot tartalmaz. A PAM8403 modulokat kiegészítő áramkörök azonosak mind a 4 teszt slotban.

Az tesztáramkör tartalmaz a teljesítményerősítő modulok előtt egy jelentős tápszűrést, valamint egy árammérő kapcsolást a hatásfok mérésekhez (1.4. ábra).



***1.4. ábra**. PAM8403 audio amplifier modul teszt áramkör, bemeneti rész*

A tesztáramkör szintén tartalmazz egy 4 ohmos terhelést és a D osztályú erősítők számára ajánlott LC kimeneti szűrőt (1.5. ábra).



***1.5. ábra**. PAM8403 audio amplifier modul tesztáramkör, terhelési oldal (jobb csatorna ugyanez)*

A teljesítményerősítő kimenete a mérési pontok előtt még egy aluláteresztő átlapolódásgátló szűrőn (1.5. ábra) is áthalad.

A tesztkártyán jumperek segítségével megszakítható az egyes slotok felé menő mindkét gerjesztőjel, illetve szintén jumperek segítségével leválaszthatóak a 4 ohmos műterhelések is.

A tesztkártya és az NI DAQ mérésadatgyűjtő kártya bekötési összerendelését az 1.1. táblázat tartalmazza.

NI USB-6351	Teszt panel jel
Ao0	Rin
Ao1	Lin
AI 0+	Mod1 L out +
AI 0-	Mod1 L out -
AI 4+	Mod1 R out +
AI 4-	Mod1 R out -
P0.0	Mod1 Green LED
P0.1	Mod1 Red LED
AI 1+	Mod2 L out +
AI 1-	Mod2 L out -
AI 5+	Mod2 R out +
AI 5-	Mod2 R out -
P0.2	Mod2 Green LED
P0.3	Mod2 Red LED
AI 2+	Mod3 L out +
AI 2-	Mod3 L out -
AI 6+	Mod3 R out +
AI 6-	Mod3 R out -
P0.4	Mod3 Green LED
P0.5	Mod3 Red LED
AI 3+	Mod4 L out +
AI 3-	Mod4 L out -
AI 7+	Mod4 R out +
AI 7-	Mod4 R out -
P0.6	Mod4 Green LED
P0.7	Mod4 Red LED

***1.1. táblázat**. Az NI USB-6351 kártya és a tesztpanel összeköttetései*

1.5. Mérési feladatok

Az automatikus tesztrendszer működésének demonstrálásához az alábbi feladatokat kell elvégeznünk:

1. A teszt kártyát ellátó labortáp automatikus kezelése
2. A mérésekhez tartozó paraméterek és limitek ***Limits_data.xls*** file-ból való betöltése:
    a. **Frek_Values[0..18]**: Mérési frekvenciák
    b. **Frek_THD_Limit[0..18]**: a harmonikus torzítás felső határa a frekvencia függvényében
    c. **Frek_Gain_Limit_L[0..18]**: az erősítési limitek alsó határa
    d. **Frek_Gain_Limit_H[0..18]**: az erősítési limitek felső határa
    e. Ampl_Values[0..11]: Mérési amplitúdók
    f. **Ampl_THD_Limit[0..11]:** a harmonikus torzítás felső határa a kimenő teljesítmény függvényében

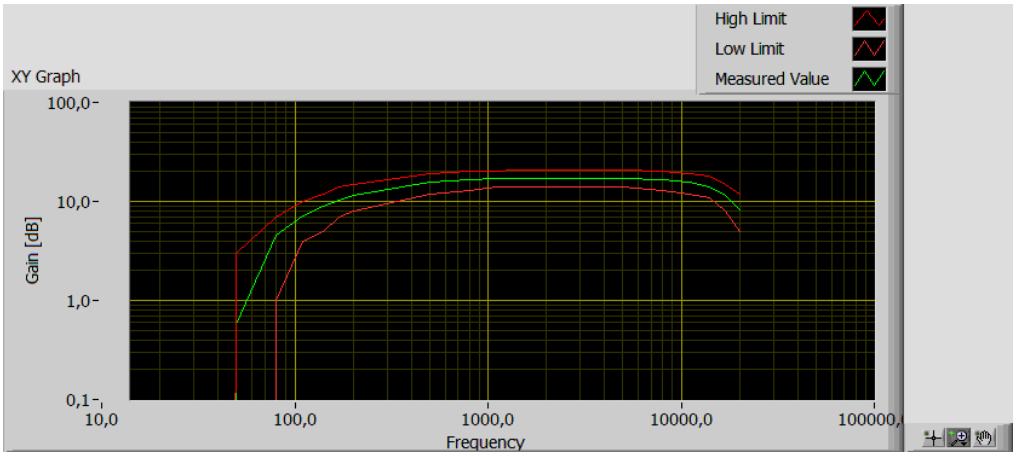
3. A tesztkártya 4 slotjában lévő modulok legalább egy csatornájára az **erősítési karakterisztika** mérése **a frekvencia függvényében**
    a. A gerjesztő szinuszjel amplitúdója legyen 0,3V

4. A teszt kártya 4 slot-jában lévő modulok legalább egy csatornájára a **harmonikus torzítás** mérése a **frekvencia függvényében**
    a. A gerjesztő szinuszjel amplitúdója legyen 0,3V

5. A teszt kártya 4 slotjában lévő modulok legalább egy csatornájára a **harmonikus torzítás** mérése a **kimeneti teljesítmény függvényében**
    a. A gerjesztő szinuszjel frekvenciája legyen 1000 Hz

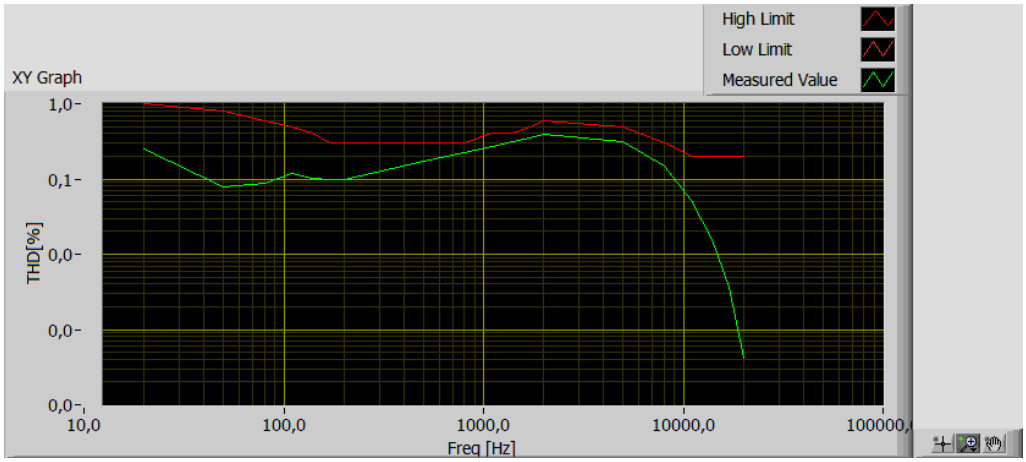
6. Az egyes slotok állapotát a slothoz tartózó LED-en jelezze vissza
    a. Zöld: sikeres teszt
    b. Piros: sikertelen teszt
    c. Sárga: tesztelés folyamatban

7. Jegyzőkönyv generálása mindegyik tesztelt modulhoz külön **xml** formátumú file-ként. A jegyzőkönyv tartalmazza:
    a. A mért értékeket
    b. Az erősítési értékek grafikonját a frekvencia függvényében a limitekkel együtt (1.6. ábra)



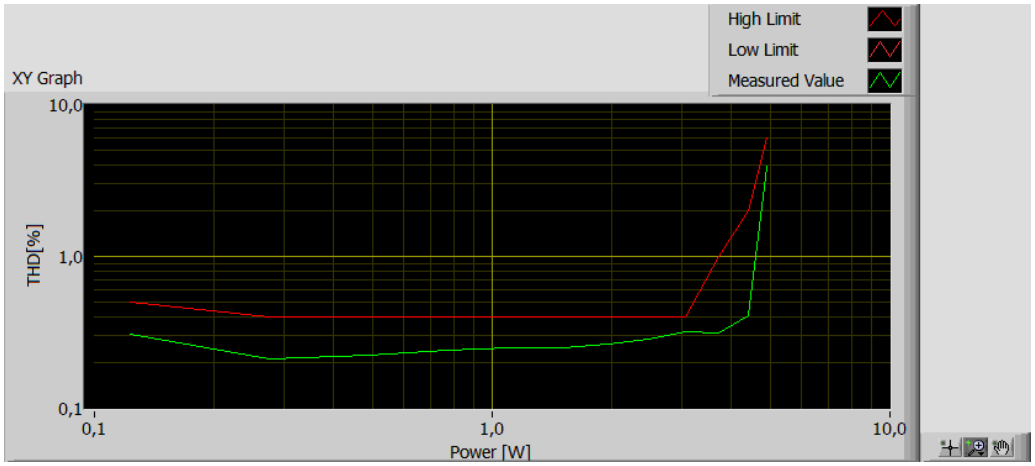
***1.6. ábra**. Minta a jegyzőkönyv erősítési ábrájához*

    c. A harmonikus torzítás grafikonját a limit értékekkel együtt a frekvencia függvényében (1.7. ábra)



***1.7. ábra**. Minta a jegyzőkönyv THD a frekvencia függvényében ábrájához*

d. A harmonikus torzítás grafikonját a limit értékekkel együtt a kimeneti teljesítmény függvényében (1.8. ábra)



1.8. ábra. *Minta a jegyzőkönyv THD a kimenő teljesítmény függvényében ábrájához*

2. Az alkalmazott műszerek kezelésének ismertetése

A műszerek kezelését – bár más megoldások is elképzelhetőek lennének, de ez talán a legpraktikusabb – a NI LabVIEW környezetből fogjuk elvégezni. Ez a fejezet bemutatja, hogy LabVIEW-ból hogyan lehet vezérelni az alkalmazott eszközöket.

2.1. Manson SDP2405 programozható tápegység

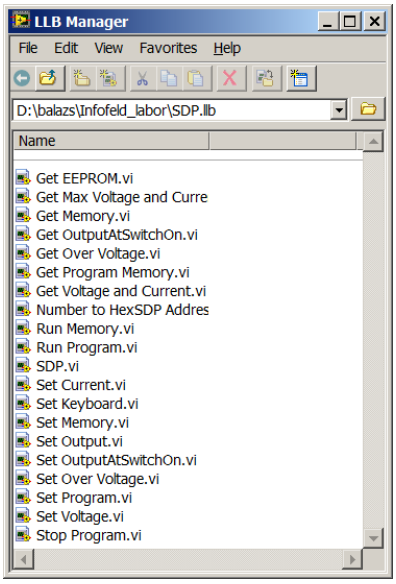
A Manson SDP2405 programozható tápegység (2.1. ábra) egy kapcsoló üzemű jó ár/érték arányt képviselő labortáp. Bár zajossága miatt nem a legideálisabb választás audio erősítők táplálására, de ezt a hátrányát a mi szituációkban a távvezérelhetősége kompenzálja.

2.1. ábra. *Manson SDP2405 tápegység*

A tápegység a dokumentációjának ***A*** jelű függelékében [8] megadott egyszerű karakteres protokoll parancsok segítségével távvezérelhető Rs232 vagy Rs485-ös kapcsolaton keresztül (a kommunikációs interfész módot a manuális kezelőfelületen ki kell választani). Ezt az egyszerű karakteres vezérlést LabVIEW-ban a Serial VISA blokkok segítségével könnyen meg lehetne valósítani, de szerencsére erre nincs semmi szükség, mert a gyártó már

elkészítette ezeket helyettünk [9] (azért is célszerű a LabVIEW alapú műszervezérlés választása, mert a legtöbb gyártó nyújt hozzá támogatást).

A gyártó honlapjáról letölthető program egy komplett távoli felhasználó felület, de ebből számunkra csak a tényleges kommunikációt, vezérlést megvalósító részek lényegesek a feladat szempontjából. Ezek a programkódok a *SDP.llb* LabVIEW VI könyvtárban találhatóak (*SDP.CD.Rev1.3(120904)\LabVIEW Driver\SDP.llb*) a letöltött program többi részére nincs is szükség.



***2.2. ábra**. A Manson SDP LabVIEW driver könyvtárának elemei*

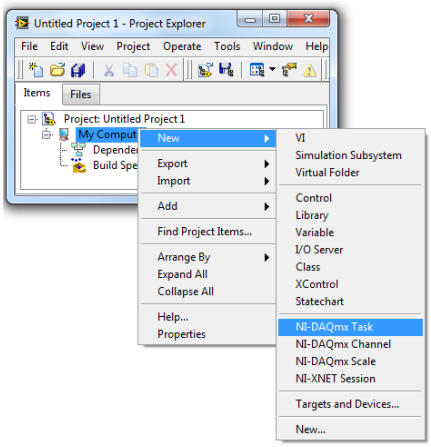
Az *SDP.llb* könyvtárból (2.2. ábra) alapvetően a *Set Current.vi*, *Set Output.vi*, *Set Voltage.vi* részekre lesz szükségünk. Ezek felhasználása a következőképpen történhet: vagy megnyitva őket ***save as*** paranccsal önálló néven elmentjük őket egyedi VI-ként (ez picit barbár megoldás), vagy picit kulturáltabban egy projectbe az ***add*** funkció segítségével hozzáadjuk a szükséges VI-kat (ne az egész llb-t). Az így kiválasztott VI-k mint Sub-VI-k felhasználhatóak saját vezérlő programjainkban. Ez a javasolt eljárás, mert projectre úgy is szükségünk lesz.

Az *SDP.llb* könyvtárat megtaláljuk a méréshez tartozó alapkönyvtárban.

2.2. NI USB-6351 DAQ kártya

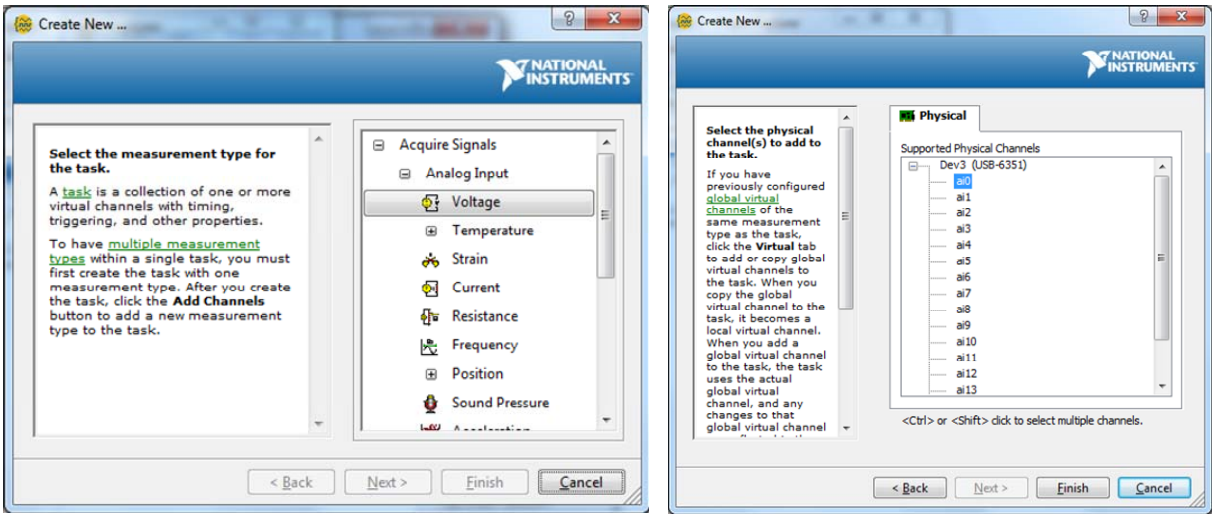
Az NI mérésadatgyűjtő kártyáit az ún. DAQmx taszkok segítségével lehet kezelni. A DAQmx taszkok segítségével szinte azonos módon tudunk analóg és digitális ki és bemeneteket kezelni, akár mintánként akár bufferelt módon.

A DAQmx taszkok létrehozására a legegyszerűbb megoldás, ha LabVIEW projectet használunk, és ott egy varázsló segítségével hozzuk létre őket (2.3. ábra).



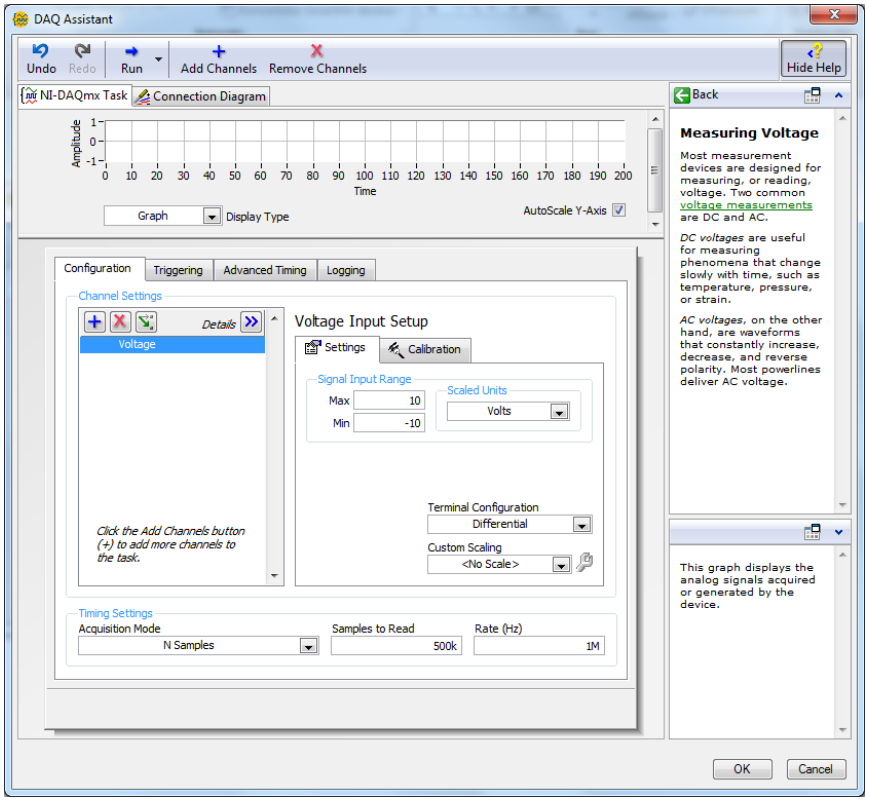
2.3. ábra. *DAQmx taszk létrehozása LabVIEW project-nél*

Mintának egy a mérés során is használható analóg bemeneti DAQmx csatornát fogunk létrehozni. Kiválasztjuk az ***Acquire Signals / Analog Input / Voltage*** opciót (2.4. ábra). majd megadjuk az általunk használt bemeneti analóg csatornát, pl ***ai0***-t.



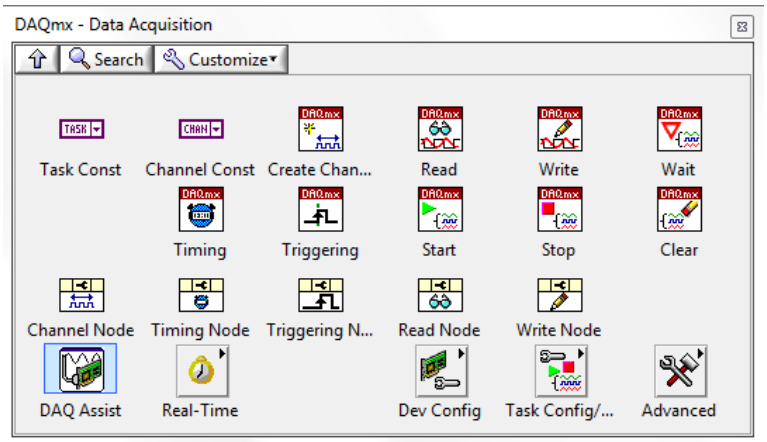
2.4.ábra. *DAQmx bemeneti taszk konfigurálása*

A következő ablakban meg kell adni a DAQmx taszk elnevezését, majd ezután pontosan meg kell adni a csatorna beállításait (2.5. ábra).



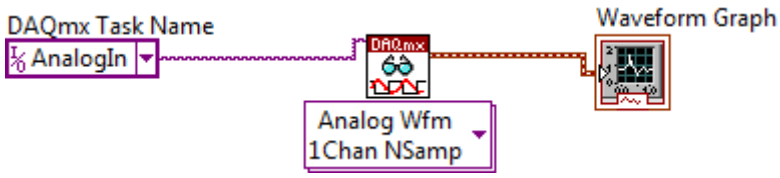
***2.5.ábra**. DAQmx taszk részletes beállítása*

Az így létrejött DAQmx taszk áthúzva a project bármelyik VI-jába felhasználhatóvá válik. A DAQmx taszk felhasználásához tartozó alap VI-kat a ***Measurement I/O / NI-DAQmx*** palettán találjuk (2.6. ábra). Szerencsére általában a rendelkezésre álló VI készlet csak kevés elemét kell ténylegesen felhasználnunk.



***2.6. ábra**. A DAQmx VI paletta*

Az adatok generálására például általában elég, ha csak a Write VI jól konfigurált verzióját használjuk. A VI a mérési taszkot önállóan elindítja és lezárja (akkor lehet szükség plusz VI-kra, ha biztosítani akarjuk a folyamatos lejátszást, vagy meg akarjuk várni, hogy egy hullámforma kijátszása befejeződjön). Olvasáskor is általában elég pusztán a Read VI használata (2.7. ábra).



***2.7. ábra**. DAQmx taszk bemenet felhasználása*

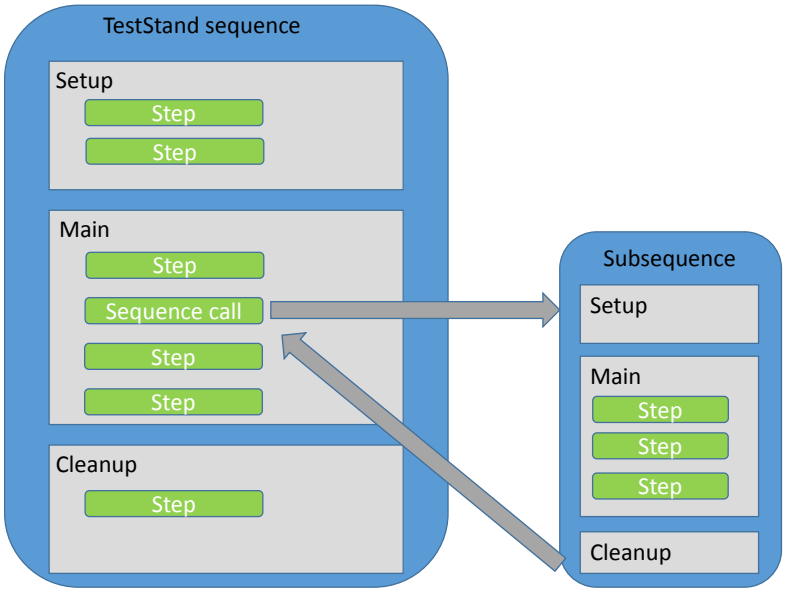
Érdeklődők a DAQmx modul felhasználásáról rengeteg példát találhatnak a LAbVIEW ***Help / Find Examples / Hardware Input and Output / DAQmx*** menüpont alatt.

3. TestStand alapok

A TestStand [1] a National Instruments automatikus tesztek fejlesztéséhez létrehozott környezete, ami különböző nyelvekben megírt teszt lépések integrálását, kezelését teszi lehetővé. Ez a leírás egy alap bevezető a TestStand használatába.

3.1. TestStand alapfogalmak

A TestStand-ben használt alapvető tesztlépések a **Step**-ek, amelyek tesztszekvenciákba **Sequences**-ekbe rendezhetőek. Mindegyik **Sequences** három funkció szerint elkülönülő lépéscsoportot **Step Groups** tartalmaz. Ezek a ***Setup, Main*** és ***Cleanup,*** amelyek nevüknek megfelelően a teszt környezet előkészítésének, a tesztek végrehajtásának, és a teszt környezet lezárásának lépéseit foglalják egységbe. A szekvenciák természetesen tartalmazhatnak alszekvenciákat **Subsequences,** (3.1 ábra) így lehetővé téve hierarchikus tesztek létrehozását.

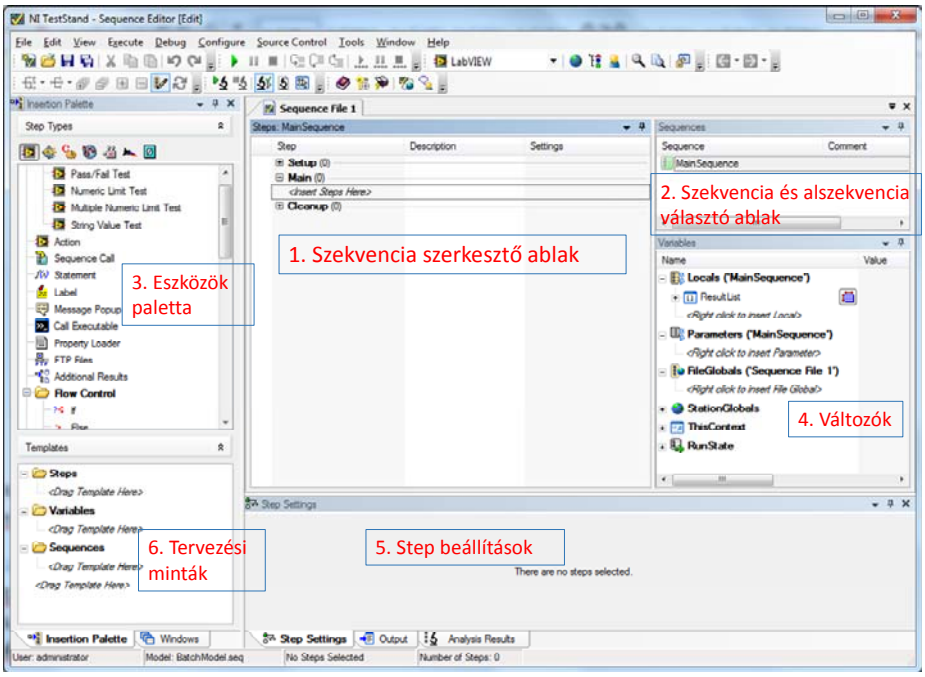


3.1. ábra. *Egy automatikus teszt felépítése TestStand-ben*

Az így összeállított tesztlépésekből és, szekvenciákból valósul meg az automatikus teszt, amit utána a TestStand lefuttat és dokumentál.

3.2. TestStand felhasználói felület

A TestStand elindulása után egy login dialógus ablakkal fogad minket, mert éles esetben különböző privilégiumok adhatóak meg a felhasználó csoportok számára. ***A default adminisztrátori password az üresen hagyott mező!***



3.2. ábra. *A TestStand felhasználói felülete*

A TestStand szerkesztőfelülete (3.2 ábra) a következő főbb részekből áll:

1. **Szekvenciaszerkesztő ablak**: Ebben a részben tudjuk összeállítani a tesztünk lépéseinek végrehajtási sorrendjét, vezérlési szerkezetét.
2. **Szekvencia- és alszekvencia- választó ablak:** Ebben az ablakban tudunk a teszt egyes szekvenciái, alszekvenciái között váltani
3. **Eszközök paletta:** Az itt lévő komponensekből tudjuk összeállítani a 1. Szekvencia szerkesztő ablakban a tesztet. A tesztünket egyszerűen a kívánt komponens megfelelő helyre történő behúzásával tudjuk bővíteni.
4. **Változók:** A teszt végrehajtása során felhasznált változókat, paramétereket, és a létrejövő teszteredményeket összefoglaló ablak.
5. **Step beállítások:** A kiválasztott tesztlépés beállításai. Ebben a részben lehet a tesztet végrehajtó modult specifikálni, annak működési módját pontosan meghatározni.
6. **Tervezési minták:** Ide lehet elhelyezni tipikus, általunk sokszor használt megoldásokat. Számunkra nem lényeges.

3.3. Teszttípusok, kódmodulok

A ***3. Eszközök paletta (Insertion paletta)*** komponenseinek segítségével tudjuk a tesztünk egyes lépéseit létrehozni. Az ablak tetején található az ún. ***module adapter*** kiválasztó paletta (3.3. ábra). Ennek segítségével tudjuk megadni, hogy az adott beillesztendő tesztlépéseket milyen nyelven valósítjuk meg. Az alapértelmezett adapter a LabVIEW, ami számunkra megfelelő, de szükség esetén lehetne más nyelven létrehozott tesztlépéseket is integrálni a TestStand-be.

3.3. ábra. A Module adapter kiválasztó ablak

A ***3. Eszközök paletta (Insertion paletta)*** a következő számunkra fontos komponenseket tartalmazza

1) **Tesztek (Tests)**: ezek az UUT (Unit Under Test) ellenőrzésére szolgáló lépések. Eredményeik belekerülnek a teszt jegyzőkönyvbe.
   a) **Pass/Fail Test**: Egy egyszerű tesztlépés, ami egy boolean eredményt ad vissza annak függvényében, hogy a teszt sikeres volt, vagy nem.
   b) Numeric Limit Test: A tesztlépés egy számmal tér vissza, amelynek a tesztlépés konfigurációjánál megadott limitek közé kell esnie.
   c) Multiple Numeric Limit Test: Hasonló az előző határérték teszthez, csak itt a teszt egy számtömbbel tér vissza, amelynek az értékei a limit tömb értékei közé kell, hogy essenek.
   d) String Value Test: A tesztlépés egy string értékkel tér vissza, amit összehasonlít a rendszer egy megadott referencia stringgel.

2) **Akciók, (Action)**: Szabadon felhasználható kód modul. Tipikusan olyan funkcióra használják, amely nem közvetlenül az UUT tesztelésére irányul. Például klímakamra vezérlése, tápegység feszültségszintjének beállítása stb.

3) Szekvenciahívás (Sequence Call): ennek a modulnak a segítségével lehet alszekvenciákat meghívni. Azon kívül, hogy segít modulárisabbá tenni a kódot, lehet a segítségével párhuzamos végrehajtást kezdeményezni, ami számunkra is előnyös lesz.

4) **Statement**: Képletszerkesztésre van benne lehetőségünk, amelynek segítségével a TestStand környezeti változók értékeit meg tudjuk változtatni. Pl. egy korábbi tesztlépés eredményeiből ki tudunk számolni valamilyen paramétert.

5) Property Loader: File-ból tud hatékonyan teszt property-ket betölteni. Tipikusan bonyolultabb teszteknél a limit értékek automatikus kezelésének előkészítésére használják. Nekünk is szükségünk lesz rá.

6) Flow control: Ezek a blokkok a hagyományos szekvenciális programozásban alkalmazott vezérlési szerkezeteket tartalmazzák.

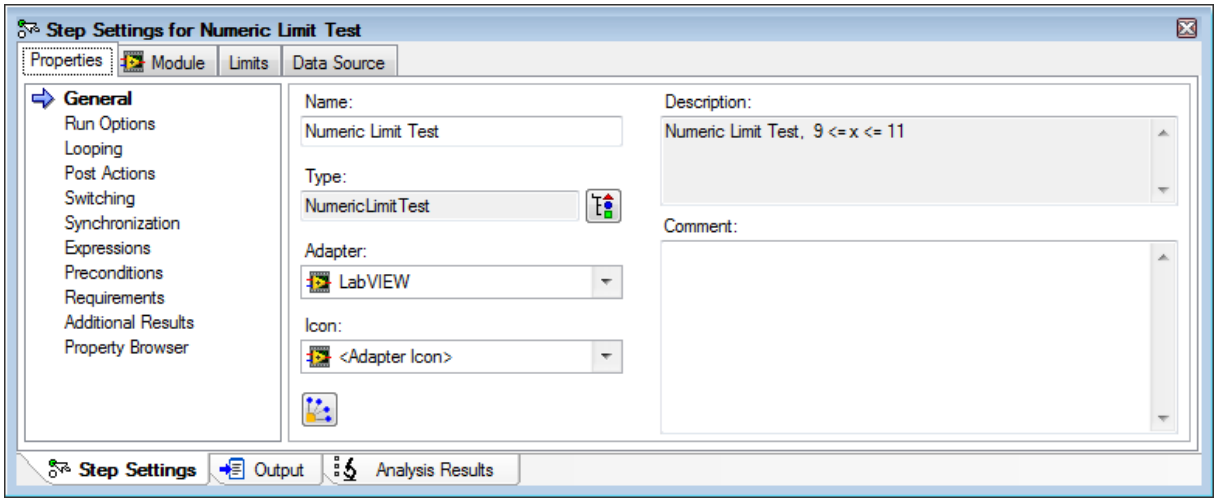
7) Syncronisation: Párhuzamos teszt végrehajtásnál alkalmazható szinkronizációs objektumok

A többi eszköz használata vagy magától értetődő, vagy nincs rá szükségünk.

3.4. Egy egyszerű teszt létrehozása

Az alap tesztlépések alkalmazása szinte független a tényleges tesztlépés típusától, ezért egy közepesen bonyolult teszt a **Numeric Limit Test** alkalmazását mutatjuk be.

Első lépésben húzzuk át a ***3. Eszközök paletta (Insertion paletta)*** ablakból a **Numeric Limit Test** ikont Numeric Limit Test az ***1. Szekvenciaszerkesztő*** ablak megfelelő helyére húzzuk (tipikusan a main group-ba). Ennek hatására, mivel alapértelmezetten ez a lépés lesz kiválasztva az *5.* ***Step beállítások*** ablakban megjelennek a teszt tulajdonságai és beállítási lehetőségei.



3.4. ábra *Test Property ablak*

3.4.1. Tesztek tulajdonságai (Properties) fül

Ebben a fülben (3.4. ábra) lehetőségünk van a **General** opció alatt átnevezni a tesztet, illetve leírást fűzni hozzá (érdemes ezt is átírni, mert beállításfüggően megjelenhet a teszt jegyzőkönyvben). A tesztlépéssel kapcsolatban számtalan tulajdonságot be tudunk állítani, ezek közül egyedül a ***Syncronisation*** opciót fogjuk használni, amely a teszt párhuzamos futásával kapcsolatos beállításokat tartalmazza, de azt is majd csak később.

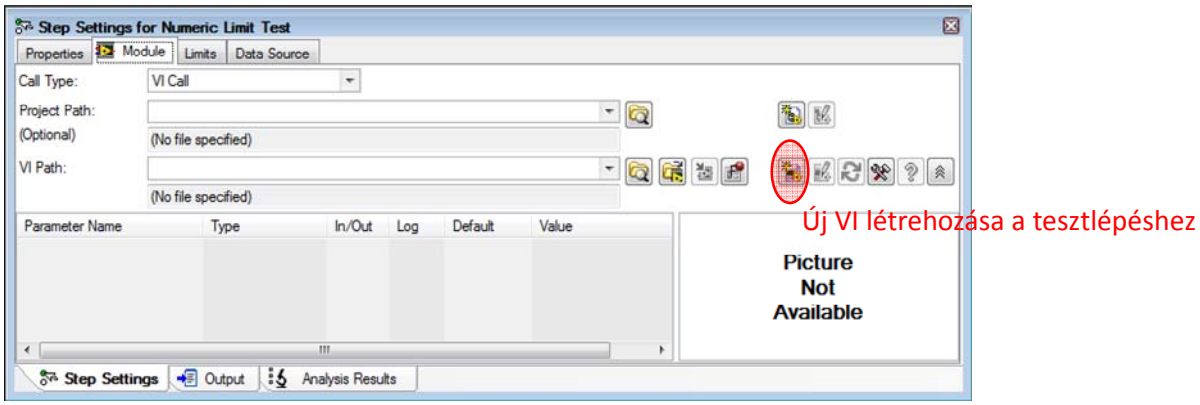
Általunk nem használt opciók: A ***Run*** opcióban különböző futási beállításokat specifikálhatunk, amelyek számunkra nem lényegesek. A ***Looping*** *opcióban* meg tudjuk határozni, hogy az adott tesztet hányszor hajtsa végre a rendszer ugyanazon az UUT-n (tipikusan statisztikai tesztekre használt, ahol pl. 5-ből 4 helyes válasz elég). Mi nem alkalmazzuk ezt az opciót. ***Post Action***-ban be tudjuk állítani, hogy mi történjen a teszt végrehajtása után. Alapértelmezetten továbblépünk a következő lépésre, ami számunkra megfelelő lesz. A ***Switching*** részben külső switch modult tudunk vezérelni, amely segítségével hardware csatornák között tudunk váltani. Nem használjuk. Az **Expression** részben a TestStand környezeti változói és a tesztlépés bemenetei, kimenetei között teremthetünk kapcsolatot kifejezések létrehozásával. ***Precondition*** résznél a teszt végrehajtásához tudunk feltételeket tenni. Az ***Additional Results*** opciónál a teszt által a Limit ellenőrzésre szánt eredményeken kívül létrehozott más eredményeket lehet megadni. A ***Property Browserben*** a teszt tulajdonságait, paramétereit, bemeneteit tudjuk áttekinteni.

3.4.2. Module fül

Ebben a fülben tudjuk specifikálni a tesztet végrehajtó kódmodult. Mivel mi LabVIEW adaptert választottunk, ezért itt LabVIEW kódmodulról lesz szó. A Module fül lehetővé teszi számunkra, hogy létrehozzuk a tesztet végrehajtó VI-t, projectet, vagy egy meglévő VI-t, projectet linkeljünk be. Alapvetően elég a kódot végrehajtó VI-t megadni, de azokban az esetekben, ahol szükséges DAQmx csatorna alapú hardware kezelést is használnunk, a LabVIEW projectet is specifikálnunk kell.

Arra vigyázzunk, hogy meglévő VI-k esetében a connector panelra rakott, kívülről is látható kimeneteket, bemeneteket össze kell rendelni a Value oszlopban konstansokkal, a teszt lokális vagy rendszerváltozóival. Automatikusan létrehozott VI-k esetében az alap ki- és bemenetek

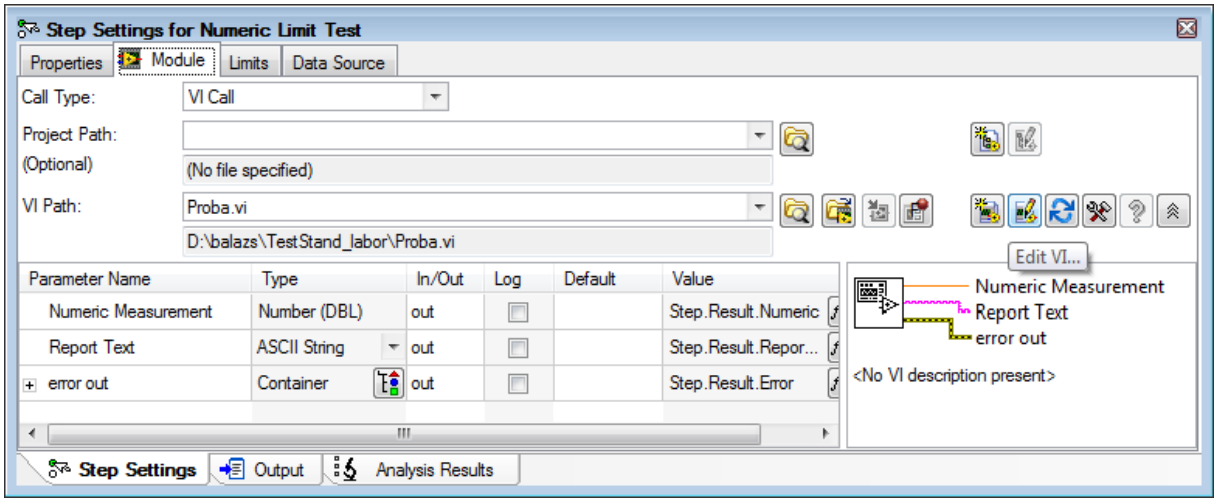
automatikusan kezelődnek, de nagy valószínűséggel szükségünk lesz saját be- és kimenetekre, amelyek összerendelését a TestStand környezettel szintén a Value oszlopban kell megadni. A példánkban egy új a tesztet végrehajtó VI-t hozunk létre (3.5. ábra).



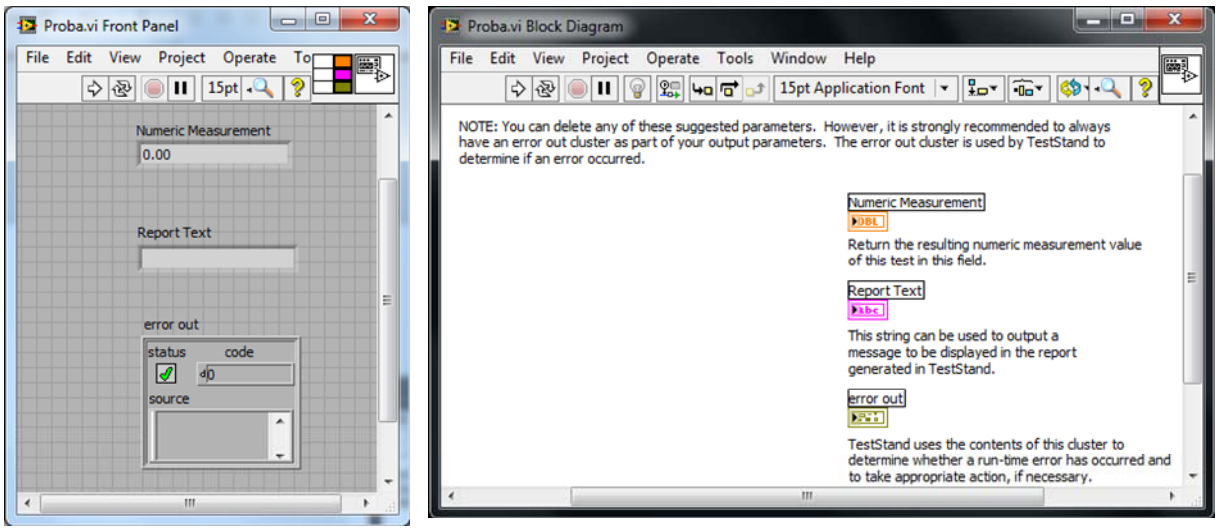
3.5. ábra *Test Module ablak, új VI létrehozása*

A VI létrehozása után lehetőségünk van ebből a vezérlő ablakból szerkeszteni azt, illetve látható (3.6. ábra, 3.7. ábra), hogy a létrehozott VI connector plane-jére helyezett bemenetei és kimenetei megjelennek a module ablakban.

Numeric Limit test esetében az alapértelmezett kimenet ***Numeric Measurement***, amelynek a limitek közé esését ellenőrizzük. A Value oszlopban ez a kimenet van a Step.Result.Numeric tulajdonsághoz rendelve, amely a teszt alapértelmezett kimeneti eredménye. Ehhez a mérési eredményhez járul még hozzá a Report Text, amely a teszt jegyzőkönyvben a méréshez fűzött kommentként jelenik meg, illetve az error out, amely a VI végrehajtása közbeni hibákat jelzi (meg kell tudni különböztetni az UUT hibáját az esetleges tesztelő eszköz hibájától. Ez utóbbi kimenet tipikusan akkor aktív, ha például az egyik tesztelő műszer hibát dob).



3.6. ábra *Test Module ablak, létrehozott VI konfigurálása*



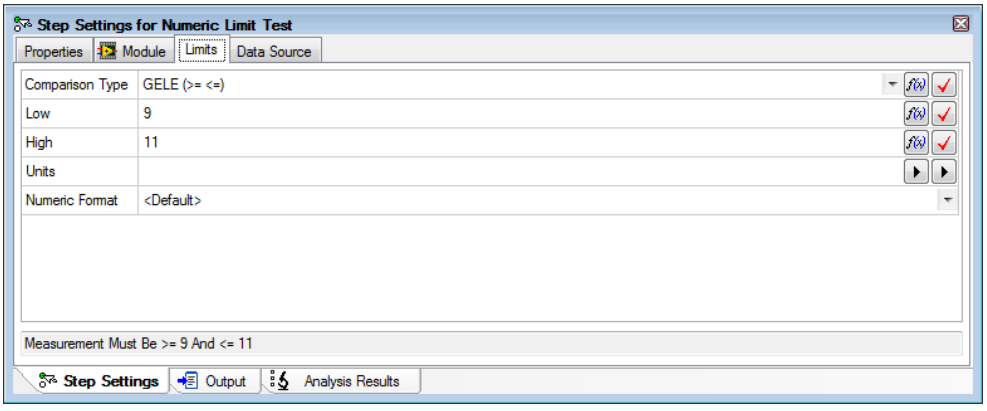
3.7. ábra *Az automatikusan létrehozott Numeric Limit típusú teszt VI*

Szükség szerint az alapértelmezett bemeneteken, kimeneteken túl újabbakat is létrehozhatunk, ezeket a VI front paneljének connector plane-jére kössük rá, és frissítés után látszódni fognak a TestStand-ben is.

Fontos, hogy ebben a module fül-ben állítható be az is, hogy a tesztelés során a tesztet végrehajtó VI front panelje megjelenjen vagy ne (*Show VI front panel ikon* a *Create, Edit* VI ikonokkal egy sorban). A tesztlépéshez tartozó VI front paneljének megjelenítése sok esetben megkönnyíti a tesztfejlesztését.

3.4.3. Limit fül

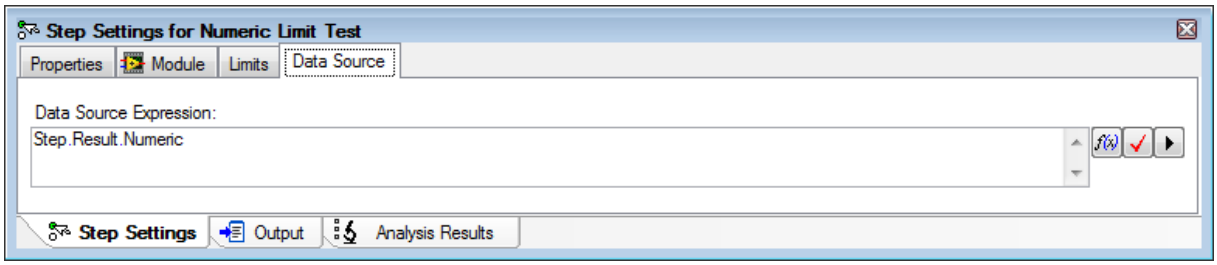
Itt tudjuk a tesztlépés limit értékeit meghatározni. Ez a legegyszerűbb esetben abból áll, hogy egyszerűen kézzel beírjuk a limit értékeket (3.8. ábra), de a gyakorlatban, sok esetben a limiteket valamilyen file-ban tárolják, és onnan kell előhívni őket, majd az egyes lépéseknél változó értékként átadni azokat (lásd később).



3.8. ábra *A Numeric Limit tesztlépés Limits füle*

3.4.4. Data Source fül

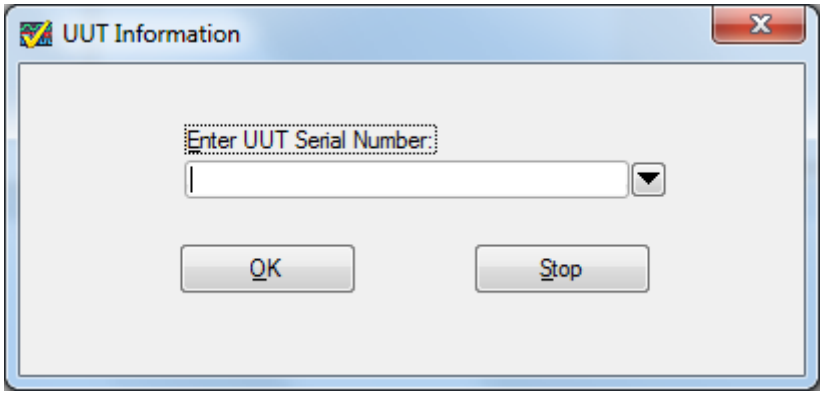
Itt adjuk meg, hogy a Limit ellenőrzés milyen adaton történik. Ebben az esetben ez a Numeric Limit teszt Step.Result.Numeric értékén történik meg (3.9 ábra), ami az alapértelmezett forrás.



3.9. ábra *Data Source fül*

3.5. Tesztek futtatása szekvenciális modellnél

A teszteket a TestStand felső ikonsorának Play ikonjával le tudjuk futtatni. A TestStand alapbeállítása az ún. szekvenciális modell, ami azt jelenti, hogy az egyes tesztelendő eszközöket egymás után sorrendben kezdi el letesztelni. Ez azt jelenti, hogy a teszt elindítása után a rendszer megjelenít egy ablakot, amely az adott UUT sorozatszámát kéri, és miután ezt megadtuk, a rendszer végrehajtja az általunk specifikált teszteket. A tesztelés leállítása úgy lehetséges, hogy az UUT információs ablakban (3.9. ábra) a stop gombra nyomunk, ilyenkor létrejön a teszt report a beállított formátumban (3.10. ábra).



3.9. ábra Teszt futtatása: az UUT azonosítása

UUT Report

Station ID	SZLAB21
Serial Number	1
Date	Monday, February 08, 2016
Time	4:13:40 PM
Operator	administrator
Execution Time	0.0955081 seconds
Number of Results	1
UUT Result	Passed

Begin Sequence: MainSequence
D:\balazs\TestStand_labor\Sequence File 1.seq

Step	Status	Measurement	Units	Limits		
				Low Limit	High Limit	Comparison Type
Numeric Limit Test	Passed	10		9	11	GELE(>= <=)

3.10. ábra Teszt futtatása: teszteredmények

A debuggolás könnyítésére lehetőségünk van Breakpointokat lerakni az egyes teszt lépésekhez, vagy kihagyni őket, esetleg az eredménytől függetlenül Pass/Fail állapotba kényszeríteni őket (a tesztlépésen jobb egérgombot nyomva). Természetesen Breakpoint használata után a tesztlépések léptetésére, stepelésre is szokott módon lehetőségünk van.

3.6. Változók és adatok használata a TestStand-ben

Sok tesztlépés, akció igényel valamilyen adatot, paramétert. Ezeknek a megadására tipikusan a TestStand változóit tudjuk felhasználni. A változókat a 4. Variables (Változók) ablakban tudjuk megadni és kezelni. A változók értékének megváltoztatására például az **Expression** kifejezéseket tudjuk használni.

A TestStand sokfajta, különböző láthatóságú, funkciójú változót használ:

1. **Locals:** Az adott szekvenciára lokális láthatóságú változók.
2. **Parameters:** Az adott szekvencia paramétereit tartalmazza.
3. **File Globals:** A Sequence file-ra vonatkozó globális változók.
4. **StationGlobals**: Az adott tesztállomásra vonatkozó globális változók.
5. **ThisContext**: Az adott szekvenciára vonatkozó referencia.
6. **RunState**: A végrehajtás állapotára vonatkozó információk.
7. **Step**: Az adott Stepre vonatkozó változók.

Új változót a megfelelő funkciókörre jobb egérgombbal kattintva lehet felvenni. A változó neve mindig modulárisan a tartalmazott csoportokat is beépítve áll elő pl. ***Locals.i*** egy ***i*** nevű lokális változó teljes neve. A tömbök kezelése megegyezik bármilyen más környezet tömb kezelésével. Nekünk tipikusan **Locals** láthatóságú változókra lesz szükségünk, de néhány rendszerváltozót is használnunk kell majd.

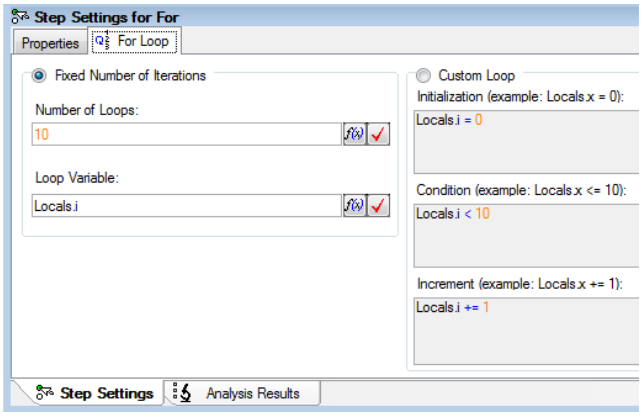
3.7. Változók használata for ciklus létrehozásánál

A mérés során sokat kell majd **for ciklust** alkalmaznunk. A for ciklus létrehozásának a következőek a lépései:

1. **Variables (Változók)** ablakban Jobb gomb, ***Insert Local*** segítségével létrehozunk egy ciklusváltozót a **Locals** részhez.

2. Behúzzuk a bal oldali **Eszközök** palettából a kívánt helyre a **for ciklust** és a megjelenő **END** lezárást is a megfelelő helyre tesszük.

3. A **For ciklus** beállításainál (alsó **Step Settings** ablak) specifikáljuk a ciklus viselkedését Expressions-ok segítségével (az f(x) ikonra kattintva gyakorlatilag interaktív módon választhatjuk ki a rendszer változóit és szerkeszthetjük meg az alkalmazni kívánt kifejezést) (3.11 ábra).



3.11. ábra. For ciklus konfigurálása

3.8. Változók értékének file-ból való betöltése és felhasználása limitként

Valós tesztkörnyezeteknél a legtöbb esetben a tesztlépések limit értékeit nem egyesével szokás megadni, hanem adatbázisból, vagy file-ból olvassuk fel. Ez azért hasznos módszer, mert sok esetben változik a tesztelt termék típusa és tulajdonságai, de maguk a tesztlépések ugyanazok maradnak. Tehát egy új termék verzió esetében a tesztszekvenciát nem kell átírnunk, csak az alkalmazott limiteket kell lecserélnünk.

A TestStand tartalmaz egy ún. **Property Loader** eszközt, amit erre a feladatra fejlesztettek ki. Ez az eszköz képes arra, hogy automatikusan megváltoztassa bizonyos tesztlépések limit értékeit egy file-ban tárolt konfiguráció alapján. Sajnos a mi alkalmazásunkban ezeket a tulajdonságokat nem tudjuk teljesen kihasználni, mert az esetek többségében arra lesz szükségünk, hogy egy ciklusban különböző frekvencia- vagy amplitúdóértékekkel végezzük el ugyanazokat a tesztlépéseket. Ezt a **Property Loader** pedig nem tudja automatikusan kezelni. Ettől függetlenül a **Property Loader**-t fel tudjuk arra használni, hogy egy file-ból beolvassunk értékeket lokális változókba, amelyeket utána limitértékekként fel tudunk használni.

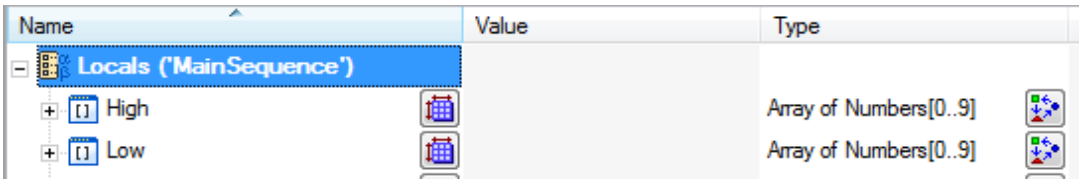
Ennek használatára egy egyszerű példa: *adott egy* ***for*** *ciklusban egy* ***Numeric Limit*** *teszt, ennek a tesztlépésnek a limitjeit szeretnénk felolvasni a property loader segítségével, majd alkalmazni azokat. A Limits.xls file-ban a High és Low értékek tartalmazzák az egyes iterációkban ellenőrizendő limitértékeket (az xls file formátuma elég sajátos) (3.12 ábra).*

START	
<Step Name>	
<Locals>	Variable Value
High	<Prop Name='High' Type='Array' LBound='[0]' HBound='[9]' ElementType='Number'><Value ID='[0]'>0</Value><Value ID='[1]'>1</V
Low	<Prop Name='Low' Type='Array' LBound='[0]' HBound='[9]' ElementType='Number'><Value ID='[0]'>0</Value><Value ID='[1]'>1</V
<FileGlobals>	Variable Value
<StationGlobals>	Variable Value
END	

3.12. ábra. Minta a Limitértékeket tartalmazó Excel file-ból

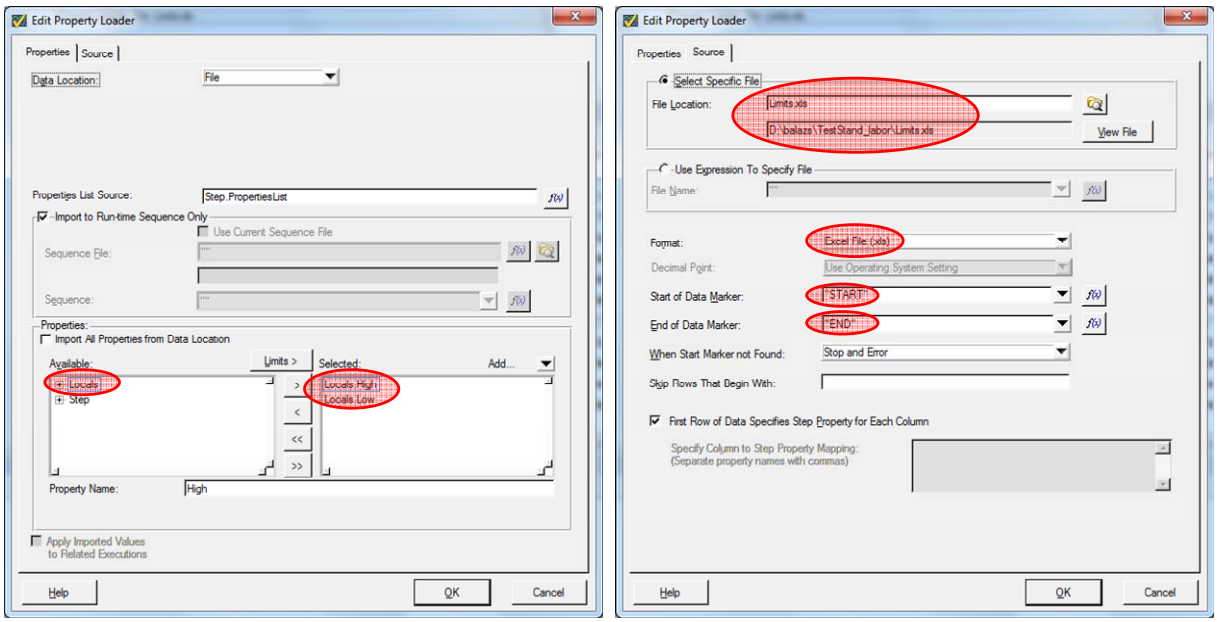
A beolvasáshoz és alkalmazáshoz végrehajtandó feladataink a következőek:

1. **A Variables (Változók)** ablakban Jobb gomb, ***Insert Local*** segítségével létre kell hoznunk két lokális tömböt, a High és Low tömböt. Mindegyiket 10 értékkel (*3.13. ábra*).



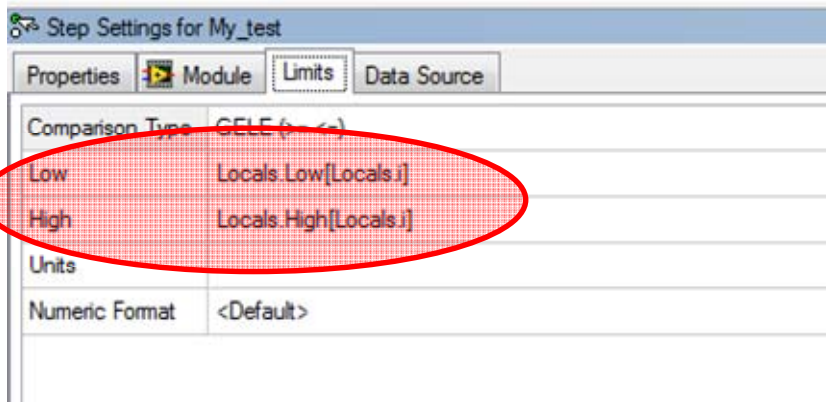
3.13. ábra Létrehozott lokális tömbök

2. Rakjunk le a Setup group-ba egy **Property Loader** dobozt!
3. Állítsuk be a Property loadert úgy, hogy xls-file-ból olvasson be, és adjuk meg neki a beolvasandó paraméterként a két lokális tömböt (3.14. ábra).
    a. Kiválasztva a Source fült, megadjuk a file formátumának az Excelt-t,
    b. Megadjuk a file elérési útvonalát,
    c. Beállítjuk az adatok kezdetét és végét jelző szimbólumot,
    d. A Properties ablakból beállítjuk a Locals változókból a két számunkra szükségest.



3.14. ábra A **Property Loader** konfigurálása

4. A beolvasott limitértékeket felhasználjuk a ***Numeric Limit*** tesztnél, úgy, hogy egyszerűen a konstans számok helyett hivatkozunk a Limit tömb megfelelő elemére (3.15. ábra).



3.15. ábra A beolvasott limitek felhasználása egy tesztlépésben

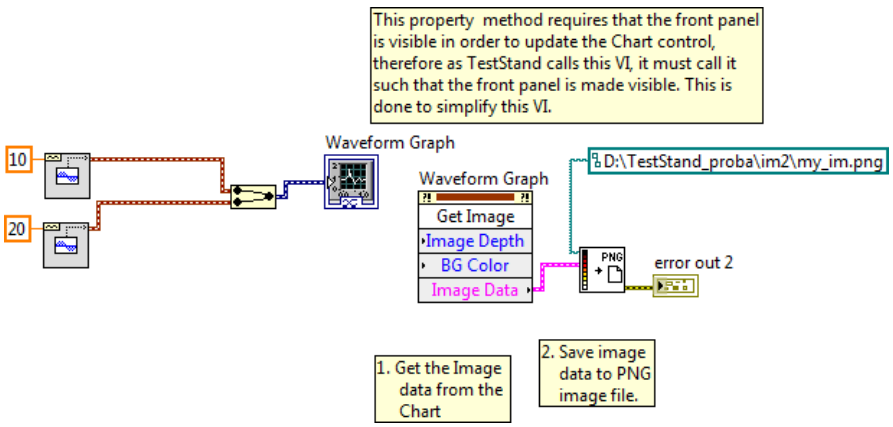
3.9. Képek exportálása a jegyzőkönyvbe

A TestStand, bár képes arra, hogy egyszerű ábrákat generáljon a létrejövő teszt jegyzőkönyve, de bonyolultabb ábrák esetében, mint például egy logaritmikus skálát használó amplitúdókarakterisztika, a felhasználónak kell megoldania a kép beillesztését. Szerencsére ez viszonylag egyszerű feladat:

1. **A Configure/Result Processing** fülben a Report formátumát állítsuk XML alapúra, ha nem azon lenne (csak XML és HTML alapú reportoknál működik a bemutatott módszer).

2. LabVIEW VI segítségével rajzoljuk meg a kívánt ábrát pl. egy XY Graph-ra, vagy Waveform Graph-ra (akár a tesztlépés részeként, akár egy külön **action**ként, ha előre összegyűjtött adatokból dolgozunk).

3. A VI-ból exportáljuk ki az ábrát file-ként.
   a. Jobb gombbal kattintva a Waveform Graph vagy XY Graph graph block diagram megjelenítésére válasszuk a **Create/Invoke Node/Get Image** opciót, ami létrehoz egy dobozt, amelyek egy nyers képadat a kimenete (*3.15. ábra)*.
   b. A nyers képadatot kössük a **Write PNG File** dobozra (Ctrl+space a név szerinti keresés a LabVIEW-ban), majd adjuk meg a leendő file elérési útvonalát.

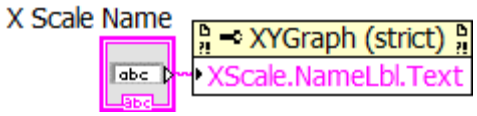


3.16. ábra. Ábra készítése a jegyzőkönyv számára

c. A lépéshez tartozó **Report Text** mezőnek adjuk meg a létrejött .png file elérési útvonalát az alábbi szintaktikával (könyvtárat nem kell feltétlenül megadni. A my_im.png pedig a példa file neve):
<IMG ALT='Data' SRC='D:\TestStand_proba\im2\my_im.png'></IMG>

A képek létrehozásánál sokszor szükségünk lesz arra, hogy a file nevét és az arra való hivatkozást dinamikusan állítsuk össze. Ehhez a LabVIEW stringkezelő függvényei a ***concatenate string*** és a ***string to path*** doboz jelentős segítséget ad.

Szintén sokszor szükség lehet arra, hogy az ábra tengelyeinek feliratát dinamikusan változtassuk. Az ilyen, ábrához tartozó feliratok ún. Property Node-ként érhetőek el a LabVIEW-ban. Például egy XY Graph X tengelyére kiírandó szöveget az XScale.NameLbl.Text property átadásával tudunk megadni.



3.17. ábra. Ábra tengelyének dinamikus feliratozása

A Property Node-okat jobb gombbal a Waveform Graph vagy XY Graph graph block diagram megjelenítésére kattintva a **Create/Property Node/** opcióval tudjuk előhozni. Sajnos elég sok van belőlük, de számunkra csak a tengelyfeliratok a fontosak:

?Scale/Name Label/Text**:** A ***?*** tengely felirata.

A Property Node-ok általában olvasható módban jönnek létre, de ha lerakás után jobb egérgombot nyomunk rajtuk, akkor a ***Change To Write*** paranccsal át tudjuk őket állítani írhatóra.

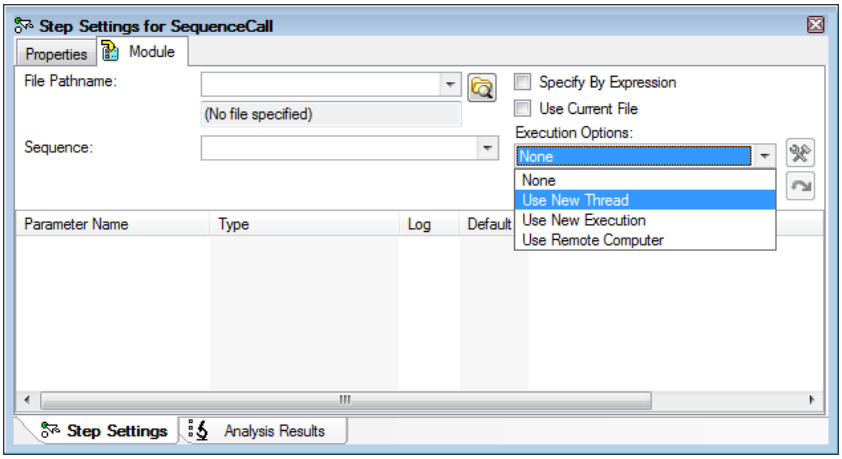
3.10. Párhuzamos tesztelés: feladatok párhuzamos futtatása

Az eddigi fejezetek bemutatták a TestStand legalapvetőbb működését. Eddig a környezet nem nyújtott sokkal többet, mintha egy saját ütemező programot készítettünk volna. Az automatikus tesztrendszerek nagy előnyei pedig leginkább ott mutatkoznak meg, amikor a tesztesetek párhuzamos végrehajtására, vagy több UUT-n történő párhuzamos tesztelésre kezdjük el alkalmazni őket.

A TestStand esetében a legegyszerűbb párhuzamosítási példa, amikor egy feladatot a többi feladattal párhuzamosan kezdünk el futtatni. Ez egy elég sokat alkalmazott eljárás. Gondoljunk csak arra, hogy ha mérni szeretnénk egy rendszer válaszát egy gerjesztésre, akkor párhuzamosan kell előállítani a gerjesztőjelet és párhuzamosan kell mérni az erre adott választ is. Persze ezeknek a feladatoknak egy részét meg lehet úgy is oldani, hogy maga a tesztlépés lekezeli ezt a párhuzamosságot, de sok esetben egy gerjesztéshez több mérés is tartozik, és ilyenkor már indokolt a párhuzamos futtatás.

A TestStand-ben a párhuzamos feladatvégrehajtás a **Sequence Call** Sequence Call lépéssel oldható meg. A **Sequence Call** segítségével egy új alszekvencia végrehajtását indítjuk el, és kiválaszthatjuk, hogy ez a szekvencia párhuzamosan vagy sorosan hajtódjon végre a main

szekvenciával. Ehhez a **Step Settings** ablakban kell az **Execution options** mezőt a megfelelő állapotba (**New Thread**) állítani.



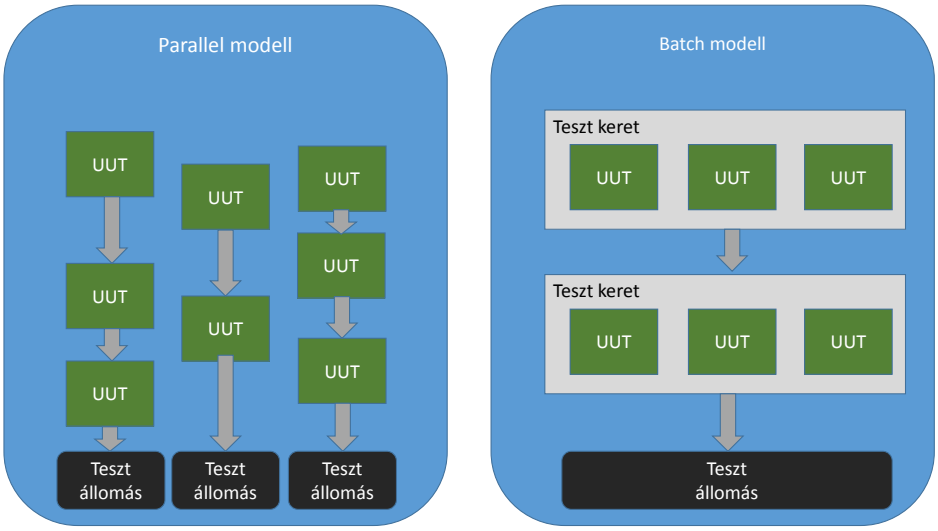
3.18. ábra. Az alszekvencia párhuzamos futtatásra állítása

Az alszekvenciák használatához még érdemes megjegyezni, hogy amennyiben paramétert szeretnénk átadni az alszekvenciának, az a legegyszerűbben úgy tehető meg, hogy az alszekvencia **Változók** ablakában létrehozzuk a kívánt változókat a **Parameters** opció alatt. Az így létrehozott paraméterek megjelennek a **Sequence Call** Step Settings ablakában, ahol a normál kifejezésekkel értékek adhatóak nekik.

3.11. Párhuzamos tesztelés: Több UUT párhuzamos tesztelése

Az automatikus tesztkörnyezetek akkor mutatják meg igazán az előnyüket, amikor egyszerre több terméket tudunk a segítségükkel úgy tesztelni, hogy külön jegyzőkönyvet gyártanak a tesztelt eszközökhöz és adott esetben a tesztelésben résztvevő hardware-ek használatát is automatikusan ütemezik.

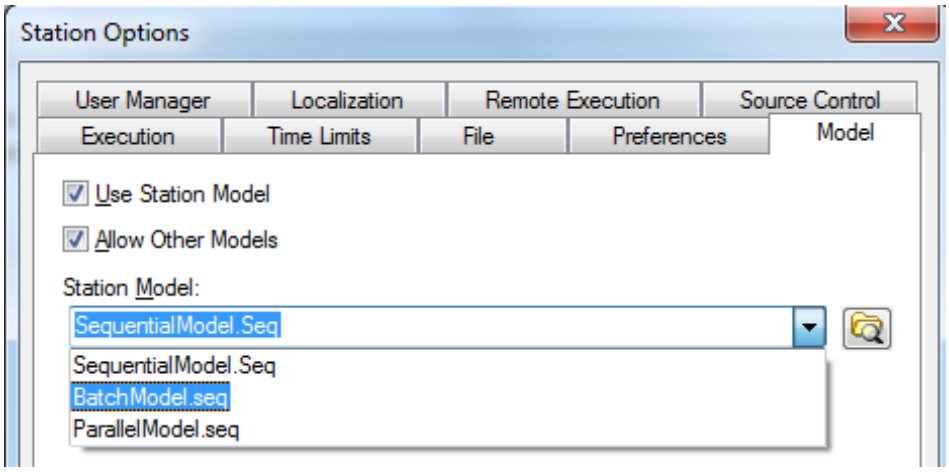
A TestStand alapvetően kétfajta párhuzamos tesztelési modellt támogat, az ún. **Parallel modell**-t és a **Batch modell**-t. A **Parallel modell**-t akkor célszerű használni, ha a tesztelt termékek teljesen szinkronizálatlanul párhuzamosan jelennek meg több csatornán, míg a **Batch modell** esetében azt feltételezzük, hogy a több tesztelendő termék egy közös rekeszbe, táblába van behelyezve, és ezen az ún. Batch-en nagyjából szinkron futtatjuk le a teszteket (***3.19. ábra***).



3.19. ábra. A Parallel és Batch modell összehasonlítása

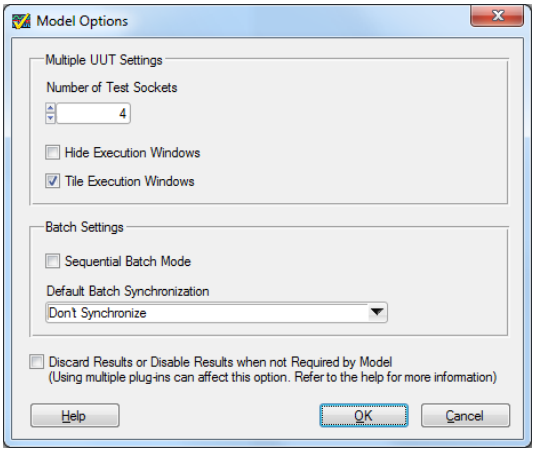
A mi esetünkben a **Batch modell**-t fogjuk alkalmazni, mert a mérőkártyánk is ennek megfelelően lett kialakítva: úgy, hogy egyszerre 4 UUT-t tudjon befogadni, amelyeken kb. egyidejű teszteket tudunk végrehajtani.

A **Batch modell** alkalmazásához a **Configure / Station Options** menüpont **Model** fülében tudjuk átállítani a tesztvégrehajtást az alapértelmezett sorrendi, szekvenciális modellről a Batch modell-re (***3.20. ábra***).



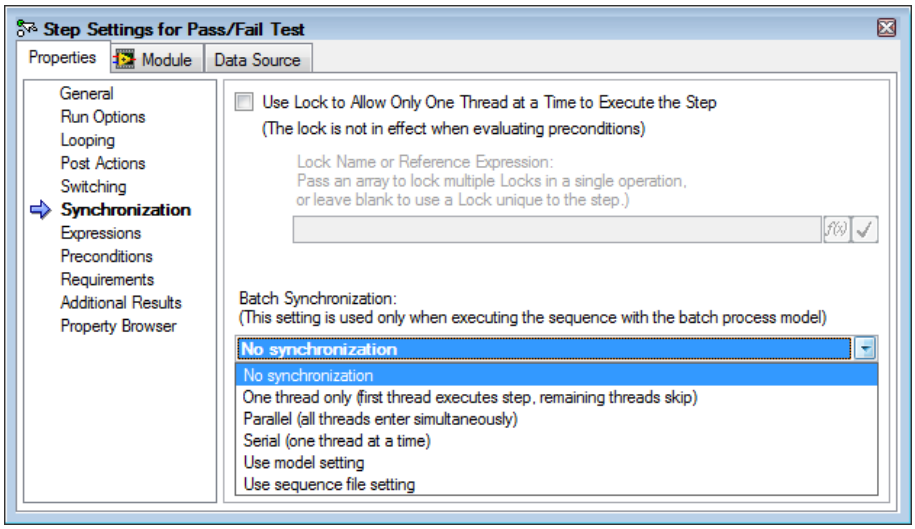
3.20. ábra. A tesztvégrehajtási modell átállítása Batch modell-re

A tesztelési modell átváltása után a **Configure / Model Options** részében tudjuk meghatározni, hogy egyszerre hány eszközön hajtsuk végre a tesztet (3.21. ábra).



3.21. ábra Az egyszerre tesztelt UUT-k számának megadása

Az egyes tesztlépésekre ezek után három végrehajtási lehetőség adódik, amelyeket a **Steps Setting** ablak **Properties** fülének **Syncronizations** opciójában tudunk meghatározni (3.22. ábra).



3.22. ábra Az egyes lépések lefutási módjának meghatározása párhuzamos teszteknél

A fontosabb lehetséges beállítások értelme a következő:

- **No syncronization:** Ennél a beállítási módnál nem kényszerítünk semmit a végrehajtandó lépésre.

- **One thread only:** ***Ez egy lényeges beállítás, ugyanis ebben a módban az adott lépés csak egyetlenegyszer hajtódik végre.*** Ez azért lényeges, mert egy Batch tesztelésnél is számtalan olyan parancsot kell végrehajtani, amely az egész tálcára vonatkozik, nem egyesével a vezérlőkre. Ilyen például a klímakamra hőmérsékletének beállítása, a tálca közös tápfeszültségének beállítása stb.

- **Parallel:** A lépés végrehajtását az összes Batch beállításban megadott UUT-ra külön szálként végezzük el. ***Vigyázat: ahhoz, hogy ez LabVIEW környezetben tényleg párhuzamosan működjön, szükséges, hogy a meghívott VI reentráns legyen (ez beállítható az adott VI Front Paneljének File/VI Properties/Execution opciójában).***

- **Serial:** A lépést sorrendben egymás után hajtja végre az összes UUT-ra. Tipikusan akkor használjuk, ha az adott lépés olyan műszert használ, amely nem tud párhuzamosan méréseket végezni az összes UUT-n.

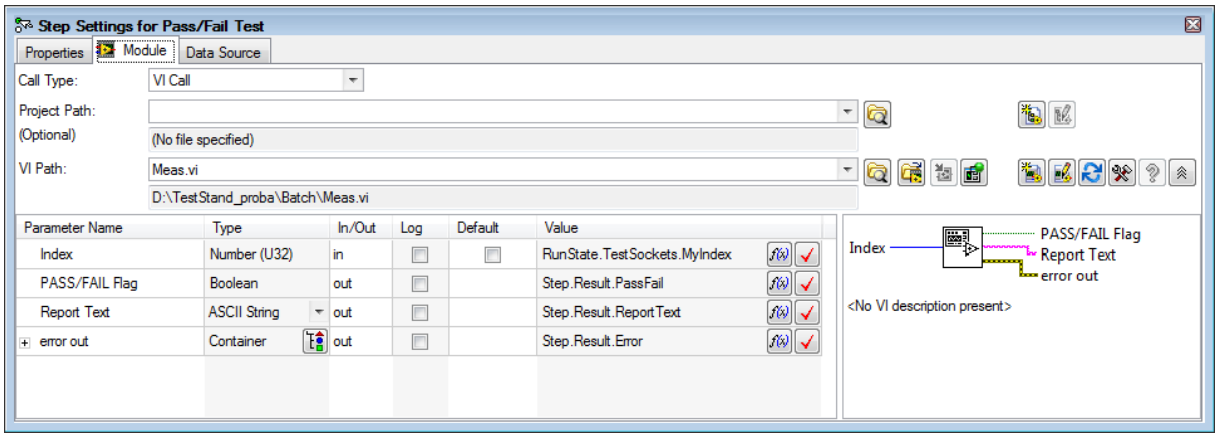
- A többi opció vagy a szekvencia, vagy a tesztrendszer alapbeállítását örökli. Ezeket sokszor használjuk default értékként, mert így központilag vezérelhető a végrehajtás módja.

3.12. Párhuzamos tesztelés: erőforrás használat

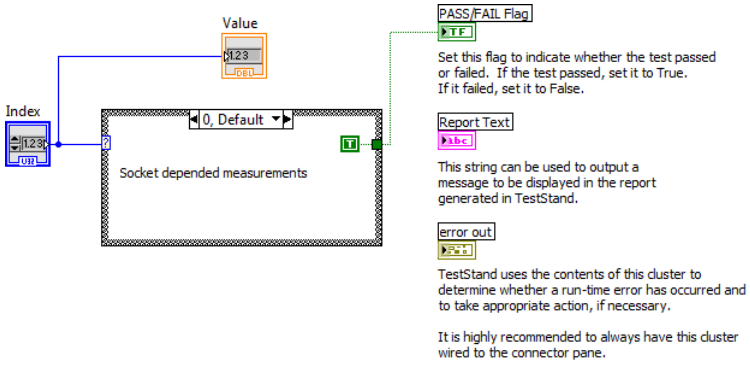
Az egyik legnagyobb probléma a több szálon futó teszteknél, hogy hogyan mondjuk meg, hogy melyik UUT melyik hardware-t használja a méréshez. Illetve, abban az esetben, ha nincs párhuzamosan elég hardware-ünk, azt is le kell vezényelnünk, hogy a mérőeszközünk egy kapcsoló mátrixon keresztül átkonfigurálja magát és az UUT-nak megfelelő bemenetet kezdje el mérni.

A mérés átkonfigurálására szolgál a TestStand-ben a **Switching** opció az egyes lépéseknél, amely segítségével megadhatjuk, hogy a mérőeszközünkre milyen jelek kerüljenek rá (bonyolultabb rendszereknél tipikus konfiguráció, hogy egy drága mérőkártya van, ami előtt egy switch kártya helyezkedik el és a switch kártya segítségével váltunk a mérőkártyára jutó jelek között). Szerencsére erre az opcióra nem lesz szükségünk.

A mi tesztjeinknél elég az, ha meg tudjuk határozni, hogy melyik UUT-hez melyik hardware csatornát szeretnénk felhasználni. Ez viszonylag egyszerűen megtehető, mert az összes Batch teszt tartalmaz egy ***RunState.TestSockets.MyIndex*** változót, amely az adott UUT mérőhelyét azonosítja. Tehát elég ezt a változót eljuttatnunk a tesztet végrehajtó lépéshez, és értékének függvényében kiválasztani a megfelelő hardware csatornát. Erre a legegyszerűbb mód, hogy a mérési lépést ellátjuk egy bemenettel is (ne felejtsük bekötni a Front Panel connection plane-jébe sem az adott bemenetet). Továbbá a hívási interfész megadásánál átadjuk a ***RunState.TestSockets.MyIndex*** változó értékét az adott lépésnek (3.23, 3.24 ábra).



3.23. ábra Egy Pass / Fail teszt kiegészítve a párhuzamos tesztelés mérőhelyét azonosító indekszel



3.24. ábra Példa a párhuzamos tesztelés mérőhelyét azonosító index felhasználására

4. Mérési feladatok végrehajtásának javasolt sorrendje

Ebben a fejezetben egy step-by-step feladatlistát adunk, amely feladatok végrehajtásával eljutunk a kész tesztrendszerig.

1. Próbáljuk ki a műszerek vezérlését: Nyissuk meg az előre létrehozott ***infofeld*** LabVIEW projectet és adjuk hozzá a *SDP.llb*-ből a tápegység kezeléséhez szükséges VI-kat, majd próbáljuk ki ezeket! ***A tesztkártya 5V-ról működik, nagyobb feszültséget ne adjunk rá!*** Ezek után hozzunk létre a projectünkben DAQmx taszkokat (a taszkok létrehozásánál mintaként felhasználhatóak a többi slotra már létrehozott DAQmx taszkok):
    a. DAQmx taszk a Mod1 tesztslot bal csatornájának mérésére. Javasolt beállítás 1 MHz mintavételezési frekvencia 500k minta szám.
    b. DAQmx taszk a Lin gerjesztő bemenet előállítására. Javasolt beállítás 1 MHz mintavételezési frekvencia 2M minta szám. ***Vigyázat az Lin bemenő jel ne legyen 0,7V-nál magasabb!***
    c. DAQmx taszk a Mod1 tesztslot LED-jeinek meghajtására.

2. DAQmx taszkok kipróbálása:
    a. Hozzunk létre egy egyszerű VI-t amivel szinuszos gerjesztést tudunk adni a tesztpanelnek!
    b. Hozzunk létre egy mérő VI-t, amivel kipróbáljuk a bemeneti DAQmx taszkunk működését!

3. TesztStand első lépések: Hozzunk létre egy új TestStand szekvenciát! A szekvenciához adjuk hozzá a következő lépéseket:
    a. A Setup csoportban állítsuk a tápot 5V-os feszültség értékre, majd engedélyezzük a kimenetét (a táp bekapcsolása után illik egy kicsit várni a ***wait*** dobozzal)! A Cleanup csoportban kapcsoljuk le a tápot!
    b. Hozzunk létre egy tetszőleges típusú tesztlépést, amelynek a sikerességét majd dinamikusan manipulálni tudjuk!
    c. Valósítsuk meg, és próbáljuk ki az első tesztslot visszajelző LED-jének működtetését: sárga: a teszt fut, piros: a teszt elbukott, zöld: a teszt sikeres volt. A teszt sikerességéről a ***RunState.SequenceFailed*** rendszerváltozó ad tájékoztatást.

4. Analóg mérések:
    a. Hozzunk létre egy új TestStand szekvenciát, és helyezzük el benne a szinuszjel generálására létrehozott VI-nkat!
    b. Az új szekvencia paraméterei közé vegyünk fel két változót: a frekvenciát és az amplitúdót, amelyek segítségével a generált szinuszjel tulajdonságait be tudjuk állítani.
    c. Hívjuk meg a main szekvenciából a létrehozott alszekvenciát és adjuk át a számára szükséges paramétereket (első lépésben konstans értékek is megfelelőek). A szekvencia hívásnál állítsuk be, hogy új szálként tehát párhuzamosan fusson a meghívott szekvencia!
    d. A gerjesztő szekvencia hívása után rakjuk be egy tesztlépést (javasolt a ***Multiple Numeric Limit Test***), amely az analóg bemenet mérését tartalmazza! Javasolt egy keveset várni a tényleges mérés előtt, hogy a generált jel biztos

megjelenjen. Valamint azt is illik megvárni, hogy a generátor taszk befejezze a működését. Ezekre a legegyszerűbb, megoldás a ***wait*** doboz alkalmazása.

e. A mérő tesztlépést alakítsuk úgy át, hogy a kimenetén az erősítés értéke és a THD mérőszám jelenjen meg (az erősítés kiszámolásához szükségünk lesz a gerjesztés értékére is). A mért eredményeket első körben elég konstans limit értékekkel ellenőrizni!

5. Mérési paraméterek, limitek file-ból való beolvasása:
   a. Hozzunk létre a Frek_Values[0..18], Frek_THD_Limit[0..18], Frek_Gain_Limit_L[0..18], Frek_Gain_Limit_H[0..18], Ampl_Values[0..11], Ampl_THD_Limit[0..11] változó tömböket a megadott névvel és méretben!
   b. Olvassuk be a tömbök értékét egy Property Loader parancs segítségével!

6. Mérések a frekvencia függvényében:
   a. Valósítsunk meg egy for ciklust, amely végiglépked a Frek_Values[0..18] tömb értékein és leméri az adott frekvenciához tartozó erősítési és harmonikus torzítási paramétereket! A gerjesztő jel amplitúdója legyen 0,3V!
   b. A mérő tesztlépés konstans limitértékeit cseréljük le a Frek_THD_Limit [0..18], Frek_Gain_Limit_L [0..18], Frek_Gain_Limit_H [0..18] tömbökben található limit értékekre!

7. Mérési eredményekből grafikon rajzolása a tesztreportba:
   a. Hozzunk létre két eredménytömböt az erősítési érték, és a THD mérések eredményeinek gyűjtésére!
   b. Hozzunk létre egy Action step-et, aminek átadjuk paraméterként az eredmény tömböket és a mérési beállításokat!
   c. Ábrázoljuk az eredményeket egy XY Graph-on!
   d. Mentsük el az ábrát egy .png file-ként, úgy, hogy a file nevébe bele legyen kódolva tesztelt UUT sorszáma! Az UUT sorszámát a ***RunState.Root.Locals.UUT.SerialNumber*** környezeti változó tartalmazza (ne lepődjünk meg, ha a rendszer nem ismeri fel ezt a változót a tesztlépés szerkesztés közben, mert ez a változó csak futási időben létezik).
   e. A tesztlépés **Report Text** visszatérési értékét módosítsuk úgy, hogy a generált ábra belekerüljön a jegyzőkönyvbe!

8. Mérések a teljesítmény függvényében:
   a. A frekvencia alapú méréseknél megismert módon valósítsuk meg a teljesítmény függvényében a harmonikus torzítás ellenőrzését! A gerjesztő jel frekvenciája legyen 1 kHz!

9. Mérések párhuzamosítása
   a. Állítsuk át a teszt modellt ***Batch model***-re!
   b. Azokat a lépéseket, amelyek az egész tesztkártyát érintik (tápforrás kezelés, jelgenerálás) állítsuk ***One thread only*** típusúvá!
   c. Azoknál a lépéseknél, ahol a végrehajtás függ a tesztslottól ott használjuk fel a ***RunState.TestSockets.MyIndex*** változót a megfelelő hardware csatornák kiválasztására! Közös erőforrás használati problémákkal nem kell törődnünk, mert bármi is a teszt beállítás, a létrehozott VI-ink alapértelmezetten nem reentránsok, ezért egymás után fognak csak lefutni, nem párhuzamosan. Ezért nem szükséges külön paranccsal sorrendi végrehajtást kérni rájuk.

5. Irodalomjegyzék

Az irodalomjegyzék összes web linkje 2016 februárjában volt utoljára ellenőrizve.

[1] National Instruments TestStand teszt automatizáló környezet: www.ni.com/**teststand**/

[2] Data Sheet „PAM8403 Filterless 3W Class-D Stereo Audio Amplifier" Diodes Incorporated 2012. http://www.diodes.com/_files/datasheets/PAM8403.pdf

[3] „PAM8403 audio power amlifier module" lehetséges beszerzési hely http://www.aliexpress.com/item/10pcs-lot-3W-2-Mini-Digital-Power-Audio-Amplifier-Board-USB-DC-5V-Power-Supply-PAM8403/32422014812.html?spm=2114.01010208.3.139.Kymov3&ws_ab_test=searchweb201556_6,searchweb201644_2_505_506_503_504_301_502_10001_10002_10016_10005_10006_10003_10004,searchweb201560_2,searchweb1451318400_-1,searchweb1451318411_6451&btsid=d3e623d1-c9e0-4830-a1f1-e36e75fece29

[4] Bruce Hofer: *„Measuring Switch-mode Power Amplifiers"* White Paper. 2003. Audio Precision company.

[5] Richard Palmer: *„Audio power amplifier measurements"* Texas Instruments Incorporated. 2001

[6] Mike Score: *„Reducing the output filter of a Class-D amplifier"* Texas Instruments Incorporated. 1999

[7] Walt Kester: *„Understand SINAD, ENOB, SNR, THD, THD + N, and SFDR so You Don't Get Lost in the Noise Floor"* Analog Devices, MT-003 Tutorial. Rev.A, 10/08, WK.

[8] „Laboratory Grade Remote Programming Switching Mode DC regulated Power Supplies SDP Series. SDP – 2210 / 2405 / 2603." User Manual. 7673-2405-0009. Rev.0 01/2009. Manson. http://www.manson.com.hk/getimage/index/action/images/name/51498e6b1ecfa.pdf

[9] „For all SDP series power supply". Remote Control Software Labview Driver. Verison 1.3 (120904). http://www.manson.com.hk/products/detail/168