

1. Számítógépes mérések vezérlőszoftverei

Az elmúlt időszakban nemzetközi viszonylatban is, de Magyarországon különösen nagy mértékben nyert teret a LabView fejlesztő szoftver rendszere a mérés- és automatizálási feladatok megoldásában, ezért a tananyag elsősorban ennek a fejlesztőkörnyezetnek az ismertetését tűzte ki célul. A LabView gyakorlatilag minden olyan ipari vállalat tesztmérnökségén megtalálható, ahol számítógéppel vezérelhető mérőrendszereket alkalmaznak, és a magyar műszaki felsőoktatás méréstechnikai kurzusai is elsősorban erre a szoftverre épülnek.

1.1. Bevezetés a LabView szoftver alkalmazásába

A LabView egy grafikus programfejlesztő, amely elsősorban méréstechnikai és a hozzákapcsolódó jelfeldolgozási feladatok megoldására szolgál, de alkalmas más, pl. szimulációs munkákra is. A grafikus programozás egy látványos, könnyen követhető programozási módot jelent, amelyet könnyen sajátíthatnak el a hagyományos programnyelveket nem ismerő szakemberek is. Mindazonáltal meg kell jegyezni, hogy a programozás alapvető jellemzői (változók deklarálása, ciklusok) teljes mértékben a C programnyelvhez hasonlóan, illetve azzal analóg módon történnek, ezért bármilyen hagyományos programnyelv alapszintű ismerete sokat segít a kezdeti lépésekben. Természetesen az, hogy mi az *integer*, vagy *double* (Pascalban *real*) típusú változó, vagy hogyan használható a *for* és *while* ciklus, gyorsan megtanulható apróságok, ezért valóban ajánlható a LabView mindenkinek, aki gyorsan és egyszerűen szeretne saját mérőprogramot készíteni. Ez az ismertető elsősorban azoknak készül, akik legalább alapszinten ismernek valamilyen hagyományos programozási nyelvet, ezért a fentebb említett ismereteket itt nem tárgyaljuk. Azoknak a programozóknak, akik nagy integritású rendszereket kívánnak létrehozni és ismerik a C programnyelvet, javasoljuk a LabWindows CVI (Measurement Studio) alkalmazását a LabView helyett, mert ennek használatával gyorsabban és egyszerűbben juthat eredményre. Meg kell azonban jegyezni, hogy bizonyos területeken a LabView abszolút kiemelkedő és egyedülálló szolgáltatásokat nyújt, ilyen például a 3.fejezetben ismertetett Ethernet alapú távvezérelt mérések.

Mennyi időre van szüksége egy műszaki egyetemi hallgatónak a LabView alapszintű elsajátításához?

A mellékelt példaprogramokon keresztül gyorsan elsajátíthatja az alaplépéseket:

A rendszer működésének megismerése:	max. 3 óra
A példaprogramokon keresztül a programozás alapszintű megismerése:	kb. 5-6 óra
Mérésadatgyűjtés alapszintű megismerése:	kb. 5-6 óra

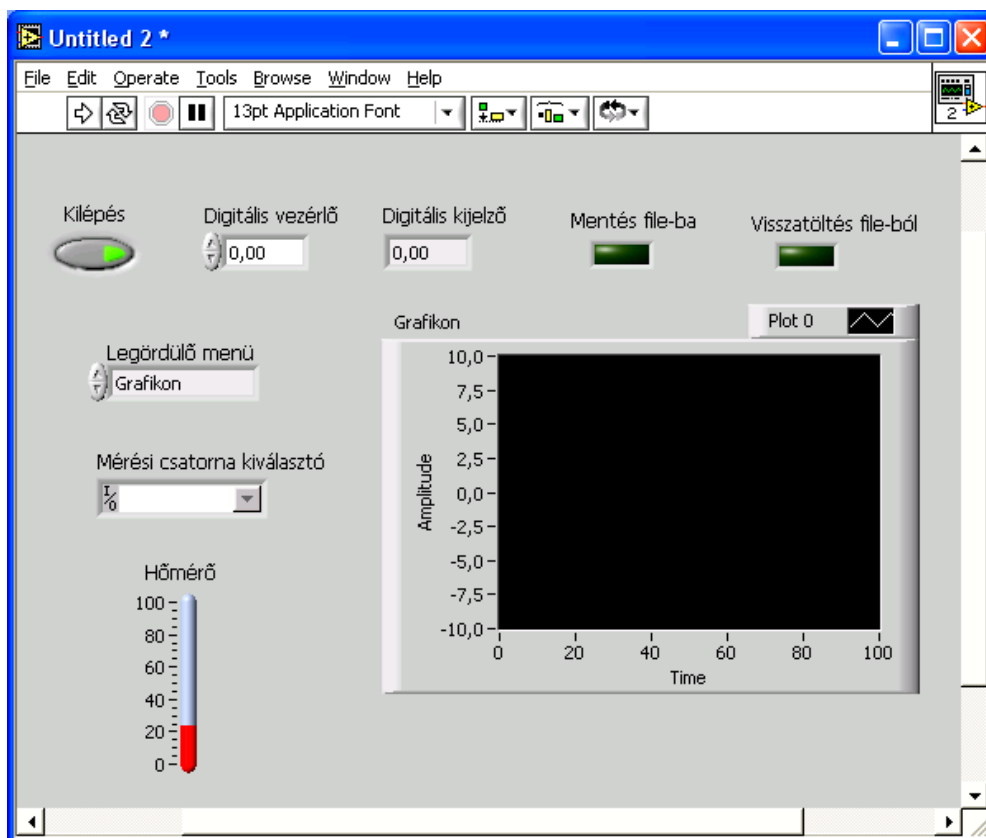
1.2. A LabView működése

Mivel a rendszer virtuális műszerezésre szolgál, így az elkészült forrásprogramok „vi” (virtual instrument) kiterjesztést kapnak. A *vi* programok speciális könyvtárakba - VI Library – rendezhetők. A VI library a Windows intézőben egy file-ként jelenik meg „llb” kiterjesztéssel, de LabView környezetben egy VI library természetesen több *vi* programot is tartalmazhat.

Ha LabView-ban programozunk, akkor a legegyszerűbb esetben kétfajta ablakot használunk: a felhasználói kezelőfelület ablakát (Front Panel) és a grafikus program ablakát (Block Diagram).

A Panel

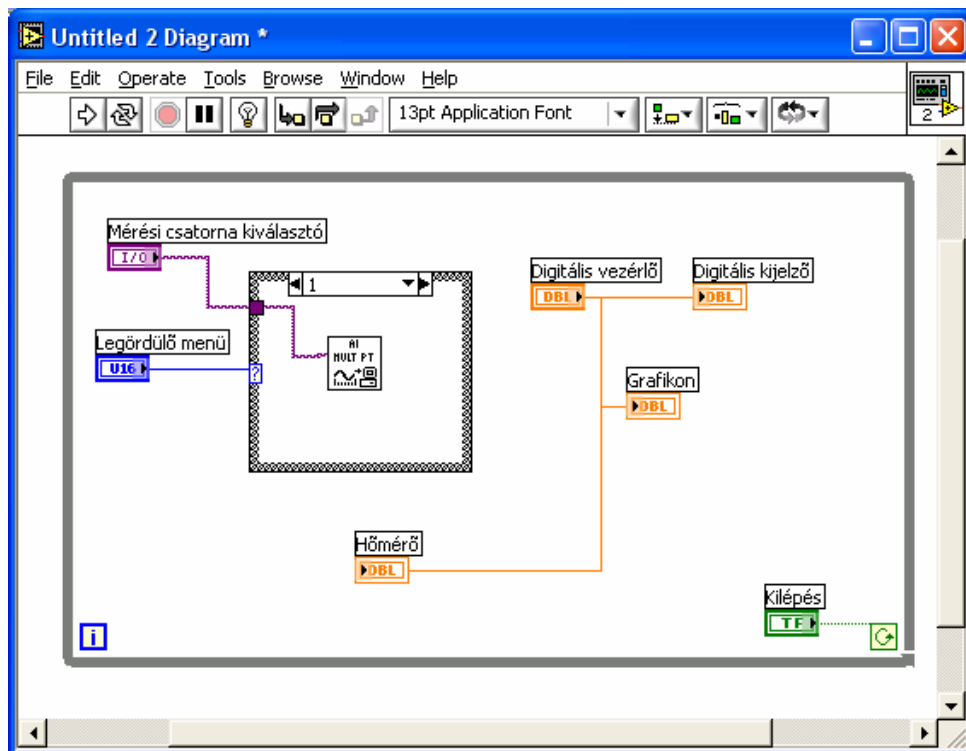
A felhasználói ablak jelenik meg az elkészült *.exe program futtatásakor a képernyőn. Ezen az ablakon vannak elhelyezve a programot vezérlő nyomógombok, kapcsolók, (pl. kilépés a programból, mérés indítása, file-ba mentés, visszatöltés, stb.), értékmegadó mezők (pl. mintavételi frekvencia beállítása, csatornák számának megadása, stb.), különböző kijelzők, grafikonok, vagy legördülő típusú menük, vagyis amit a program használója lát. A program fejlesztése közben a LabView fejlesztő környezetéből történő futtatás során is ezen a képernyőn tesztelhető a program.



4-1. ábra Felhasználói kezelőfelület

A Diagram

A Panelen elhelyezett objektumok névvel ellátva automatikusan megjelennek a grafikus program ablakában. Itt tervezzük meg a program futását. Tulajdonképpen olyan ez, mint a szokásos programnyelvekben (Pascal, C) a szöveges formátumú forrásnyelvű program, csak itt nem a szövegesen beírt sorok futnak le sorba egymás után, hanem a grafikus jelekkel meghatározott műveletek, függvények, a köztük lévő vezetékezésnek megfelelően.



4-2. ábra Grafikus program LabView-ban

1.3. A LabView kezelése

Csak a Windows környezetben megszokott funkcióktól eltérő menük kerülnek bemutatásra, a teljesség igénye nélkül, kezdők számára.

A fejlesztő kétféle módban működtethető, szerkesztő és futató módban. Az újabb verziókban (6.) kezdetben nincs jelentősége, hogy melyiket használjuk, mert a szerkesztő üzemből is megfuttatható a program. Ezért javaslom kezdetben a szerkesztő módot használni a fejlesztés alatt.

Átváltás a szerkesztő és futató üzem között: *Operate – Change to Run/Edit mode*

Menü



A program futtatása ezzel az ikonnal történik,



futás közben átvált feketére,



de ha a nyíl összetöredezett képet mutat, akkor a program nem futtatható, mert hibás. Ilyenkor az egérrel a nyílra kattintva kapunk egy hibalistát. A listán bármely hibára rákattintunk, részletes leírást ad a kiválasztott hibáról, és a *Show Error* gombot megnyomva, a helyét is megmutatja a diagramon.



A program folyamatos futtatása. Ha a program a végére ér, kezdi előlről. Alkalmazását nem javaslom, legfeljebb különleges programtesztelésekkor.



Vészgomb, a program vészleállítására. Üzemszerűen a program a panelon leállítható kell legyen, ezért ezt a gombot csak akkor használjuk, ha a program leállító gombjával ez nem sikerül (pl. végtelen ciklust hoztunk létre).



pillanatmegállítás a program futása közben



Csak a Diagram ablakban találjuk meg, ha futtatás közben a „lámpát” bekapcsoljuk, akkor vizuálisan követhetjük a program futását a grafikus programban. Láthatjuk, mikor melyik változó, milyen értéket vesz fel és továbbít.

Hasznos menüpont lehet az *Edit* menü *Remove Broken Wires* pontja

Tools Palette

Szerkesztés közben mindig legyen a képernyőn. Megjeleníthető a *Windows - Show Tools Palette* menü kiválasztásával. A program kezelését segíti. A különböző kurzorok segítségével a kezdeti értékek átírhatóak, az objektumok átnevezhetőek, átméretezhetőek, áthelyezhetőek, átszínezhetőek, stb.

A legfontosabb eszközök a kezeléshez:



A kezdeti érték megváltoztatása, nyomógomb átkapcsolása szerkesztő üzem közben.



Általános kurzor az objektumok áthelyezésére, átméretezésére szolgál. Ezzel a kurzorral az objektumok cím mezője az objektumtól függetlenül is mozgatható.



Objektumok megnevezéssel együtt történő áthelyezésére szolgál.



Bármilyen szöveg átírása. Akár menüsorban, akár címben, akár számérték mezőben ezzel átírható a beállított szöveg vagy érték.



Ezt a kurzort csak a Diagram ablakban használjuk, az objektumok összekapcsolására. Ezzel a „vezetékekkel” jelöljük ki a program futásának sorrendjét.



Színek megváltoztatása.



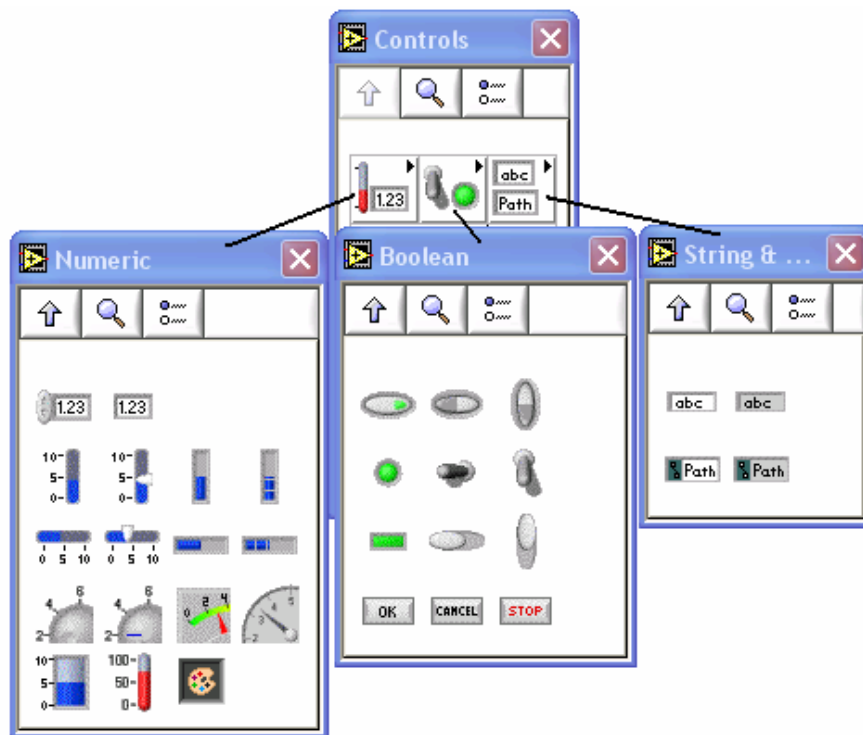
A program futása közben megszakítási pontok (breakpoint) elhelyezése. Ezt a kurzort csak a Diagram ablakban használjuk.

A beállított értékek csak akkor kerülnek elmentésre, tehát akkor fogja a program default-ként kezelni, ha az *Operate – Make Current Values Default* menüpontra lépünk.

Panel

A Panelre a vezérlő palettáról (Controls Palette) választhatjuk ki a programunkhoz szükséges vezérlő és kijelző objektumokat. A vezérlő paletta megjeleníthető a jobb egérgombbal, vagy a *Window – Show Controls Palette* menüpont kiválasztásával.

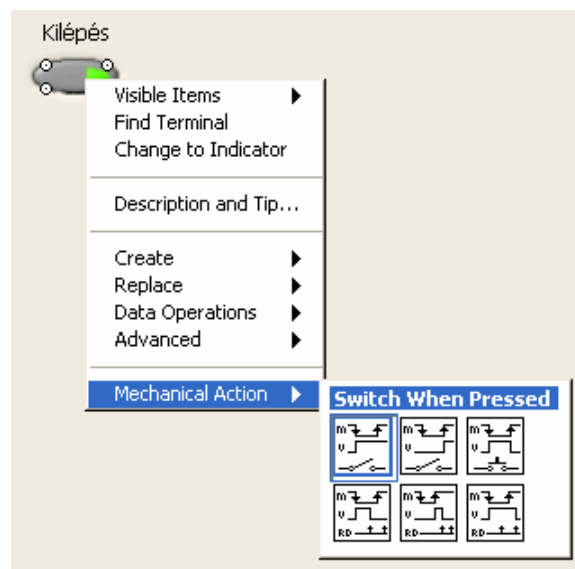
A paletta felső sora:



számérték megadása kétállású kapcsolók szövegmezők

4-4. ábra A Controls Palette felső sorából választható objektumok

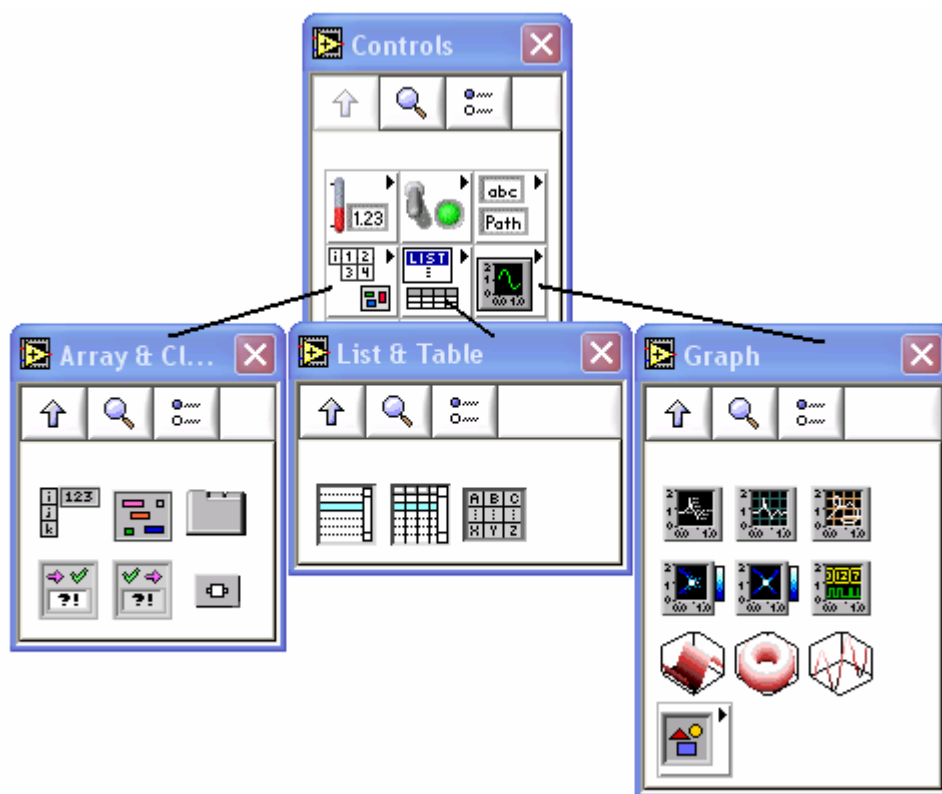
Amennyiben a Boolean típusú kapcsoló vezérlő, akkor meghatározhatjuk a kapcsolás módját:



4-5. ábra Kétállású vezérlő kapcsolók kapcsolási módjai

- Megnyomásra kapcsol, és bekapcsolva marad.
- Elengedésre kapcsol, és bekapcsolva marad.
- Addig tart bekapcsolva, amíg a gombot lenyomva tartjuk.
- stb. az ábrák szerint.

A paletta második sora:



tömbök megadása

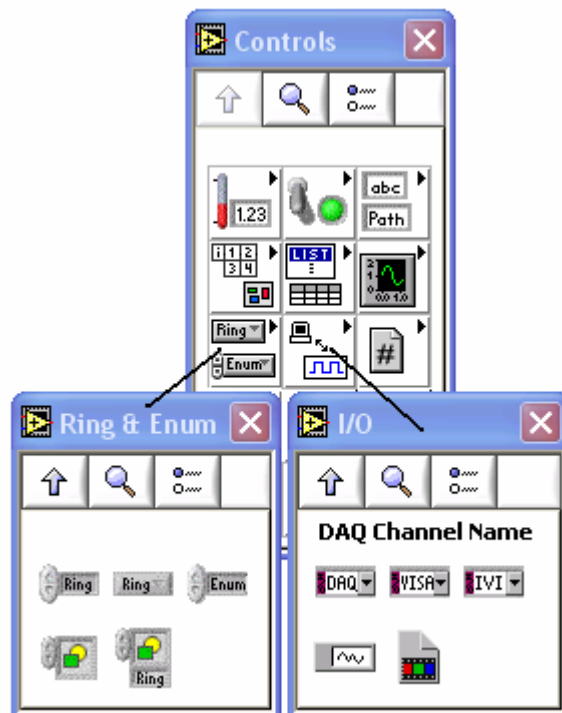
táblázatok

grafikonok (2D és 3D)

4-6. ábra A Controls Palette másodík sorából választható objektumok

A grafikonok alkalmazásánál figyeljünk oda a *Strip Chart* és a *Graph* közötti különbségre. Az előző folyamatosan „futó” megjelenítésre, az utóbbi „álló” függvények megjelenítésére szolgál.

A paletta harmadik sorában találjuk a legördülő menüket, és a mérésadatgyűjtésnél használható csatorna kiválasztó mezőket.



legördülő menük mérőcsatornák kiválasztása

4-7. ábra A Controls Palette harmadik sorából választható objektumok

Diagram

LabView programozásban a változó típusok a C (Pascal) programnyelvekhez hasonlóan alkalmazhatóak, de itt a különböző típusok, különböző színekkel jelennek meg:



integer



double (real)



Boolean



string

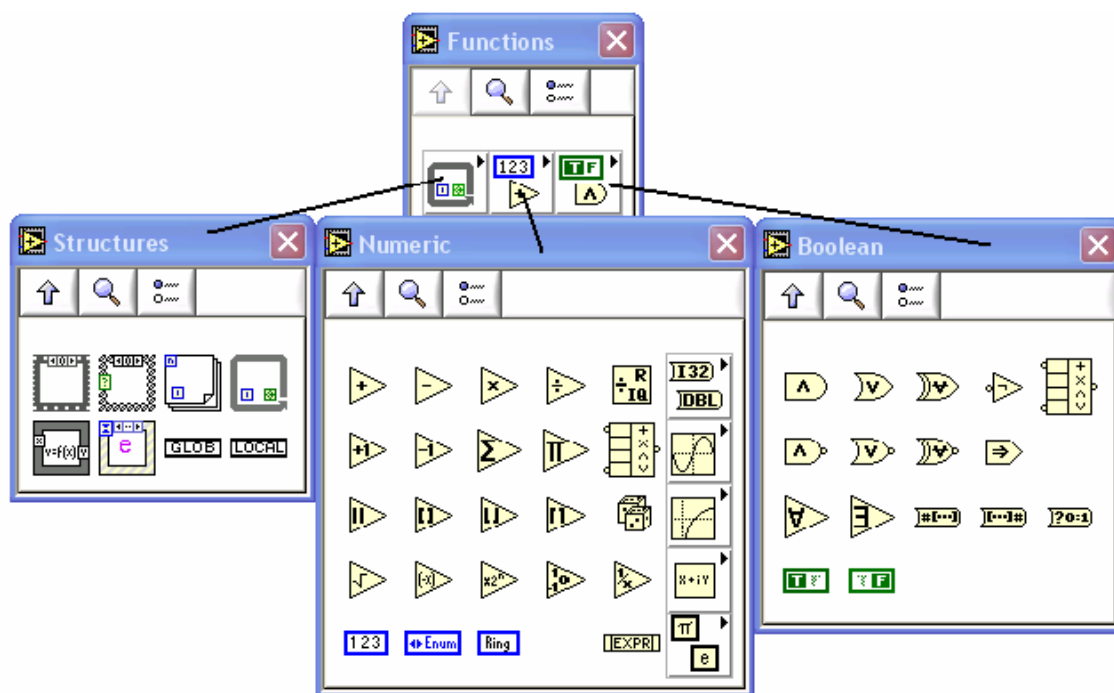


a Measurement and Automation – ban meghatározott virtuális csatorna. Lásd az 5. fejezetben

Integer és double változók típusa megváltoztatható a *Representation* menüpontban (egér jobb gomb)

A programban alkalmazható függvényeket a függvény palettáról (Function Palette) hívhatjuk be.

A függvény paletta felső sora:



ciklusok
while, for
case, sequence
egyenletek

számtani alpműveletek
trigonometrikus függvények

Boolean műveletek

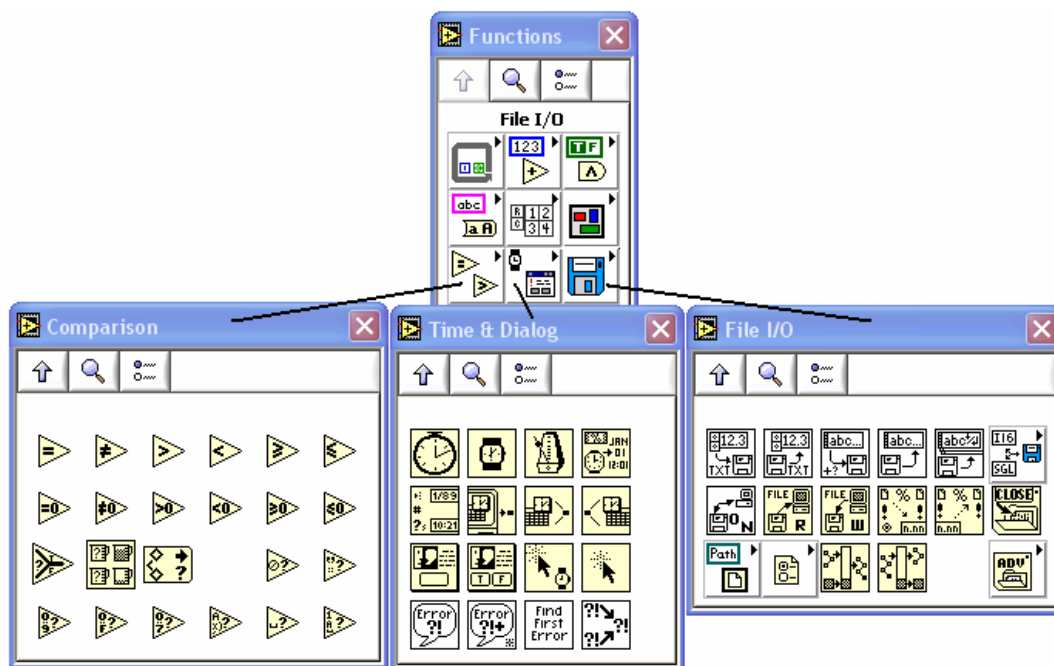
4-8. ábra A Function Palette felső sorából választható függvények

A függvény paletta második sora:



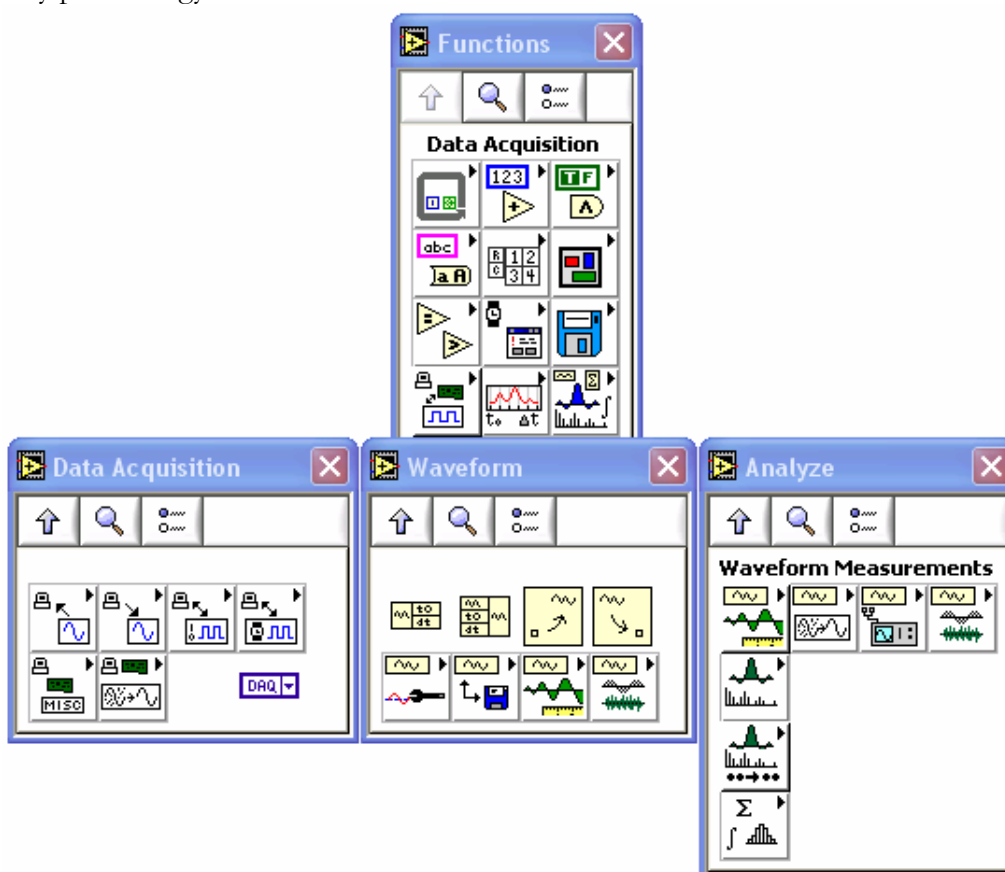
4-9. ábra A Function Palette második sorából választható függvények

A függvény paletta harmadik sora:



4-10. ábra

A függvény paletta negyedik sora:

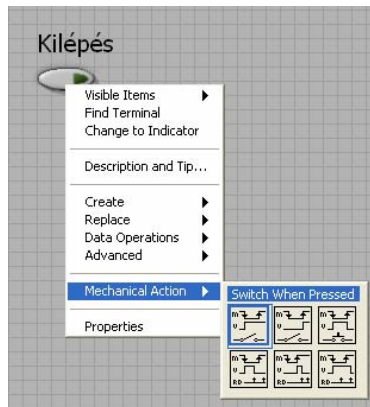


4-11. ábra

1.4. Bevezető mintaprogramok

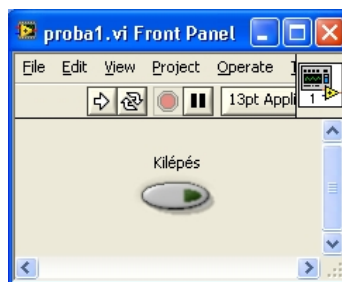
1.4.1. proba1.vi

1. Tegyük fel a panelra egy Boolean típusú kétállású kapcsolót. Adjunk nevet a kapcsolónak.
2. Tanulmányozzuk a kapcsolók üzemmódjait (Mechanical action – csak vezérlő üzemmódban). Jobb egér gombbal kattintsunk a kapcsolóra.

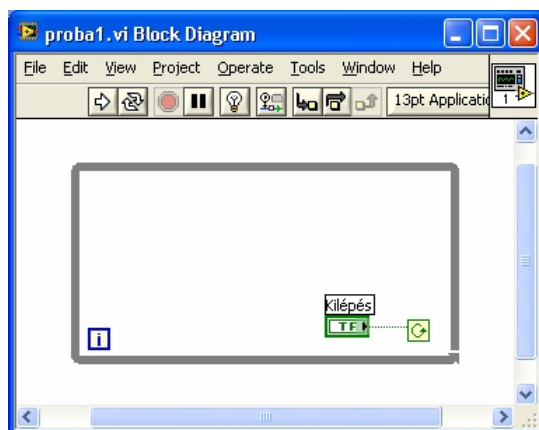


4-12. ábra

3. A diagram oldalon készítsük el a while ciklust. A while ciklus teljesen hasonlóan működik, mint a C programnyelvben, a bal oldalon lévő mező („i” betűvel) az iterációkat számolja, a jobb alsó sarokban lévő mezőhöz pedig a feltételt lehet hozzákapcsolni. A feltétel Boolean típusú értéket fogad, tetszőlegesen beállítható, hogy TRUE vagy FALSE értékre álljon le a while ciklus. Ezt a mezőre jobb gombbal kattintva érhetjük el a „Stop if TRUE” vagy „Continue if TRUE” menüpontok megfelelő kiválasztásával. A feltételhez kössük be a panelra feltett gombot.
4. A „program” futtatható, de futtatás előtt mindig mentjük el a munkánkat.



4-13. ábra Front Panel: proba1.vi

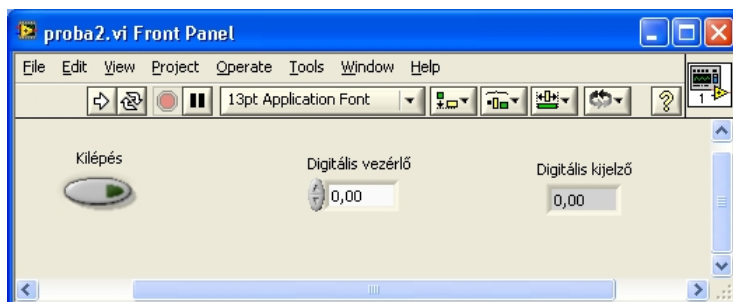


4-14. ábra Block Diagram: proba1.vi

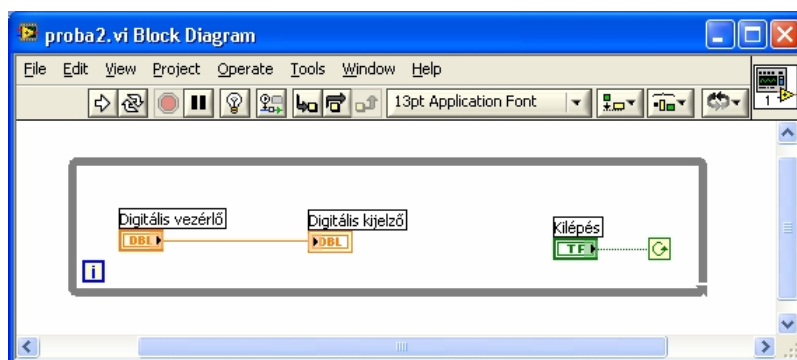
Megjegyzés: Ezt a feladatot az újabb verziókban egyszerűbben is megoldhatjuk, hiszen a block diagramm *Functio palette*-n közvetlenül létrehozható a programkeretet adó while ciklus. Didaktikai megfontolásból azonban első lépésnek a fenti módszer ajánlható.

1.4.2. proba2.vi

1. Tegyük fel a panelra egy digitális vezérlőt és digitális kijelzőt.
2. A while cikluson belül kössük össze a digitális kijelzőt és vezérlőt



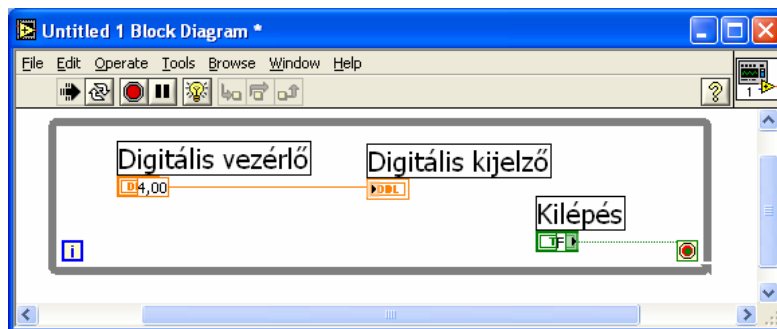
4-15. ábra Front Panel: proba2.vi



4-16. ábra Block Diagram: proba2.vi

Futtassuk a programot.

Nézzük együtt a panelt és a diagrammot úgy, hogy a diagramm oldalon a menüben található lámpát fölkapcsoljuk. Ebben az üzemmódban látványosan követhetjük a program futását.



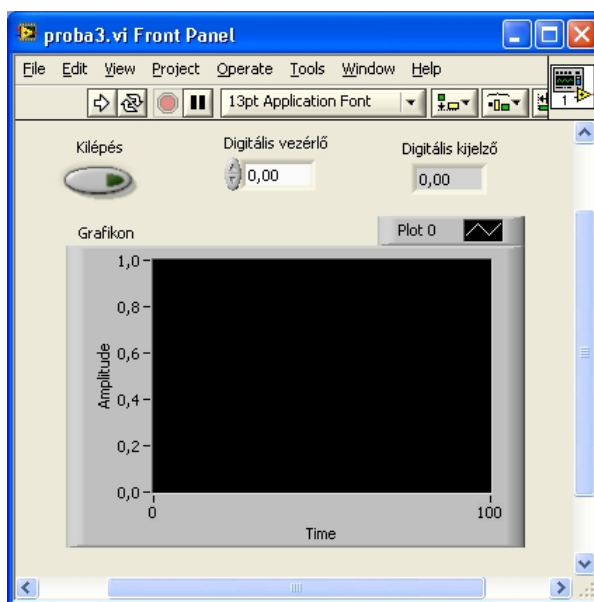
4-17. ábra

1.4.3. proba3.vi

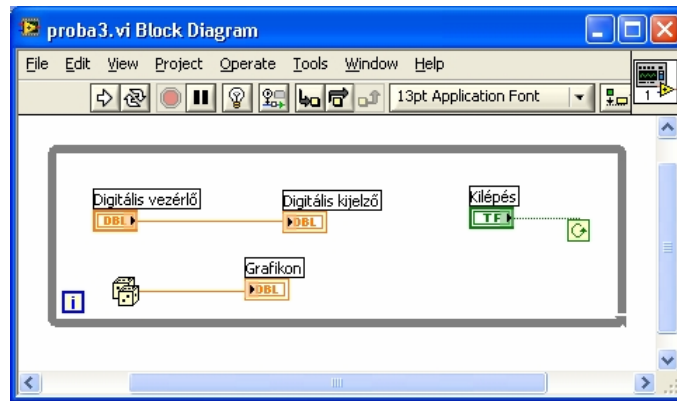
1. Tegyük fel a panelra egy waveform chart típusú grafikont és egy véletlenszám generátort (Numeric könyvtár, Random number).
2. A while cikluson belül kössük össze a generátort a grafikonnal.
3. Így mind a digitális vezérlő-kijelző, mind a véletlenszám kirajzolás működik.

Tanulmányozzuk a grafikon beállítási lehetőségeit. Nézzük meg az X és Y tengely autószkálázását, a kijelzés határainak beállítását, a grafikon jellemzőinek kijelzését, stb.

A LabView nagyon jól használható súgóval rendelkezik. Kapcsoljuk be a súgót a Help menü, Show Context Help menüpont kiválasztásával. Ha egy függvény „dobozkája” fölé állunk a kurzorral, akkor a súgó ablakban azonnal megjelenik a függvény leírása, és láthatjuk a bemenetek és kimenetek meghatározását és változó típusát. Bekötéskor jelzi a súgó, hogy melyik csatlakozási pont fölött vagyunk éppen. A súgó használata különösen ajánlott a mérésadatgyűjtő függvények alkalmazásánál.



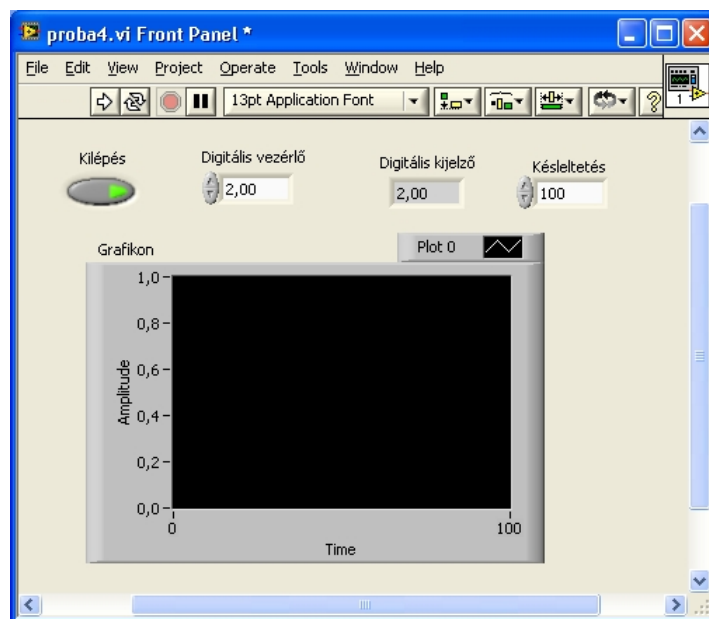
4-18. ábra Front Panel: proba3.vi



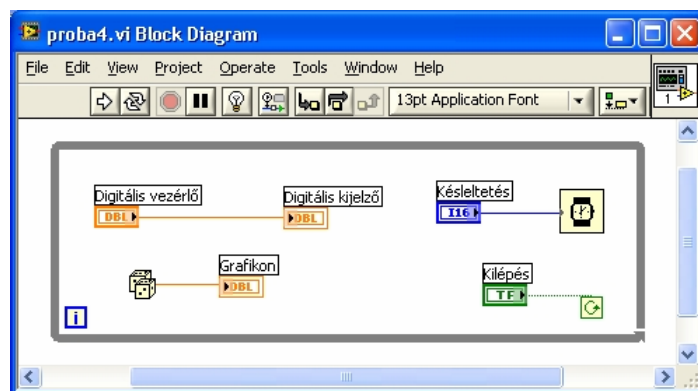
4-19. ábra Block Diagram: proba3.vi

1.4.4. proba4.vi

1. Tegyük a panelra egy digitális vezérlőt. Váltuk át integer típusú változóra (Representation, I16).
2. Tegyük fel egy késleltetőt a diagram oldalra - Time & Dialog menu Wait(ms).
3. Kössük össze a kettőt, és próbáljuk ki.



4-20. ábra Front Panel: proba4.vi

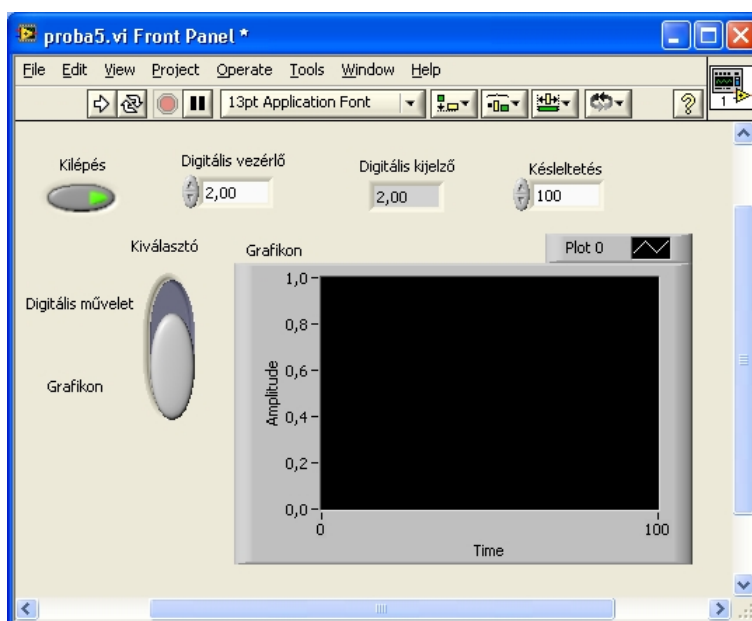


4-21. ábra Block Diagram: proba4.vi

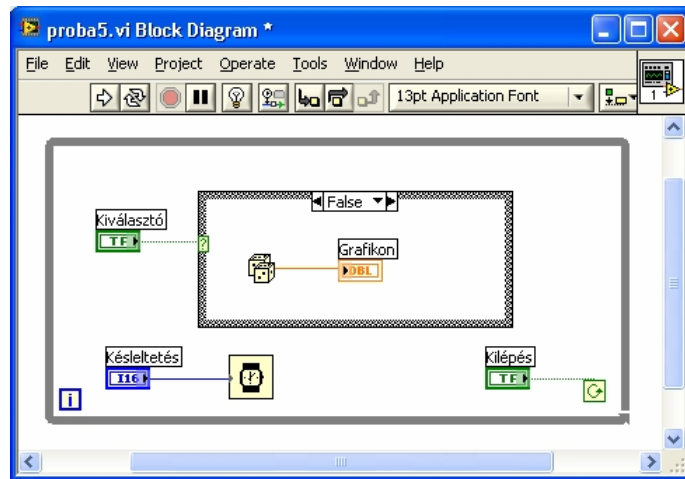
1.4.5. proba5.vi

Alakítsuk át a programot úgy, hogy egy kiválasztó gomb segítségével vagy a digitális vezérlő-kijelző páros, vagy a véletlenszám generátor-grafikon páros működjön.

1. Tegyük fel a panelra egy második kétállású kapcsolót.
2. A diagram oldalon készítsünk egy case struktúrát. A case struktúra két ablakot tartalmaz, egy TRUE és egy FALSE ablakot egymás mögött. Az egyik ablakba tegyük a digitális vezérlő-kijelző párost, a másik ablakba a véletlenszám generátor-grafikon párost. A case feltételmezőjéhez (kérdőjel a falon) kössük a második kapcsolót. Így az egyik állásban csak a grafikon működik, a másik állásban csak a digitális kijelző.



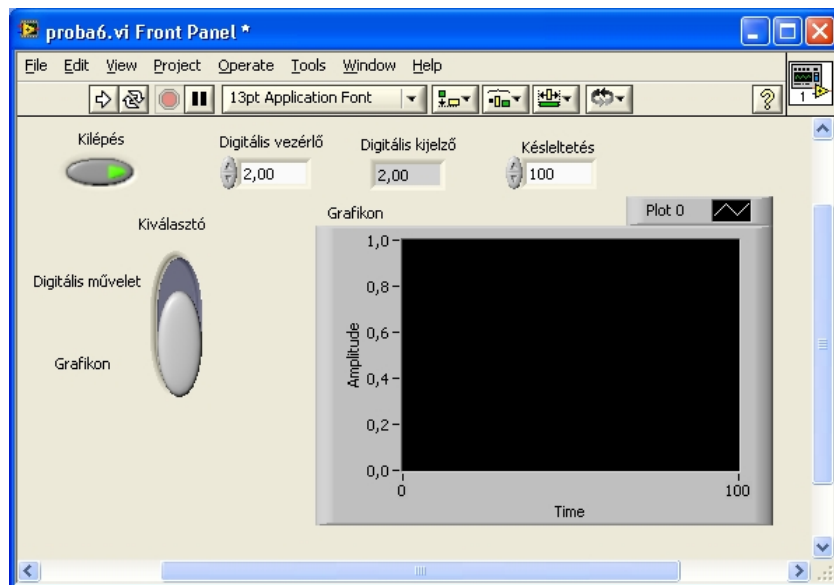
4-22. ábra Front Panel: proba5.vi



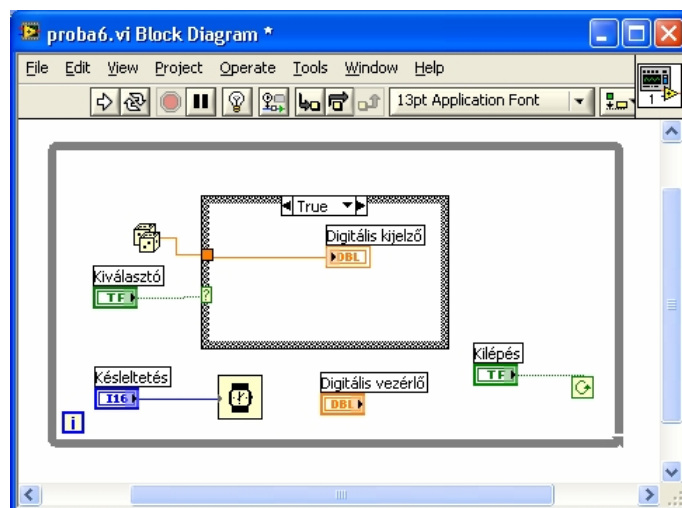
4-23. ábra Block Diagram: proba5.vi

1.4.6. proba6.vi

Alakítsuk át úgy a programot, hogy mindkét kijelzőn (grafikon és digitális kijelző) a véletlenszám generátor értéke megjelenhessen, de egyidőben csak az egyik jelenjen meg. Az, hogy éppen melyiken, azt a kétállású kapcsoló állása döntse el. Egyik állásban csak a grafikonon, a másik állásban csak a digitális kijelzőn lássuk „futni” az értékekekt. (A digitális vezérlőre nincs szükség).



4-24. ábra Front Panel: proba6.vi



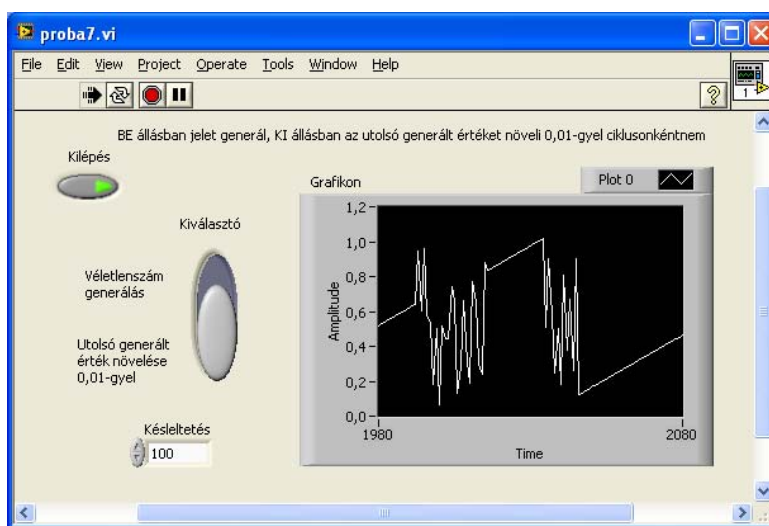
4-25. ábra Block Diagram: proba6.vi

1.4.7. proba7.vi

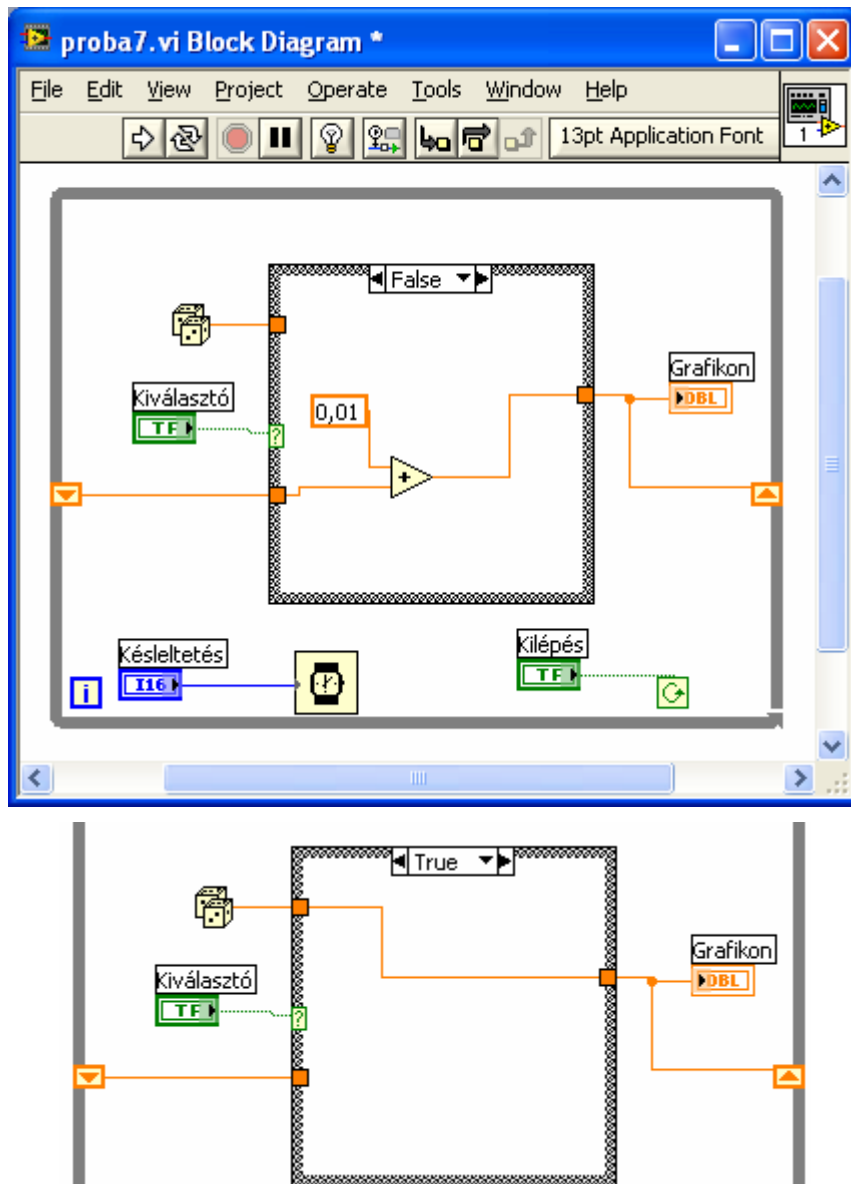
A LabView programozásában az egyik legnagyobb probléma, hogy a while cikluson belül megadott/kiszámolt értékek a ciklusok között „elvesznek”. Vagyis minden ciklus után úgy tűnik, mintha újra indulna az értékmegadás. A szöveges programnyelvekben ezzel ellentétben egy pl. globális változó, bármikor is kap értéket a program futása közben, a program végéig megtartja értékét, vagy, amíg azt meg nem változtatjuk.

A kérdés, hogyan tudjuk átadni az értékeket a LabView-ban az egyik while ciklusból a következőbe? Erre szolgál a shift register.

A while ciklus shift regiszterének kipróbálásához alakítsuk át úgy a programot, hogy csak grafikonra rajzoljon. A kiválasztó gomb egyik állásában generálja továbbra is a véletlenszámokat, a másik állásában pedig az átkapcsolás előtt generált utolsó értéket rajzolja ki úgy, hogy minden egyes ciklusban növelje meg az értéket 0,01-gyel.



4-26. ábra Front Panel: proba7.vi



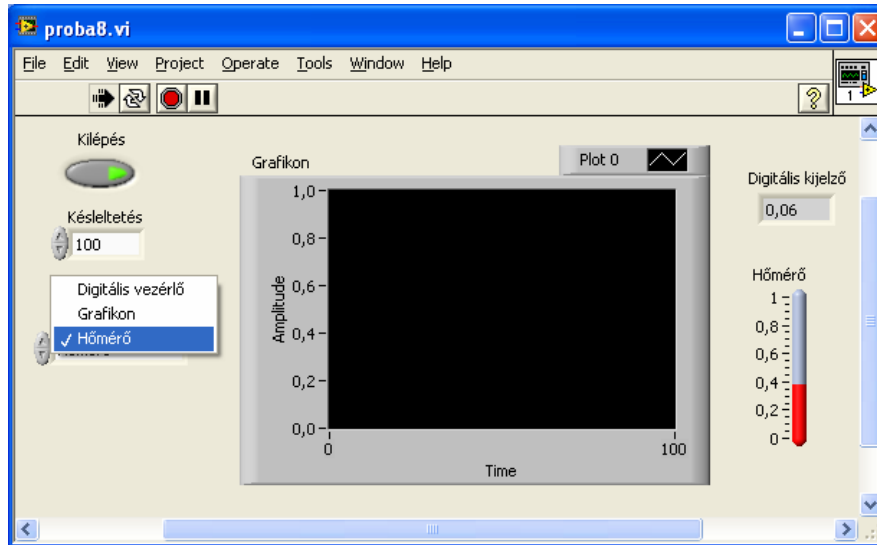
4-27. ábra Block Diagram: proba7.vi (Fent FALSE ablakkal, lent a TRUE ablak)

1.4.8. proba8.vi

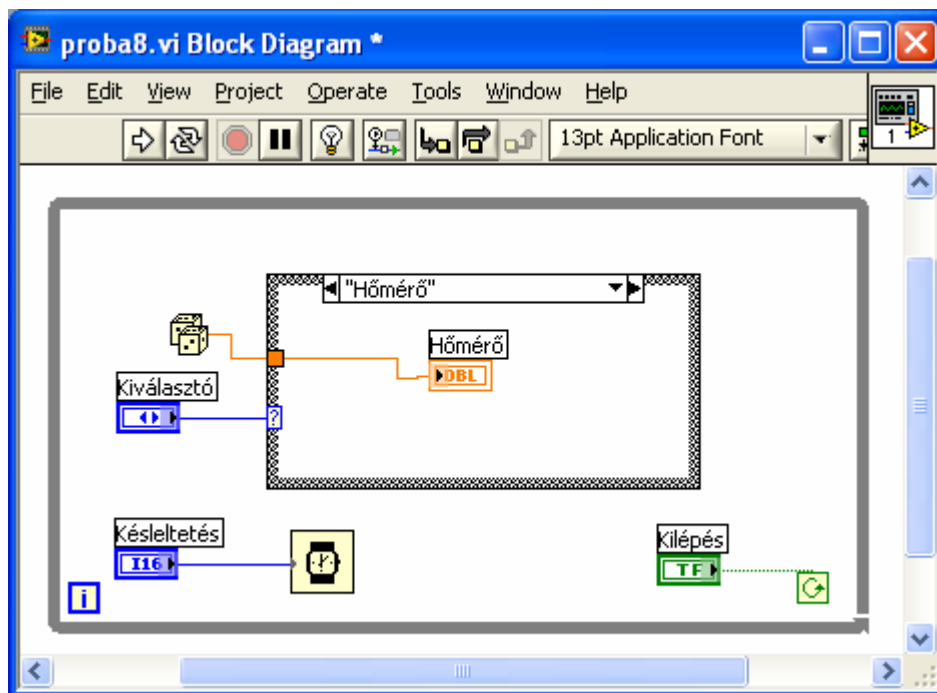
Alakítsuk át a 6. próbaprogramot úgy, hogy ne csak kétféle, hanem 3 különböző kijelzés közül választhassunk: maradjon meg a digitális kijelző, a grafikon, és legyen még egy hőmérő típusú kijelző is. Mivel ehhez a feladathoz egy három állású kapcsolóra lenne szükségünk, készítünk egy legördülő menüt 3 választási lehetőséggel.

1. Töröljük ki a kétállású kapcsolót és tegyük fel a panelre egy „Enum” típusú legördülő menüt. Ez alaphelyzetben egy sorral jelenik meg, ide beírhatjuk az egyik menüpontot, pl: „Digitális vezérlő”. A másik két menüpontot úgy tudjuk hozzáadni, hogy az egér jobb gombjával kiválasztjuk az „Add Item After” feliratot, s ekkor egy-egy menüpontot hozzáad a rendszer a legördülő menünkhöz.
2. Tegyük fel a panelre egy „Thermometer” típusú kijelzőt.
3. A Diagram oldalon kössük be az Enum ikonját a case struktúra bemenetére. A rákapcsolás pillanatában a case struktúra ablak elnevezései átváltanak az általunk beállított feliratokra. Mivel a case struktúra alaphelyzetben 2 ablakkal jelenik meg, a harmadik

ablakot az egér jobb gombjával az „Add Case After” felíratra kattintva tudjuk hozzáadni a programhoz. Ha az ablakokban lévő funkciók nem felelnek meg az adott menüpontnak, pl. a grafikon ikonja van a „Digitális vezérlő” ablakban, akkor a case struktúrára jobb egérgombbal kattintva a „Swap Diagram with Case” menüponthoz a sorrend megváltoztatható.



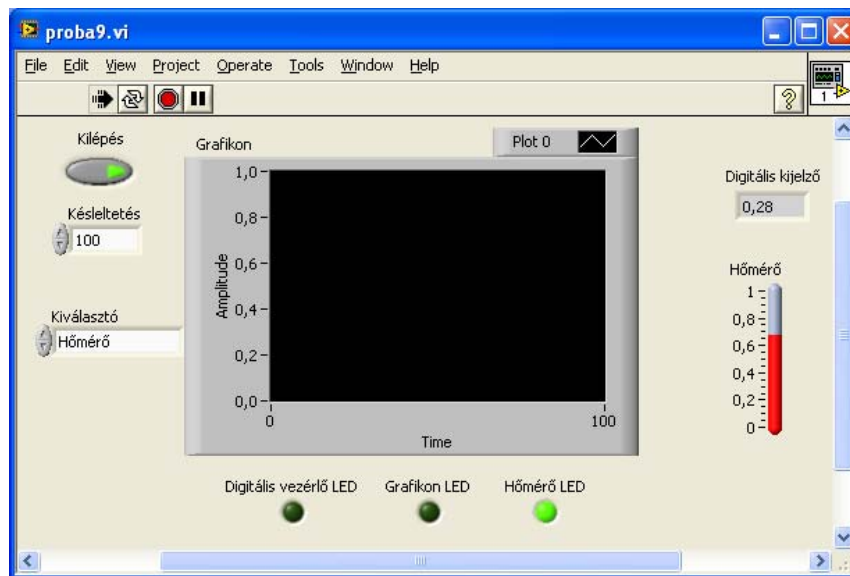
4-28. ábra Front Panel: proba8.vi



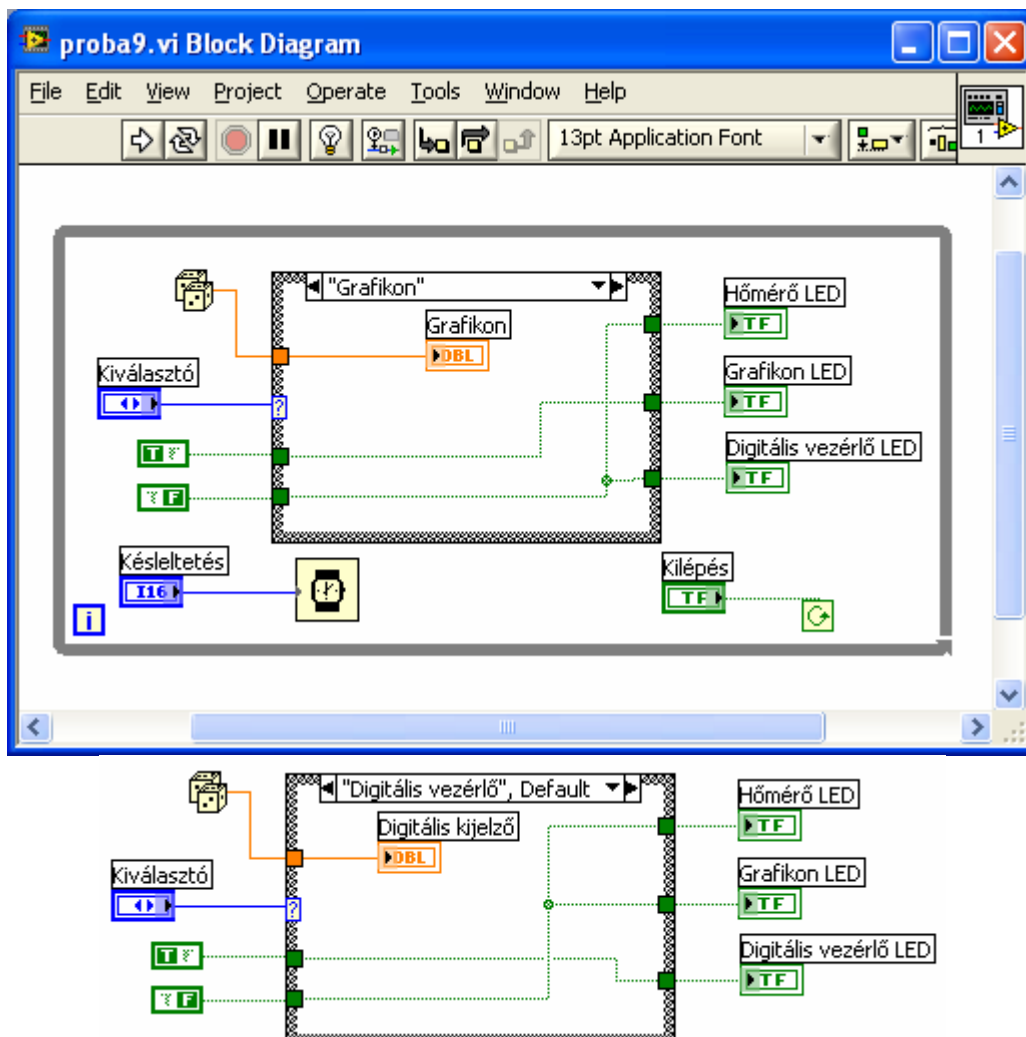
4-29. ábra Block Diagram: proba8.vi

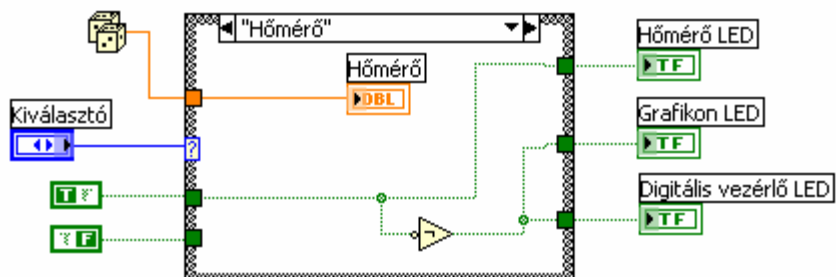
1.4.9. proba9.vi

Egészítsük ki a 8. programot 3 LED-del. Feliratozzuk őket sorra az alábbiak szerint: „Digitális vezérlő LED”, „Grafikon LED” és „Hőmérő LED”. Mindig az a LED világítson, amelyik az éppen aktuális menüponthoz tartozik.



4-30. ábra Front Panel: proba9.vi





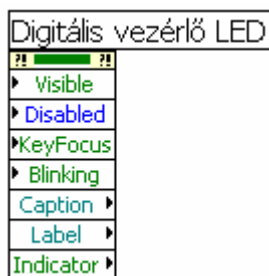
4-31. ábra Block Diagram: proba9.vi (A case struktúra 3 ablakával)

1.4.10. proba10.vi

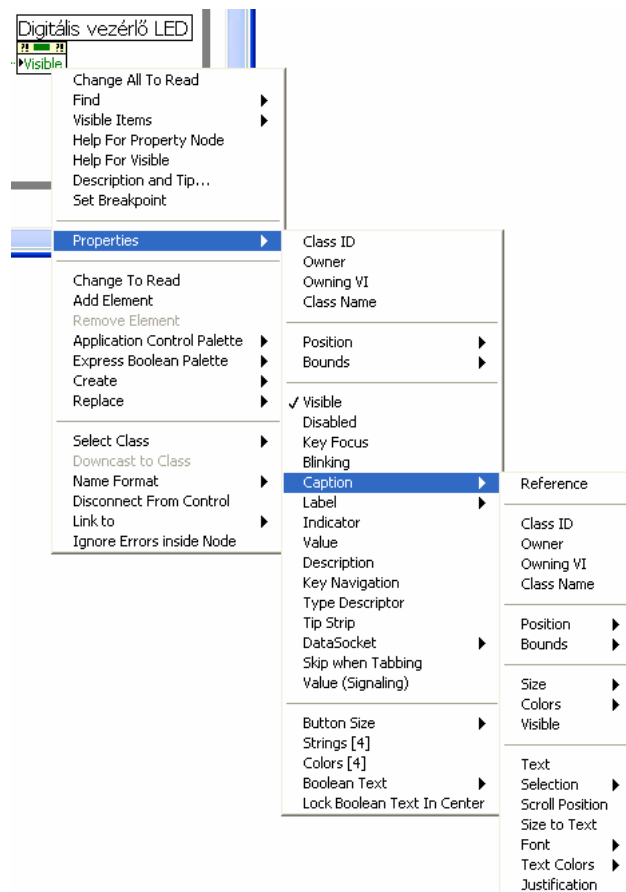
Próbáljuk ki, hogyan lehet megjeleníteni, vagy láthatatlanná tenni a panelon egy objektumot.

Alakítsuk át úgy a 9. mintaprogramot, hogy csak az a LED legyen látható a képernyőn, amelyik éppen világít, a másik kettő pedig tűnjön el a képernyőről.

Minden, panelre felhelyezhető objektumnak beállíthatjuk a jellemzőit. A jellemzők többségét azonban módunkban van a program futása közben is változtatni. Ehhez létre kell hozni egy ún. „Property node”-ot. A „Property node” segítségével számos jellemzőjét állíthatjuk be az egyes egységeknek (4-33. ábra). Ezek között a jellemzők között találjuk például az elhelyezkedését a panelon, a méretet, színezéseket, feliratozásokat, értékét, láthatóság, működési beállítások, stb. Minden jellemző lekérdezhető, vagy megváltoztatható. A vezérlő-kijelző rendszerhez teljesen hasonlóan, a „Change to write” és „Change to read” menüpontokban kiválasztható, hogy beállítani, vagy lekérdezni szeretnénk az adott jellemzőt. Egy „Property node”-ban több jellemzőt is kezelhetünk, mindegyik külön-külön állítható írásra vagy olvasásra. (4-32. ábra)



4-32. ábra Több jellemző beállítása-lekérdezése „Property node” alkalmazásával

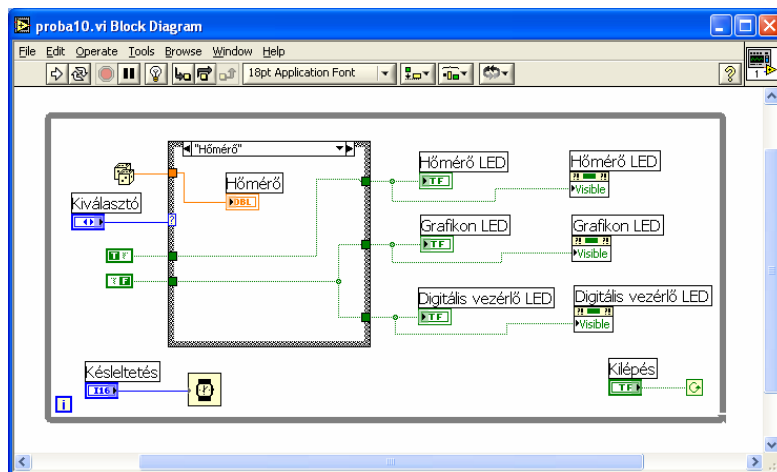


4-33. ábra Property node alkalmazása

Ismét készítsünk a LED-ekhez tulajdonság-mezőt (Property node). A tulajdonságok közül most a láthatóságot (Visible) válasszuk ki (ezzel jelenik meg, tehát nem kell átállítani). Tegyük fel a képernyőre egy kétállású kapcsolót, melynek egyik állásában látni fogjuk a LED-eket, a másikban pedig nem. Egy case struktúrával állítsuk be a true és false ablakokban a láthatóságot az előző példaprogramhoz hasonlóan.



4-34. ábra Front Panel: proba10.vi



4-35. ábra Block Diagram: proba10.vi

1.4.11. proba11.vi

Egy „Kilépés” gombbal működő while ciklusból kiindulva tegyük a panelre egy grafikon (Waveform Graph) és egy tömb változót.

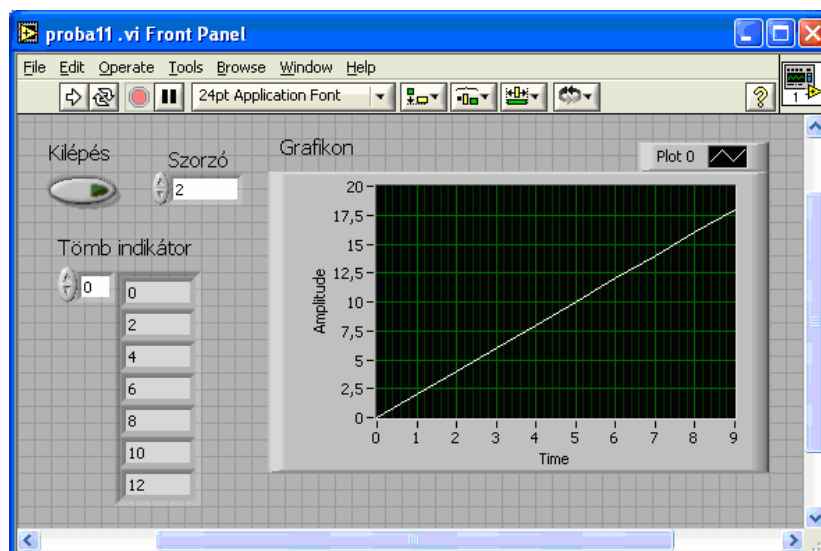
Tegyük fel a panelre egy double típusú digitális indikátort.

Helyezzük bele a digitális indikátort a tömbbe.

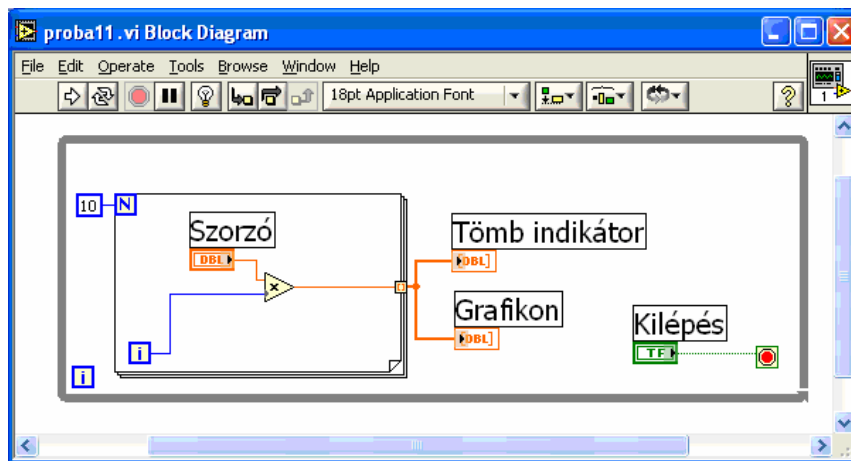
Tegyük fel a panelre egy digitális vezérlőt „Szorzó” felirattal.

A diagramm oldalon készítsünk egy 10-szer ismétlődő „for” ciklust.

A ciklus aktuálisan futó sorszámát szorozzuk meg a szorzóval, majd a for ciklus után rajzoljuk ezt ki a grafikonra, ill. írjuk ki egy digitális tömbbe is.



4-36. ábra Front panel proba11.vi



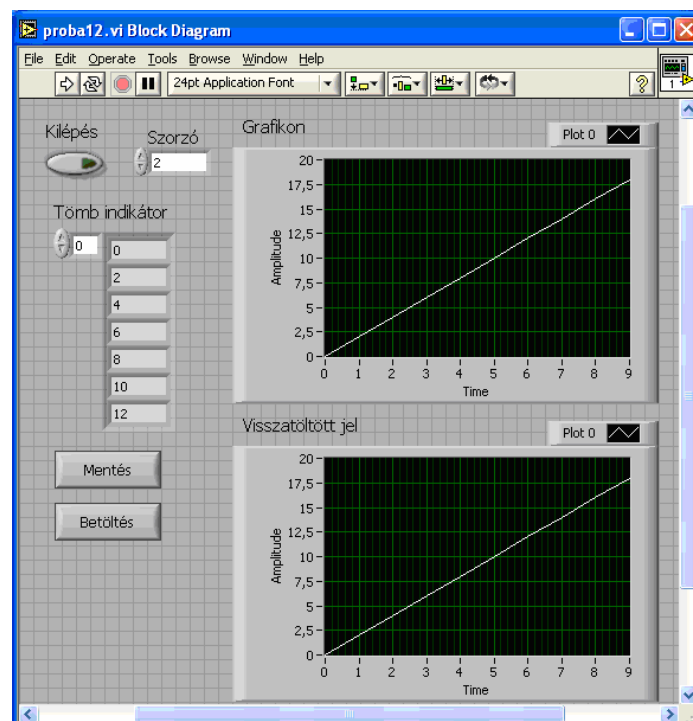
4-37. ábra Block diagram proba11.vi

1.4.12. proba12.vi

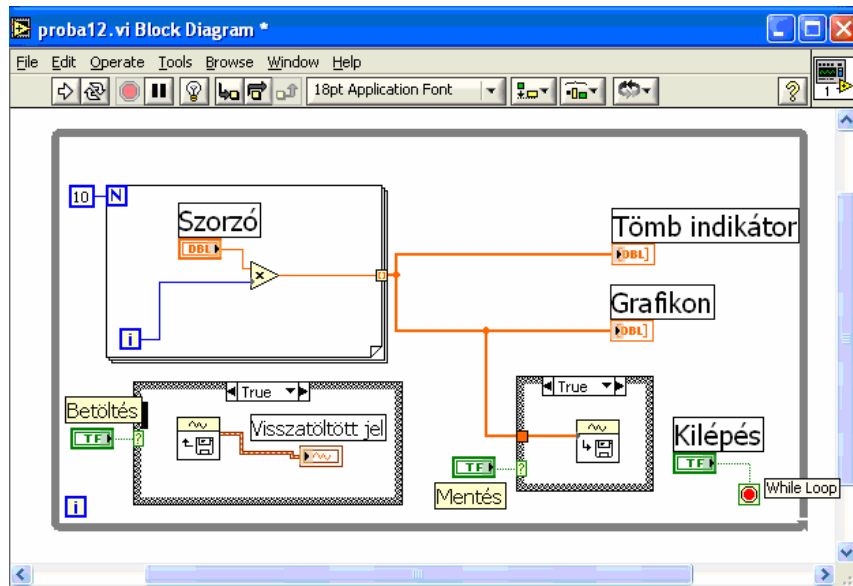
A 11. próba programot alakítsuk át úgy, hogy

1. legyen benne egy „Mentés” feliratú gomb, amelynek megnyomása esetén a generált jelet el lehet menteni egy, a felhasználó által meghatározott nevű adatfájlba;
2. legyen benne egy „Betöltés” feliratú gomb, amelynek megnyomása esetén a felhasználó által kiválasztott adatfájlból visszaolvassa a program az adatokat és kirajzolja egy második grafikonra.

Ennek a feladatnak a megoldásánál figyeljünk oda arra, hogy a mentés és betöltés nyomógombjai “csengő” típusú kapcsolók legyenek, vagyis csak addig legyenek bekapcsolt állapotban, amíg a felhasználó azt benyomva tartja (switch until released)



4-38. ábra Front panel proba12.vi

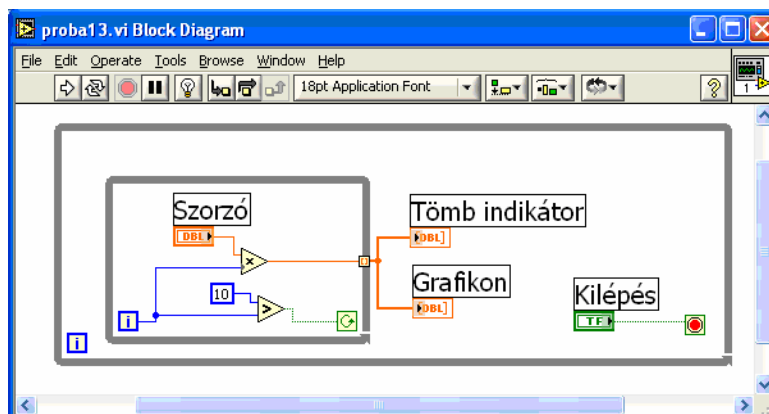


4-39. ábr:a Block diagram proba12.vi

1.4.13. proba13.vi

Készítsük el a proba11.vi programban leírt feladatot for helyett while ciklussal.

A panel megegyezik a proba11.vi paneljével.



4-40. ábra: Block diagramm proba13.vi

1.4.14. Próba feladatok

A fent közreadott bevezető mintaprogramok a LabView programozás legelső lépéseit tartalmazzák. Az alapfeladatok elvégzése után az alább megtalálható próbafeladatokat oldhatjuk meg.

1. feladat

1. Elindulás után egy „Kilépés” feliratú gomb megnyomására leáll a program.
2. Egy számbeviteli mezőn -360 és +360 fok között legyen állítható a szög értéke. A megadott tartományra korlátozza le a bevittet (az egér jobb billentyűjére előugró menüben a „Data range” menüpont alatt állítható)
3. Egy számkijelzőn jelenítse meg a beviteli mezőn beállított szög szinuszt.
4. Tegyen fel a képernyőre egy kétállású kapcsolót.
5. A kapcsoló egyik állásában a beállított szög szinuszt, a másikban a koszinuszt jelenítse meg a kijelzőn.

6. Tegyen fel a képernyőre 2 db LED-et, az egyiknek „Szinusz”, a másiknak „Koszinusz” nevet adjon.
7. Mindig az a feliratú LED világítson, amelyik értéket a kijelző mutatja.

2. feladat

1. Legyen egy while ciklusban működő „Kilépés” gomb.
2. Készítsen 4 pontból álló legördülő menüt: Szinusz, Négyszög, Háromszög, Fűrész
3. Generálja le a program és egy grafikonon jelenítse meg a menüben kiválasztott jelalakot. (Jelgenerálás: Diagram Function palette: Waveforms, Waveform generation, Sine waveform, Square waveform, Triangle waveform, Sawtooth waveform)
4. Legyen a képernyőn digitális vezérlőkkel változtatható a jel amplitúdója, frekvenciája és négyszögjel esetén a kitöltési tényezője.

3. feladat

Rövid leírás: A program rajzolja ki a felhasználó által kiválasztott trigonometrikus függvényt . 0-360 fok közötti szögek szinuszából, vagy koszinuszából, vagy -80 és +80 fok közötti szögek tangenseiből rajzol grafikont. a felhasználó egy digitális vezérlővel állíthassa be a görbe pontjainak a számát is, vagyis, hogy hány pontból készüljön a grafikon.

Részletes leírás:

1. Legyen egy while ciklusban működő „Kilépés” gomb.
2. Használjon egy digitális vezérlőt 1-100 közötti integer típusú adatok bevitelére.
3. Tegyen a panelre egy grafikont és egy Enum típusú legördülő menüt.
4. Egy for ciklus segítségével készítsünk olyan ábrát, amely a digitális vezérlőn beállított számú pontból ábrázolja a 0-360 vagy -80 - +80 fok közötti szögek megfelelő trigonometrikus értékét. A trigonometrikus függvényt a legördülő menüről lehessen kiválasztani.
5. A beállított számértéknek megfelelően ossza el a program a 360 fokot arányosan, minden pontban számolja ki a megfelelő szög megfelelő trigonometrikus függvény szerinti értékét, gyűjtse össze egy tömbbe a kiszámolt értékeket, majd a for ciklus lefutása után rajzolja ki az elkészült tömböt egy grafikonra.
6. Legyen a képernyőn egy „Mentés” és egy „Betöltés” feliratú gomb. A gombok megnyomása esetén mentse el, vagy töltsse vissza az elmentett fájlból az adatokat a program. A viszatöltött adatokat rajzolja ki egy másik grafikonra.

4. 1. feladat

1. Tegyen fel a Front panelre két kétállású kapcsolót és 3 LED-et, amelyek közül kettő a kétállású kapcsoló állapotát fogja jelezni (S és R jelöléssel), a harmadik a digitális kapu kimenetét (Q jelöléssel).
2. Tegyen fel a Front panelre egy „Enum” típusú legördülő menüt, amelynek 4 eleme van: AND, NAND, OR, NOR.
3. Tegyen fel a Front panelre egy táblázatot, amelynek 17 sora és 4 oszlopa van, az alábbiak szerint:

	S	R	Q
AND			
NAND			
OR			

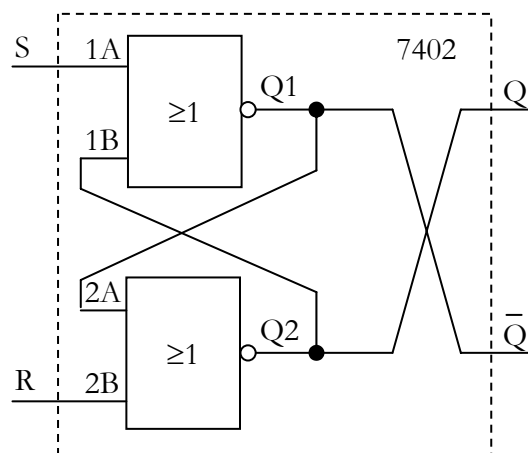
NOR			

4-1. táblázat

4. Készítse el az AND, OR, NAND és NOR kapuk szimulációját a legördülő menü és egy 4 ablakos Case struktúra segítségével és futtatáskor töltsse ki a táblázatot a függvények igazságtáblázatával.
5. Futtatás után, a kitöltött igazságtáblázattal nyomja meg a „Make current values default” menüpontot, és ezután mentse el a VI-t.

4.2. feladat

RS tároló megvalósítása NEM-VAGY kapukkal

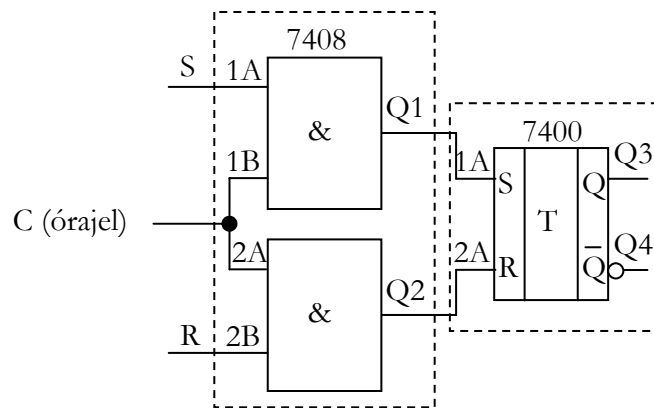


4-41. ábra R-S tároló

1. Állítsa össze az áramkör szimulációját LabView-ban és vizsgálja meg működését a digitális kimenetek és a LED-ek segítségével. Kössön kétállású kapcsolókat és LED-eket az S-re és az R-re, valamint LED-eket a Q-ra és a Q negált lábra. Készítsen az 1. feladathoz hasonló módon igazságtáblázatot!
2. Törölje a Front panelről a táblázatot, és mentse el a programot RS1.VI néven.
3. Készítsen az RS1.VI-ból egy saját ikont! A továbbiakban használja ezt az ikont a kapcsolásokhoz. Figyeljen arra, hogy mindig annyi db különböző vi-t kell használnia, ahány RS tárolót tartalmaz a kapcsolás! Nevezze el ezeket sorra RS1.VI, RS2.VI,...stb. néven.

4. 3. feladat

Kapuzott RS tároló



4-41. ábra Kapuzott R-S tároló

Az RS1.VI alkalmazásával állítsa össze az áramkör szimulációját LabView-ban és vizsgálja meg működését a digitális kimenetek és a LED-ek segítségével. Kössön kétállású kapcsolókat és LED-eket az S-re és az R-re, valamint LED-eket a Q1, Q2, Q3 és Q4 lábakra. Készítsen igazságtáblázatot a megszokott módon!

4.4. feladat

A mérés előkészítése: 8db RS tároló (RS1.VI, RS2.VI,RS8.VI) bemásolása egy külön mappába

Készítse el az alábbi digitális áramkörök szimulációját:

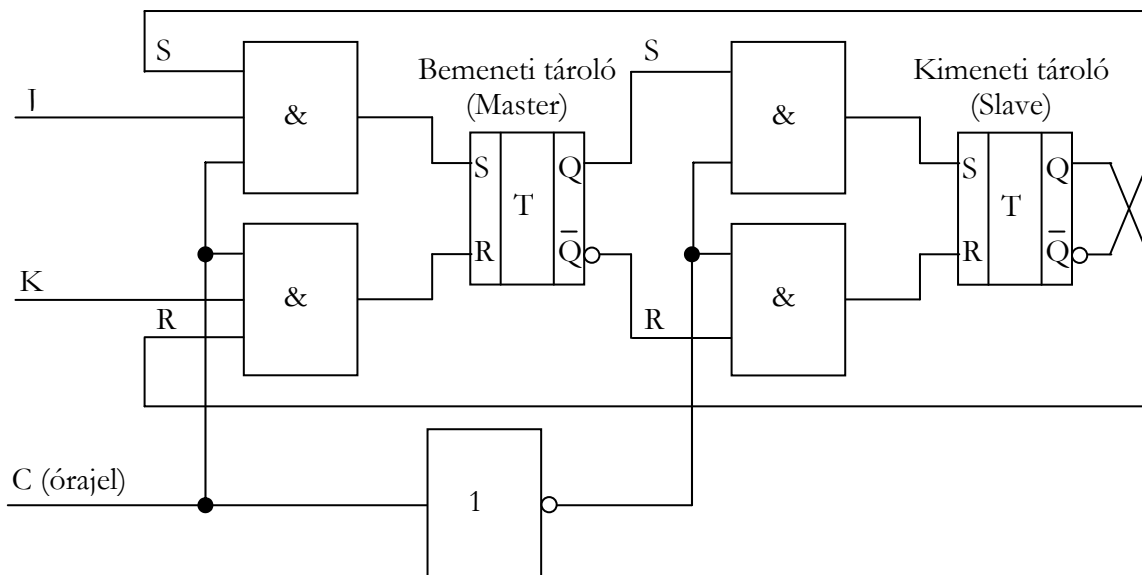
4.4.1. feladat:

2 db RS tároló felhasználásával készítse el a JK tároló programját.

Bemenet: J,K,C (kapcsoló+LED)

Kimenet: Q és Q negált (LED)

Igazságtáblázat



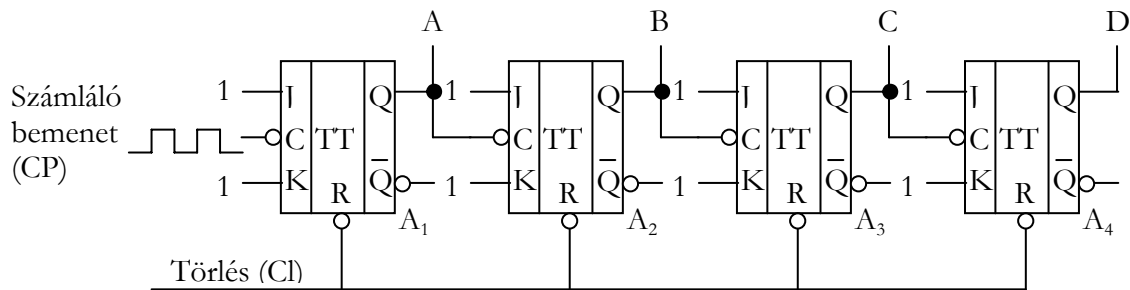
4-42. ábra J-K tároló

Törölje az igazságtáblázatot a panelről, és mentse el a programot JK1.VI néven.

8 db RS tárolóval készítsen 4 db JK ikont. Az ikonokat mentse el egy külön mappába.

4.4.2. feladat

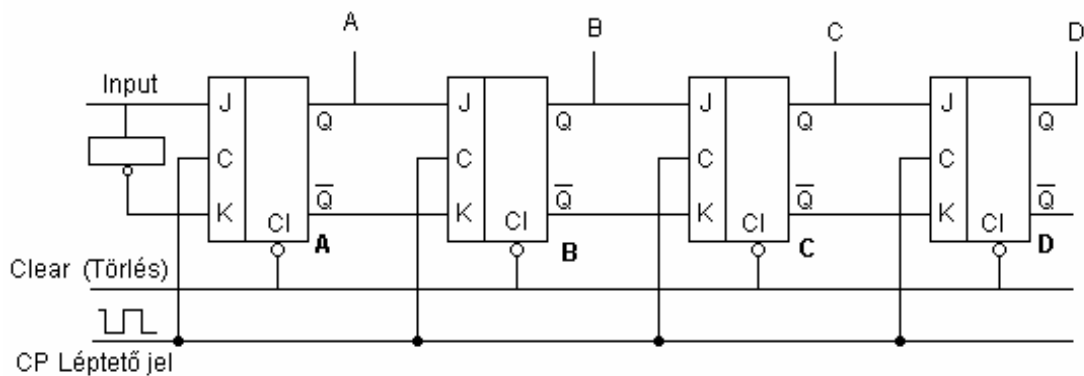
A 4 db JK ikon felhasználásával készítse el az alábbi bináris számláló szimulációját. A törlést nem kötelező felprogramozni.



4-43. ábra Bináris számláló

4.4.3. feladat

A 4 db JK ikon felhasználásával készítse el az alábbi léptető szimulációját. A törlést nem kötelező felprogramozni.



4-44. ábra Léptető regiszter

5. feladat

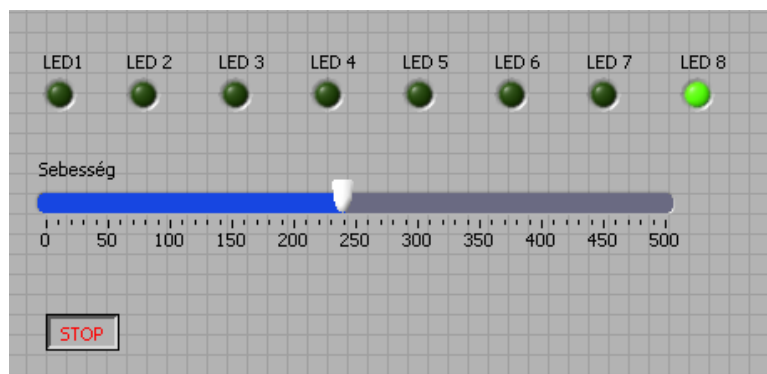
Készítsen olyan programot, amely egy 8 db LED-ből álló futófényt működtet.

Egy tolóbeállítóval lehessen változtatni a futófény sebességét 0-500 ms között.

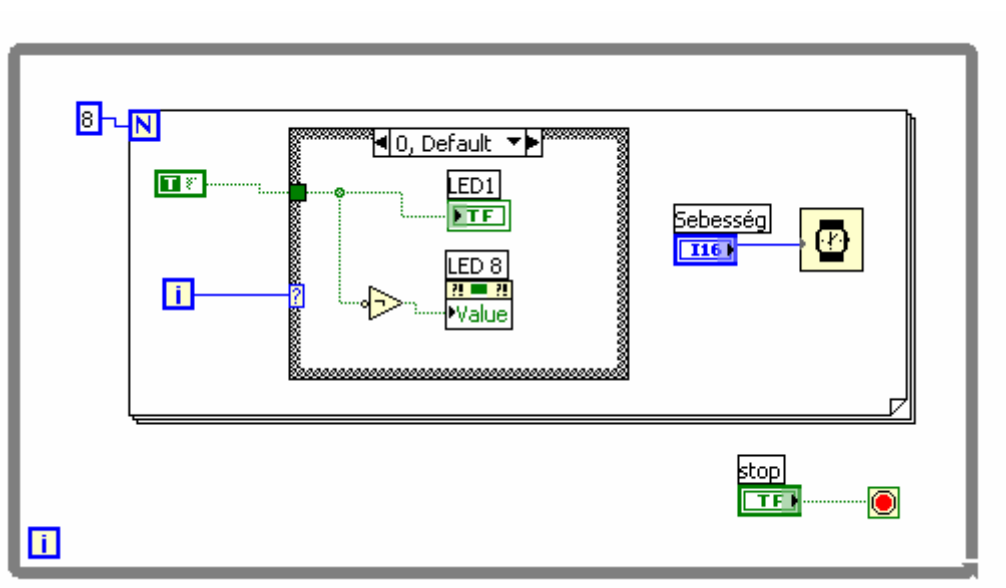
Kilépés gombbal lehessen leállítani a programot.

(Futófény: sorra gyulladnak meg a LED-ek egymás után, és amikor az egyik LED meggyullad, az előtte lévő elalszik. A tolóbeállítóval 1 LED világítási idejét állítjuk be.)

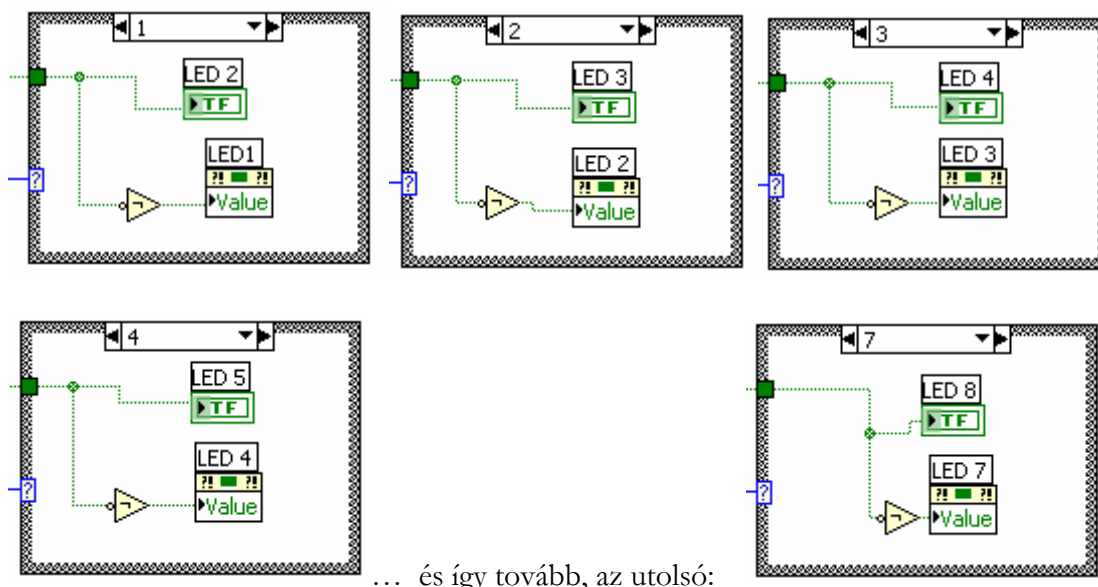
Megoldás:



4-45. ábra Futófény panelje



4-46. ábra Futófény blokkdiagramja



4-47. ábra Futófény blokkdiagramjának case ablakai

6. feladat

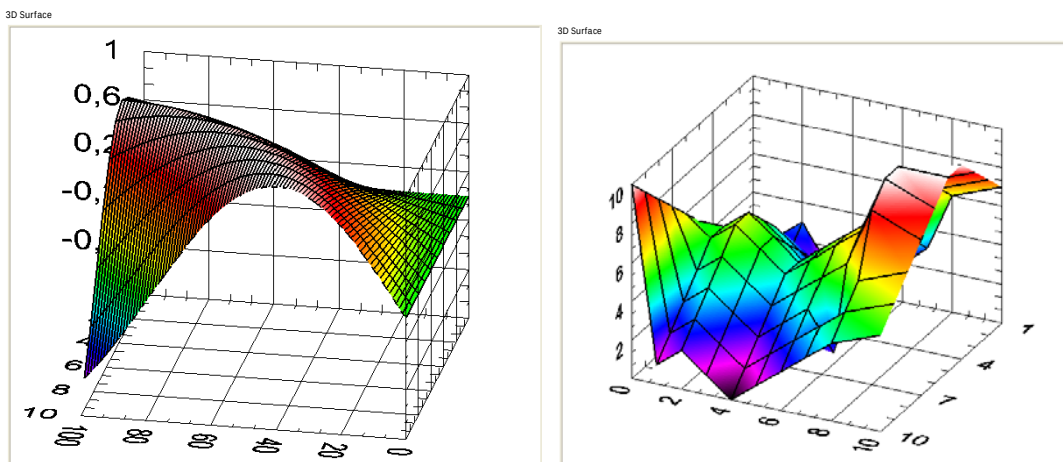
Készítsen egy olyan programot, amely a „3D Surface Graph” felhasználásával tetszőleges 3D-s felületet rajzol ki.

Az X és az Y koordináták min. 10-10 pontból, max. 100-100 pontból álljanak (válassza egyforma méretűre ezt a két tengelyt! Ezek 1D-s tömbökben tárolt adatok) A Z koordináta X*Y mintát kell, hogy tartalmazzon, egy 2D-s tömbben. HASZNÁLJA A HELPET!!

A 3D-s grafikon beállításánál a lenti mintához hasonló kép készítéséhez használja a „CWGraph3D” tulajdonságok menüpontban a „Plots”, „Plot Styles” választási lehetőségek közül a megfelelőt.

Az X, Y és Z koordináták első 10 értékét írassa ki egy digitális tömb kijelzőre.

Minta:



4-48. ábra 3D grafikon megjelenítés LabView-ban

1.5. Mérésvezérlés LabView környezetben

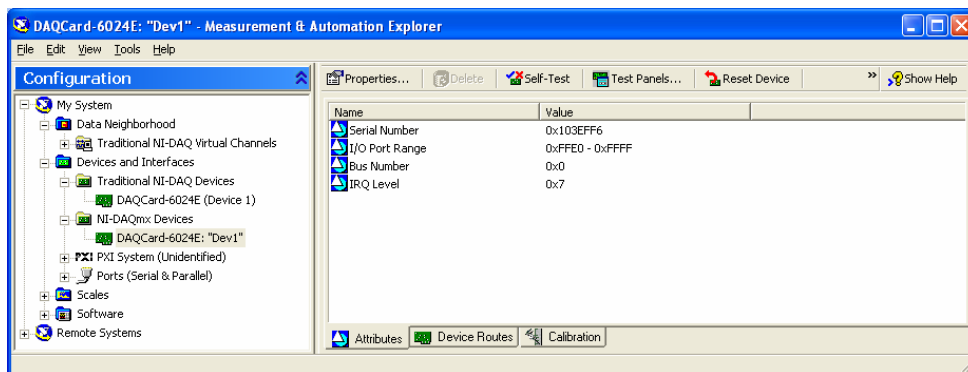
Ez a rövid összefoglaló a jegyzet 3.8.2 fejezetéhez kapcsolódik, a többfunkciós mérésadatgyűjtő kártyák analóg bemenetének programozását mutatja be.

Amint azt a 3.8.2. fejezetben bemutattuk, mérhetünk rövid ideig nagy mintavételi frekvenciával, ekkor a mérendő mintaszámot előre be kell állítani, vagy mérhetünk folyamatosan hosszú ideig alacsonyabb mintavételi frekvenciával, ekkor indítástól egy külső leállításig futhat a mintavételezés. Mindkét módszer alkalmazható multiplexerrel vagy anélkül, vagyis egycsatornás vagy többcsatornás módon.

Ezeknek a módszereknek a LabView környezetben történő alkalmazását mutatjuk be sorra a következő alfejezetekben.

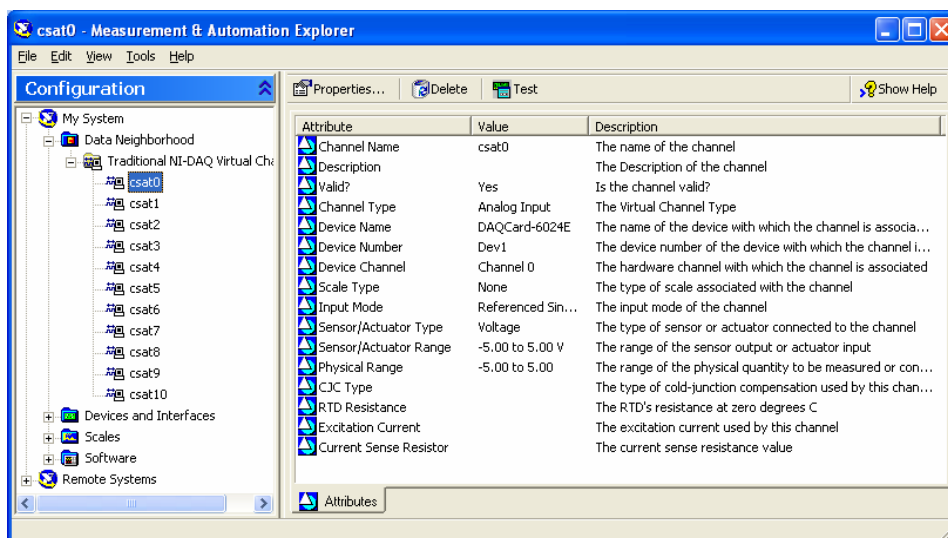
Mielőtt elkezdjük a programozást, szóljunk néhány szót a hardverről.

A mérésadatgyűjtőnkét installálás után a *Measurement & Automation Explorer*-ben (MAX) konfigurálhatjuk és ellenőrizhetjük.

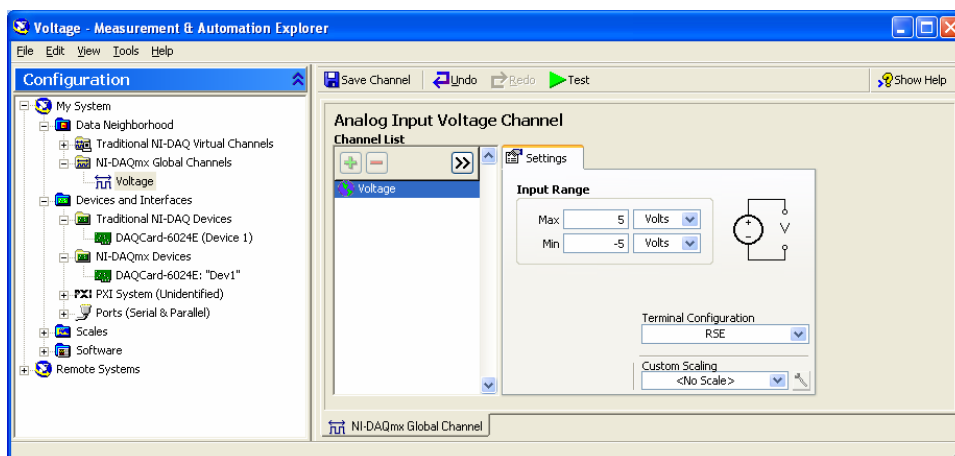


4-49. ábra Measurement & Automation Explorer

Ahhoz, hogy LabViewban mérést vezéreljünk, un. virtuális csatornákat kell létrehozni. Ezt vagy a *Measurement & Automation Explorer*-ben (MAX), vagy a DAQmx könyvtár megfelelő függvényével tehetjük meg.



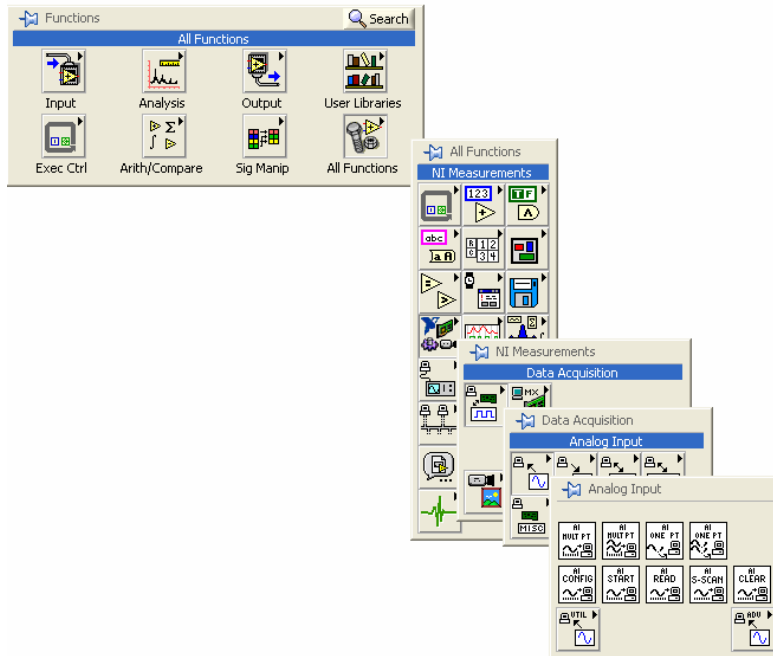
4-50. ábra Tradicionális virtuális csatornák konfigurálása a MAX-ban



4-51. ábra DAQmx virtuális csatornák konfigurálása MAX-ban

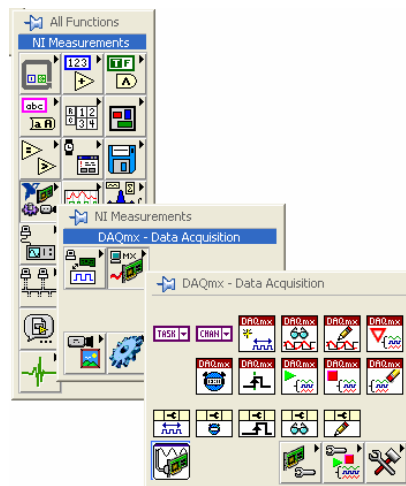
Ha a kártya és a virtuális csatornák megfelelően működnek, elkészíthetjük LabView-ban a vezérlő szoftvert.

Az, hogy milyen mérésadatgyűjtő függvényeket alkalmazunk a hardvertől is függ. A hagyományos és az “DAQmx” típusú berendezések programozásban is különböznek. A hagyományos kártyák programozása a kezdeti lépéseknél bonyolultabb, mert a függvények nem “szolgálják ki” olyan mértékben a programozót, mint a DAQmx esetében. A hagyományos adatgyűjtők függvényeit a NI Measurement könyvtárban találhatjuk meg:



4-52. ábra Hagyományos adatgyűjtők mérőfüggvényeinek könyvtára

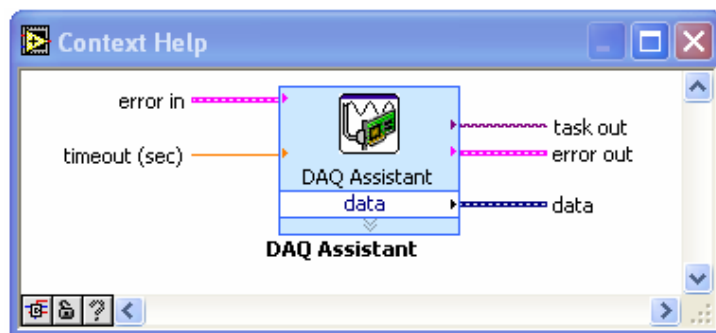
Az újabb generációs adatgyűjtők függvényei a DAQmx könyvtárban találhatóak:



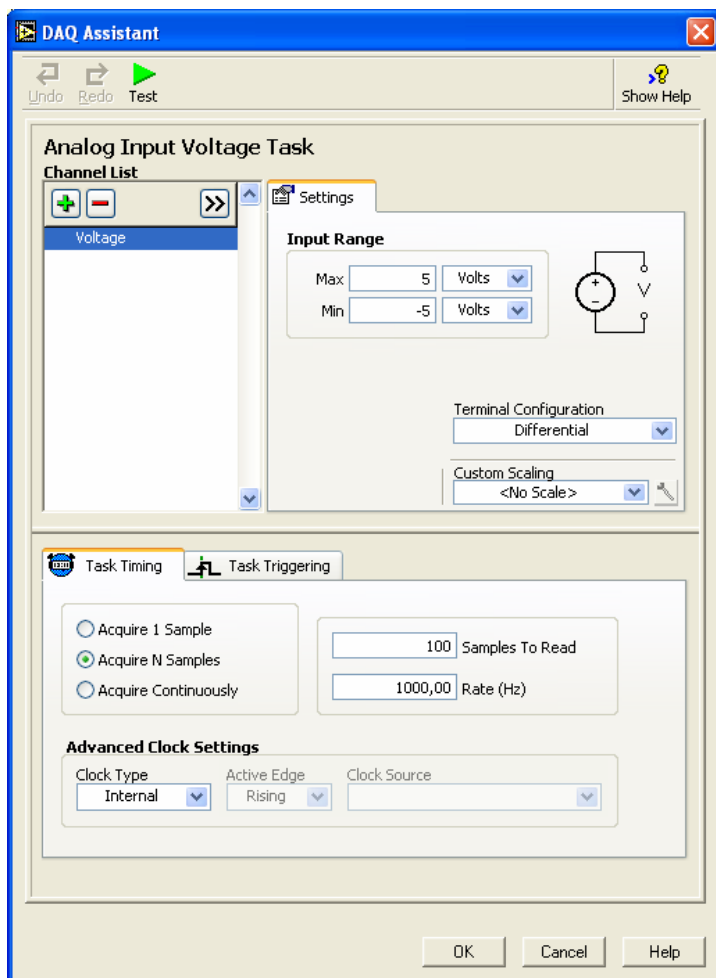
4-53. ábra DAQmx könyvtár

A DAQmx rendszere lényegesen jobban kiszolgálja a felhasználót, mint a hagyományos függvények, a DAQ Assistant alkalmazásával egy ablakban gyakorlatilag minden alapadatot beállíthatunk a mintavételezéshez, és elvégezhetjük a mérést. Ez a lehetőség egyik oldalról nagy segítség a kezdőknek és az alapfeladatokat alkalmazó haladóknak is, hiszen a lehető legteljesebb mértékben próbálja “kiszolgálni” a felhasználót, másrésztől a bonyolult és összetett mérési-vezérlési feladatoknál gyakran alacsonyabb szinten is szükség van arra, hogy a folyamatokba

beavatkozunk, vagy kövessük azokat, ami az mx rendszer alkalmazása esetén komoly nehézségeket okoz.



4-54. ábra DAQ Assistant az mx rendszerben működő mérések inicializálásához



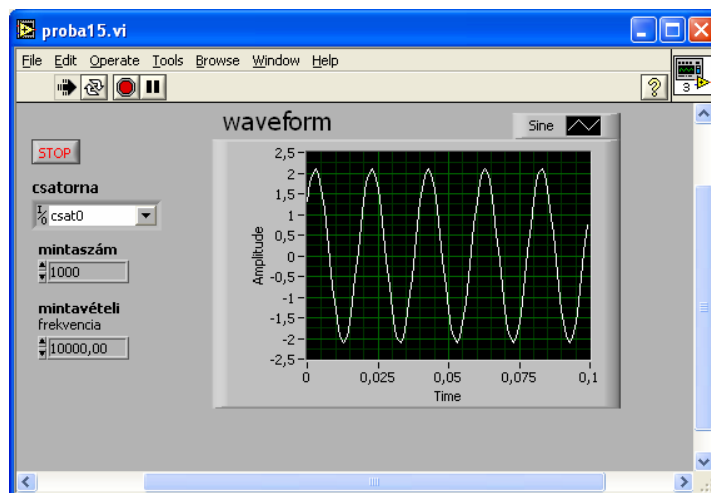
4-55. ábra DAQ Assistant konfigurálása

A hagyományos kártyák programozása esetén a különböző módszereket különböző függvényekkel tudjuk alkalmazni.

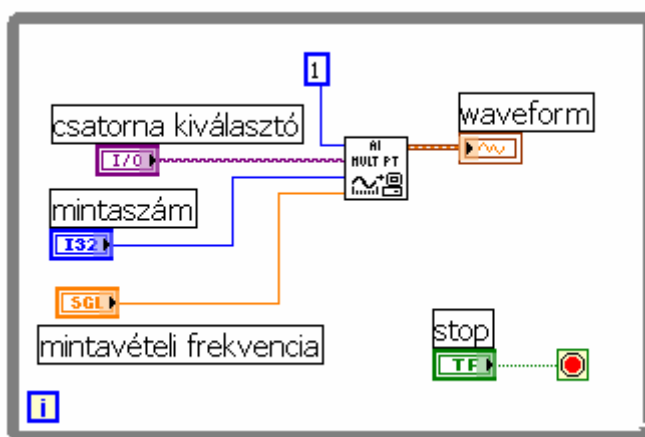
1.5.1. Egycsatornás rövid mintavételezés

Egycsatornás mintavételezéshez az "Acquire Waveform.vi" függvényt alkalmazzuk. A MAX-ban létrehozott virtuális csatornák közül választhatunk az I/O könyvtárban található

csatornakiválasztó segítségével. A mintaszám integer, a mintavételi frekvencia double típusú változó.



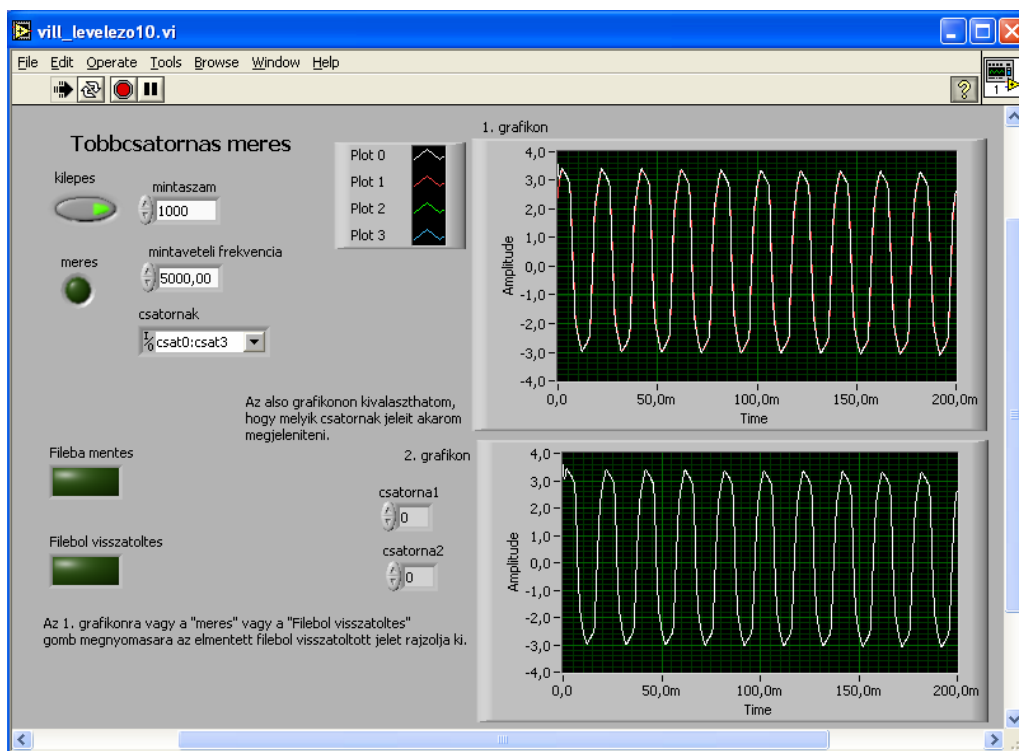
4-56. ábra Egycsatornás mintavételezés panelje



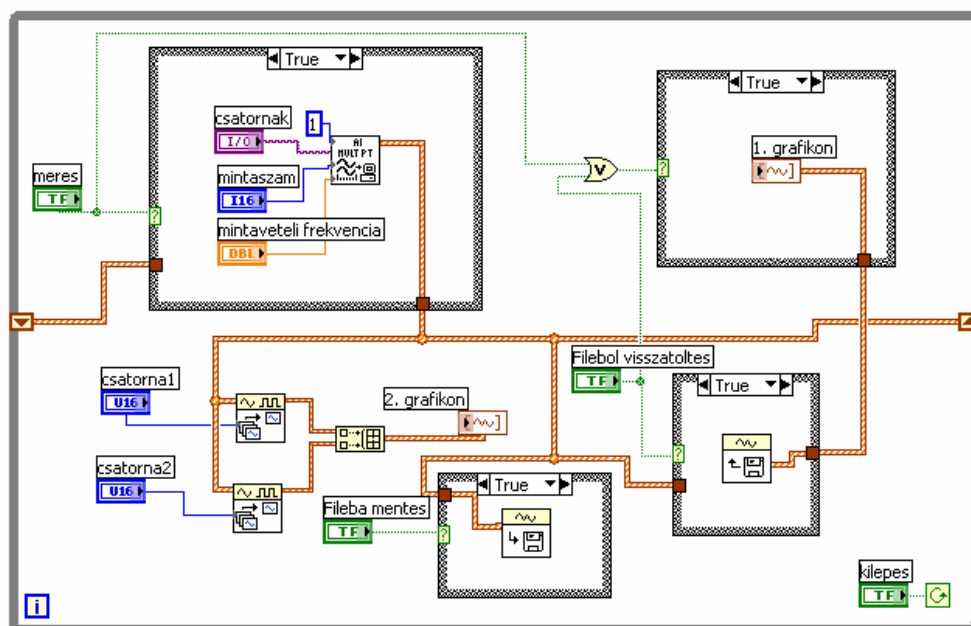
4-57. Egycsatornás mintavételezés blokkdiagramja

1.5.2. Többcsatornás rövid mintavételezés

A többcsatornás mintavételezés teljesen hasonló az egycsatornáshoz, azzal a különbséggel, hogy a függvény amit használunk az "Acquire waveforms.vi", ami már használja a multiplexert is. A függvény paraméterezése teljesen hasonló az előzőhöz, arra kell csupán figyelni, hogy a csatornakiválasztóba vagy vesszővel elválasztva soroljuk fel a csatornákat, vagy kettősponttal, ha től-ig kívánjuk megadni a mérendő csatornákat. A 4-58. és 4-59. ábrán egy olyan hallgatói mérést mutatunk be, amely többcsatornás mérés elvégzésére szolgál, különböző megjelenítési megoldásokat mutat be, valamint a jel elmentését és fájlból történő visszatöltését is lehetővé teszi. A bemutatott futtatáson 4 csatornás mérést látunk.



4-58. ábra Többcsatornás mérés panelje



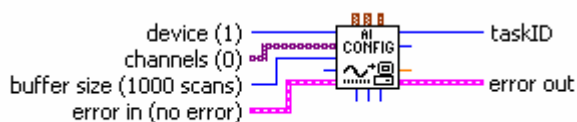
4-59. ábra Többcsatornás mérés blokkdiagramja

1.5.3. Hosszúidejű folyamatos mintavételezés

Ha a rövid idejű de nagy sebességű mintavételezés nem megfelelő a mérési feladat elvégzésére, akkor mérhetünk folyamatosan, de ebben az esetben a szinkronizálási problémára különös figyelmet kell fordítani. Ekkor a mintavételi frekvencia lényegesen alacsonyabb lehet, mint az előző két módszer alkalmazása esetén.

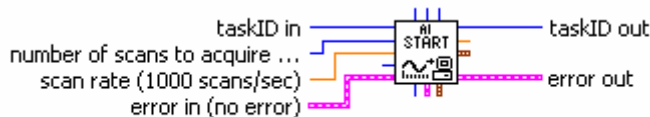
A folyamatos mintavételezés vezérlése az alábbi folyamat szerint történik:

1. A mintavételezés konfigurálása



AI Config.vi

2. A mintavételezés elindítása



AI Start.vi

3. A minták szinkronizált kiolvasása a tárolóból



AI Read.vi

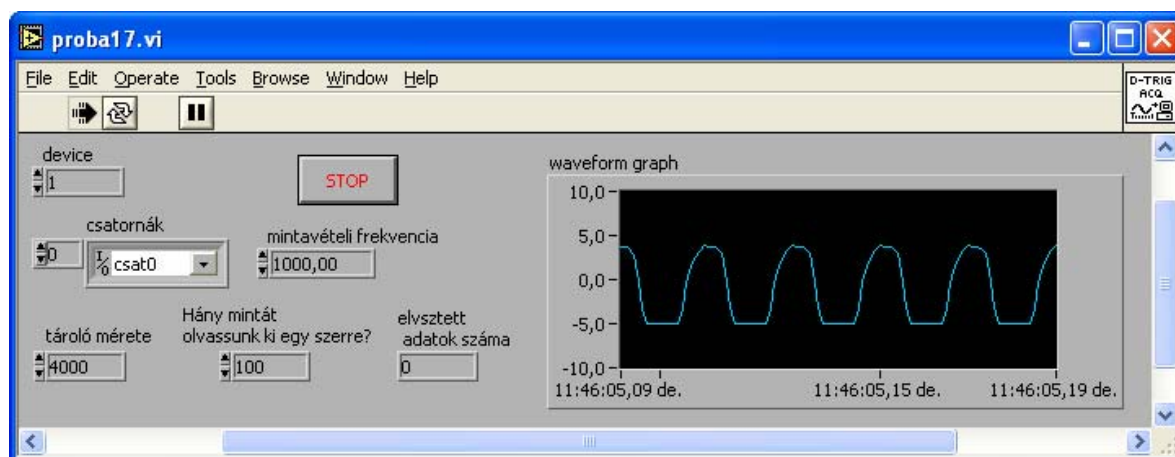
4. A mérés végén a mintavételezés leállítása



AI Clear.vi

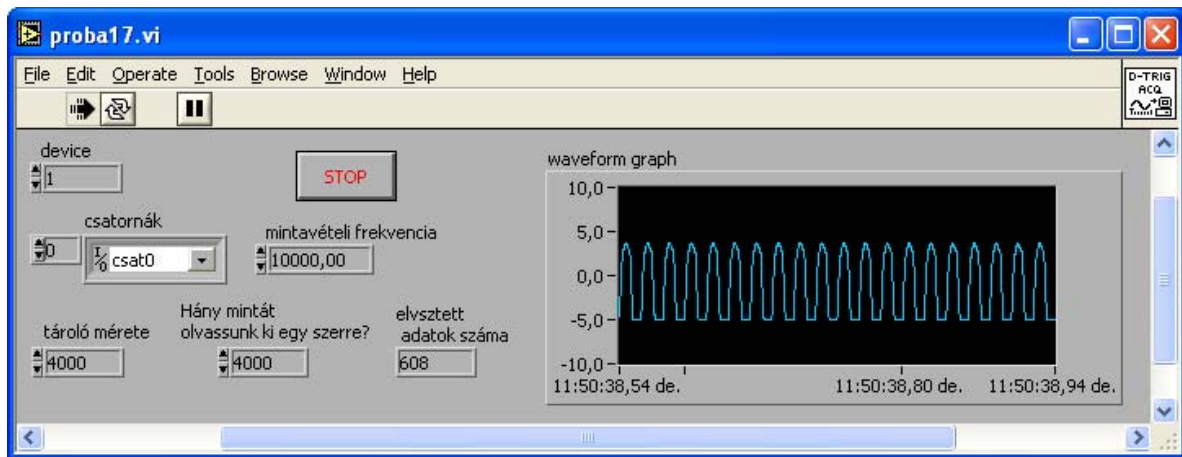
Ennek a programnak a jól követhető megvalósításához kivételesen nem egy while ciklussal kell kezdeni a programozást. Ennek az oka, hogy a mintavételezést nem szabad többször elindítani, csak egyszer. Ha a mérés egy nagy rendszer integrált része, akkor nagy figyelmet kell fordítani arra, hogy automatikus újraindulás ne történhessen.

A mérés blokkdiagramját a könnyebb érthetőség kedvéért vázlatosan mutatjuk be a 4-60-62. ábrákon. Bár a program így is működőképes, a valós mérőprogramban a pontos és biztonságos működtetés érdekében az itt be nem állított paramétereket is tanácsos használni.

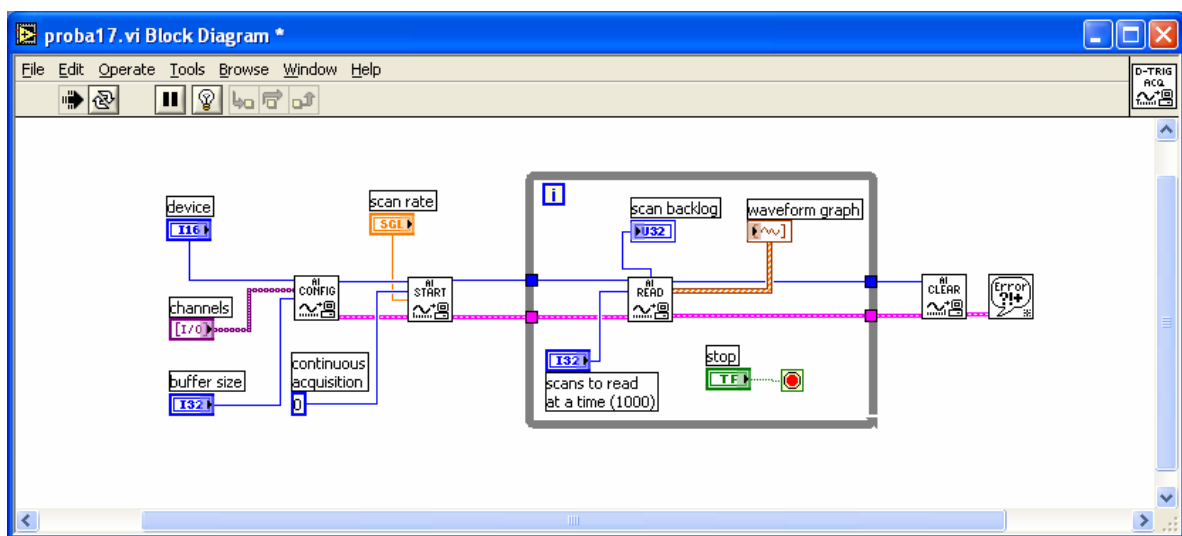


4-60. ábra Folyamatos mintavételezés panelje (jól szinkronizált mérés)

Az elvesztett adatok száma mutatja, hogy a mérést helyesen szinkronizáltuk-e. Ha ez a mennyiség 0-tól eltér, akkor a mintavételi frekvencia, vagy a tároló méretének változtatásával javíthatjuk a szinkronizálást. A 4-61. ábrán azt mutatjuk be, hogy 10.000 Hz-es mintavételi frekvencia alkalmazásakor 4000 mintából 608-at elvesztünk, mert nem tudja olyan gyorsan a rendszer kiolvasni a tárolóból az adatokat, mint amilyen gyorsan beérkeznek azok.

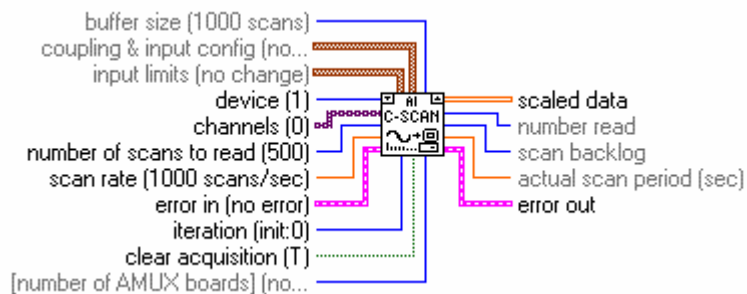


4-61. ábra Rosszul szinkronizált folyamatos mintavételezés



4-62. ábra Folyamatos mintavételezés egyszerűsített blokkdiagramja

A 4-62. ábrán bemutatott programban alkalmazott 4 függvényt a LabViewban összevontan 1 függvényben is megtaláljuk:



Az "AI Continuous Scan.vi" összefoglalja a konfigurációs, mérés indító-kiolvasó és leállító függvényeket, ennek megfelelően a paraméterezése nagyon komplex és hozzáértést igénylő feladat, csak a gyakorlott programozóknak ajánljuk.