

Beágyazott rendszerek fejlesztése laboratórium

Virtuális műszerek méréssorozat

LV0: LABVIEW ALAPISMERETEK

Mérési útmutató

V0.1

BME MIT 2018.

1.	BEVEZETÉS	3
1.1	VIRTUÁLIS MŰSZEREK	3
1.2	A GRAFIKUS PROGRAMOZÁSRÓL.....	5
2.	ALAPISMERETEK	6
2.1	LABVIEW VI FUTTATÁSA, ALAPFOGALMAK.....	6
2.2	STÁTUS- ÉS ESZKÖZSOR	7
2.3	EGY EGYSZERŰ PÉLDAPROGRAM LÉTREHOZÁSA.....	9
2.4	VEZÉRLÉSI SZERKEZETEK (STRUCTURES) ÉS ADATTÍPUSOK	12
2.4.1	<i>Ciklusok</i>	12
2.4.2	<i>Sorrendi struktúrák</i>	15
2.4.3	<i>Választás (Case) struktúra</i>	16
2.5	EGYÉB FONTOS LABVIEW ELEMELK.....	16
3.	ISMERKEDÉS A VIRTUÁLIS MŰSZEREKEL	18
3.1	A NATIONAL INSTRUMENTS MŰSZERKÍNÁLATÁNAK ÁTTEKINTÉSE.....	18
4.	LV0 MÉRÉSI FELADATOK	21
4.1	FAHRENHEITRŐL CELSIUSRA KONVERTÁLÓ VI KÉSZÍTÉSE	21
4.2	CIKLUSOK VIZSGÁLATA	22
4.3	VIRTUÁLIS FÜGGVÉNYGENERÁTOR.....	22
4.4	FÜGGVÉNYGENERÁTOR MEGVALÓSÍTÁSA A PC HANGKÁRTYÁJÁVAL	23
4.5	OPCIONÁLIS FELADAT: REAKCIÓIDŐ-MÉRŐ	24

1. BEVEZETÉS

1.1 Virtuális műszerek

Műszerek A gyors technológiai fejlődés ellenére az elektronikus műszerek funkcionális felépítése lényegében nem változott az elmúlt évtizedekben. Minden műszer blokkvázlata nagyon hasonló: mindegyiknek van bemeneti egysége, feldolgozó egysége és megjelenítő egysége. A műszerek fajtájától és az alkalmazott technológiától függően, természetesen, az egységek bonyolultsága nagyon változó. Egy egyszerű műszerben a feldolgozó lehet egy analóg átlagoló, egy bonyolultabban lehet például egy FFT-t számoló célhardver; a megjelenítő lehet egy mutató műszer, de lehet egy grafikus kijelző is.

Adatgyűjtők/vezérlők Ha a műszerek körét tágítjuk, és figyelembe vesszük az adatgyűjtő/vezérlő berendezéseket is, akkor a fenti blokkvázlatot csupán kimeneti/beavatkozó egységgel kell bővítenünk.

„Műszermag” A műszerek és az (egyszerűbb) adatgyűjtő/vezérlők lényegében csupán a bemeneti és kimeneti egységeikben különböznek, a feldolgozó és megjelenítő egységeik nagyon hasonlóak, vagyis van egy jelentős közös részük. Ezt a feldolgozó, megjelenítő közös részt „műszermagnak” is nevezhetjük.

Egy univerzális „műszermag” felhasználásával sokkal gyorsabban és költséghatékonyabban tudunk műszereket, berendezéseket kifejleszteni, mint egyedi tervezéssel.

„Mikroprocesszoros...” A „műszermagok” univerzális megvalósítását a mikroprocesszorok megjelenése tette lehetővé. Az 1970-es évek végén és az 1980-as években készített műszerek neve gyakran így kezdődött: „Mikroprocesszoros...”, hasonlóan ahhoz, ahogy az 1960-as, 70-es években a „Tranzisztoros...” jelzővel mutatták a készülékek korszerűségét. (Itt jegyezzük meg, hogy a gyakran használt „korszerű” szó nem a termék/rendszer minőségére utal, hanem csupán arra, hogy az a valami az adott korszak technikai színvonalának megfelelő. Ettől az még lehet jó is, de akár rossz is.)

Beágyazott rendszerek Lényegében a mikroprocesszorok tömeges elterjedésétől kezdve beszélhetünk beágyazott rendszerekről – noha ez a kifejezés csak jóval később, nálunk nagyjából 2000-től terjedt el –, hiszen a mikroprocesszorok tették lehetővé, hogy komplex feldolgozó, vezérlő funkciókat építsünk a műszereinkbe (elfogadható áron), és így teljessüljön a beágyazott rendszerek definíciója: Beágyazott rendszerek nevezzük azokat a processzor alapú eszközöket, illetve az ezekből alkotott rendszereket, amelyek képesek a befogadó fizikai/kémiai/biológiai környezetüket érzékelők (és beavatkozók) segítségével autonóm módon megfigyelni (és befolyásolni).

Moduláris építkezés	A hardver tipizálásával, modulokra bontásával könnyen és gyorsan lehetett egyedi műszereket, készülékeket létrehozni. A hardvermodulokból összeállított berendezések testre szabása lényegében szoftverben történt. Megjegyzendő, hogy a hardver építőkockáihoz hasonlóan a szoftvert is fel lehetett bontani tipikus szoftvermodulokra, így a szoftverfejlesztés is nagyban felgyorsult.
PC-s korszak	Az asztali számítógépek és különösen a PC-k megjelenése (nálunk az 1990-es évek elejétől) a készülékek konstrukcióját is alaposan megváltoztatta. A PC lényegében egy (viszonylag) olcsó, univerzális „műszermag”, komoly processzási kapacitással, nagy memóriával és háttértárral, gazdag ember–gép kapcsolattal (klaviatúra, egér; grafikus kijelző, opcionálisan nyomtató) és többféle kommunikációs interfésszel (korábban csak soros, párhuzamos interfész, később Ethernet, USB stb.). Csupán a feladathoz illeszkedő I/O perifériákkal kell kiegészíteni, és kész a műszer vagy adatgyűjtő/vezérlő hardvere és alapszoftvere. A készülékfejlesztés nagyobbik része az alkalmazói szoftver elkészítését jelenti.
Virtuális műszerek	Az 1990-es évektől kezdve a PC-alapú készülékfejlesztés nagyjából két irányvonalat követett. Az egyik főleg a hagyományos készülékgyártókra jellemző: megőrizték a szokásos készülékformát, kezelőfelületet, és a doboz belsejébe „elrejtették” a PC-t (speciális, ipari PC-alaplapot és perifériákat). A másik irányvonal standard asztali PC-t használ, de kiegészíti analóg és digitális I/O perifériákkal és alkalmas szoftverrel. Ez utóbbi megoldás vezetett a „virtuális műszer” fogalomhoz. Az univerzális „műszermagból” (PC-ből) és a kiegészítő, szintén univerzális perifériákból sokféle konkrét „műszer” alakítható ki, attól függően, hogy éppen milyen alkalmazói szoftver fut a „műszermagon”, vagyis a számítógépen. Ugyanazon a hardver- és alapszoftver-készleten – az alkalmazói szoftver cseréjével – sokféle „műszer” implementálható: multiméter, függvénygenerátor, oszcilloszkóp, spektrumanalizátor stb. Az így előálló „műszer” valódi abban az értelemben, hogy funkcionálisan megegyezik egy hasonló célműszerrel, de „virtuális” abban az értelemben, hogy a fizikai megjelenése lényegesen eltér azokétól.
National Instruments	A „virtuális műszer” koncepció egyik vezető képviselője az amerikai National Instruments (NI). Az 1976-ban alapított cég analóg és digitális I/O perifériákat kezdett gyártani számítógépekhez majd PC-khez beépíthető kártya és külön bedobozolt egységek formájában. Az NI készülékeit – noha nem a legolcsóbbak – az iparban széles körben használják. Az NI egyik hardvergyártó bázisa Debrecenben van.
LabVIEW	A National Instruments „virtuális műszer” koncepciója alkalmas szoftverrendszer nélkül aligha ért volna el átütő sikert. A moduláris és univerzális hardverelemekhez szintén moduláris és univerzális programozási nyelvre és környezetre van szükség. Azt is figyelembe kellett venniük, hogy a felhasználók jelentős része nem programozó, nem is villamosmérnök vagy éppen informatikus, tehát a nyelv és a programozói/fejlesztői környezetnek kevés programozói előismeret birtokában is használhatónak kell lennie. Az NI többféle

szoftverrendszert is kifejlesztett a hardvereihez; a legnépszerűbb a LabVIEW (Laboratory Virtual Instrument Engineering Workbench).

A LabVIEW

adatfolyam alapú, magas szintű grafikus programozási nyelv és környezet, amely az elmúlt húsz év során jelentős fejlődésen ment keresztül. Jelenleg (2008) a 8.5.1 verziónál tartanak. Általános programozási nyelvként is használható, de igazi erősségét virtuális műszerek létrehozásakor mutatja meg, különösen NI perifériák alkalmazása esetén.

1.2 A grafikus programozásról

Szöveges programozás A villamosmérnök- és az informatikushallgatók általában szöveges programozási nyelveken keresztül ismerkednek meg a számítógépek programozásával; korábban a BASIC vagy a Pascal, ma inkább a C és a Java nyelv fogalmkészletét ismerik meg először, így a későbbiekben ezekhez viszonyítják a többi programozási nyelvet, környezetet is.

Grafikus programozás A LabVIEW egy grafikus programozási nyelv és fejlesztőkörnyezet, és mint ilyen, teljesen másfajta gondolkodásmódot kíván, mint a hagyományos procedurális nyelvek vagy az objektumorientált nyelvek. Természetesen a LabVIEW-n kívül más grafikus programozási nyelvek is léteznek, pl. Simulink (MathWorks), VEE (Agilent).

Adatfolyam-programozás A LabVIEW megalkotói előtt a műszerekre jellemző adatbeolvasás–feldolgozás–kijelzés tipikus feladatsor lebegett. A nyelv szerkezetét és elemeit ennek az adatfolyamnak a minél egyszerűbb leírhatósága határozta meg. Ebből következően a hagyományos procedurális programozási elvek és konstrukciók nem, vagy csak áttételesen, alkalmazhatók. Szintén nem alkalmazhatók közvetlenül az objektumorientált programozás fogalmai.

„Könnyű” használat

Az adatfolyam alapú grafikus programozás – és így a LabVIEW is – a feladatok egy jelentős részére rendkívül jól illeszkedik, ezért az ilyen jellegű programok megírása kifejezetten könnyű, szinte élvezetes. Természetesen, azoknak az alkalmazásoknak is jelentős a száma, amelyekre csak nehézkesen, körülményesen használható a LabVIEW. Nagyon könnyen lehet példát találni olyan alkalmazásokra, ahol a LabVIEW segítségével néhány kattintás egy olyan program megírása, amelyet C-ben órák alatt sem lehetne elkészíteni, de ellenpéldákat is ugyanilyen könnyen találhatunk: ami pl. C-ben magától értetődően egyszerű, az a LabVIEW-ban bosszantóan nehézkes lehet, tehát mindkettőnek megvan a saját optimális alkalmazási köre. A LabVIEW sem csodaszer, alapos ismeretéhez ugyanolyan fáradtságos út vezet, mint bármilyen más programozási nyelv elsajátításához. Noha néhány órai tanulás, gyakorlás után már tudunk használható programokat írni, de hónapok, évek múlva is lesz tanulnivalónk.

2. ALAPISMERETEK

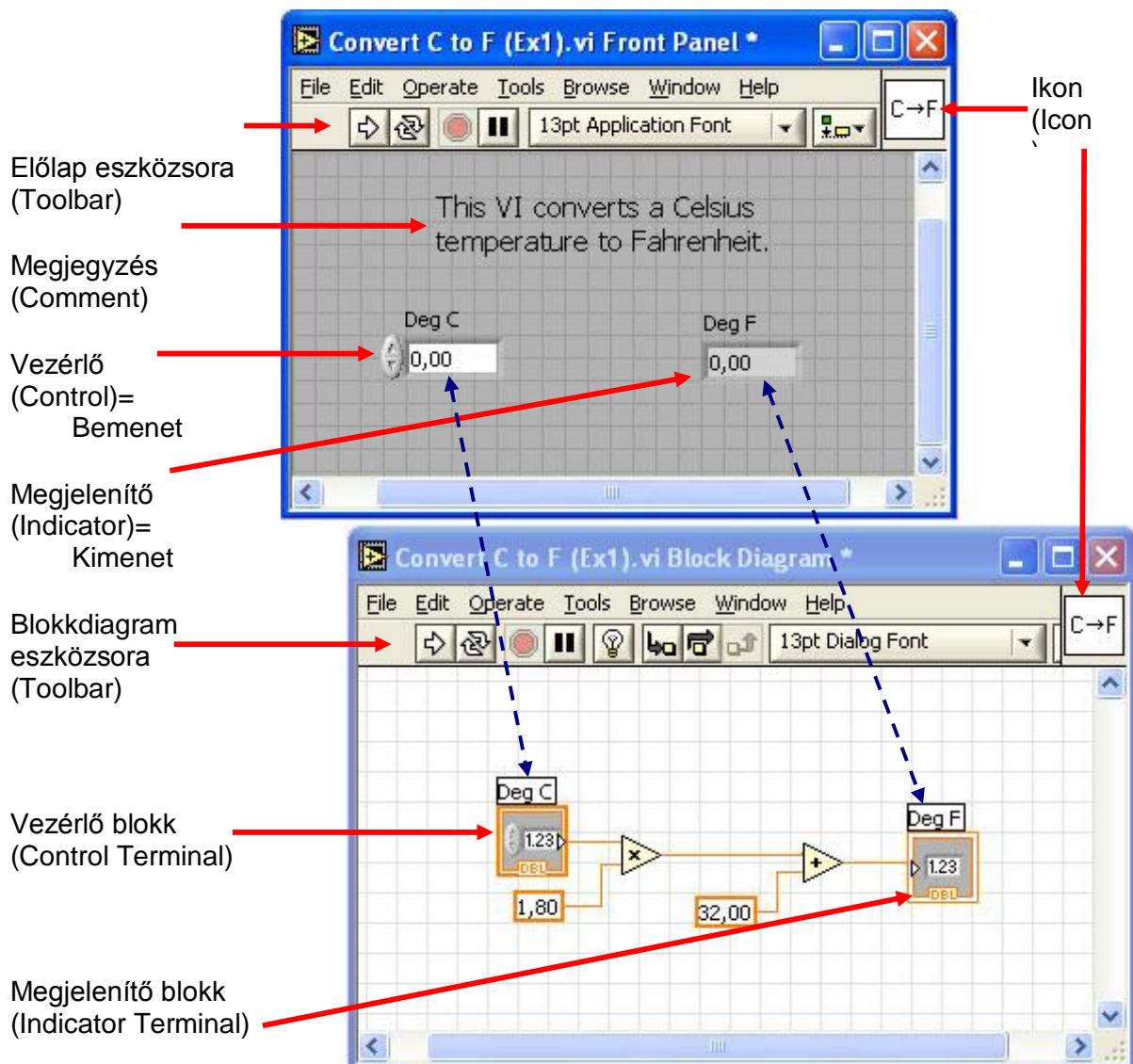
2.1 LabVIEW VI futtatása, alapfogalmak





A LabVIEW legfontosabb tulajdonságait egy egyszerű mintapéldán keresztül mutatjuk be.

Program = VI

A LabVIEW-ban a programokat „virtuális műszernek” (Virtual Instrument), vagyis VI-nek nevezik. Ha megnyitunk egy **.vi** kiterjesztésű VI fájlt a LabVIEW-val, akkor megjelenik a virtuális műszer előlapja (Front Panel). A **CTRL+E** billentyűkombináció leütésével (vagy a Window menüben Show Block Diagram), megjelenik a Block Diagram ablak is.

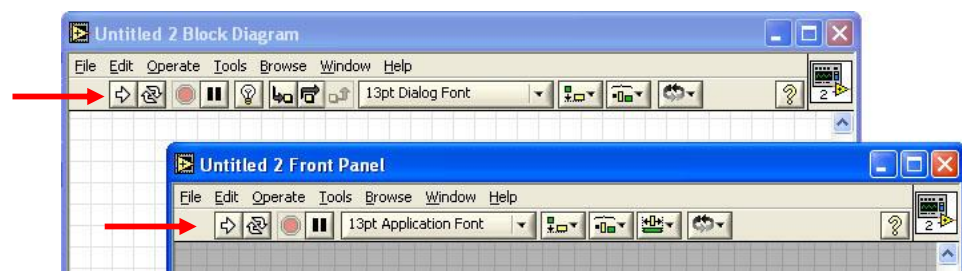
Az alábbiakban látható egy egyszerű VI Front Panel-je és Block Diagram-ja.



Előlap	Az előlapon a „virtuális műszer” (a program) ebben az esetben rendkívül egyszerű: egy Celsius fok értéket Fahrenheit fokra konvertál.
Vezérlő, megjelenítő	A minta VI-nak egy bemenete és egy kimenete van. A LabVIEW terminológiájában a bemenetet vezérlőnek (Control) nevezik, a kimenetet megjelenítőnek (Indicator). Ezen kívül vannak még konstansok (Constant), melyek értéke értelemszerűen nem változik a futás közben.
Blokkdiagram	A bemeneti és kimeneti egységek (vezérlő és megjelenítő blokkok; Control, Indicator) valamint a köztük kapcsolatot teremtő funkcionális egységek a blokkdiagramon össze vannak huzalozva. A huzalozás jelenti a végrehajtás menetét, lényegében a programot. Minden bemeneti és kimeneti egységnek kétféle megjelenési formája van: a Front Panelen látható és a Block Diagrammon látható forma. (A fenti ábrán szaggatott kék nyilakkal jeleztük az összetartozó bemeneti és kimeneti egységeket.)
Adatfolyam-programozás	Minden egyes vezérlőtől (a bemenetektől) a huzalok vagy vonalak mentén egy-egy adatút alakul ki a megjelenítőig (kimenetekig). A blokkdiagram (program) végrehajtását az adatfolyam határozza meg. Általában balról jobbra haladva szokták felrajzolni az adatfolyamokat, de fontos tudni, hogy a végrehajtást nem a felrajzolás iránya szabja meg. Egy adott funkcionális egység akkor hajtódik végre, ha az összes bemenetén rendelkezésre állnak az adatok. A funkcionális egység a végrehajtás után az összes kimenetére kiadja a megfelelő adatot.
Futtatás	 Az eszközsorban a jobbra mutató vastag, üres nyílra kattintva indíthatjuk el a VI futtatását. (Ilyenkor a nyíl ikonja átalakul feketével kitöltött szaggatott nyíllá.).
Folyamatos futtatás	  Az első ikonra kattintva a VI folyamatosan futni fog, míg az előző, egyszerű futtatás esetén csak egyszer fut le. A futás ilyenkor a második, Stop gombbal állítható le.
A futtatás animálása	 Ha a blokkdiagramon az izzólámpa ikon ki van választva, az adatok áramlását mozgó korongocskák mutatják, és az értékek megjelennek a huzalokon. A programok működésének megértéséhez és hibakereséshez ez a funkció nagyon hasznos.

2.2 Státus- és eszközsor

A Block Diagramnak és a Front Panel-nek nagyon hasonló státus- és eszközsora van (toolbar).



A Block Diagram státus- és eszközsora a következő elemekből áll (balról jobbra, az eszközsor némiképp lehet LabView verziófüggő):



Futtatás (Run) / fut / szintaktikai hiba



Folyamatos futtatás (Continuous Run)



Megszakítás (Abort Execution)



Szünet / Folytatás (Pause/Continue)



Végrehajtás szemléltetése (Execution Highlighting), debuggoláshoz



Huzalozás adatainak folyamatos elmentése a végrehajtás során, debuggolásnál használható. (Retain data values)



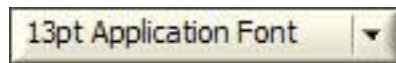
Belépés subVI-ba (Step Into)



SubVI átlépése (Step Over)



Kilépés subVI-ból (Step Out)



Szövegformázás (Text Settings)



Objektumok vonalhoz igazítása (Align Objects)



Objektumok közötti távolság beállítása (Distribute Objects)



Átrendezés (Reorder)



A kontextusfüggő sűgó megjelenítése/elrejtése (Show/Hide Context Help Window)

Csak a Front Panel státus- és eszközsorában található:



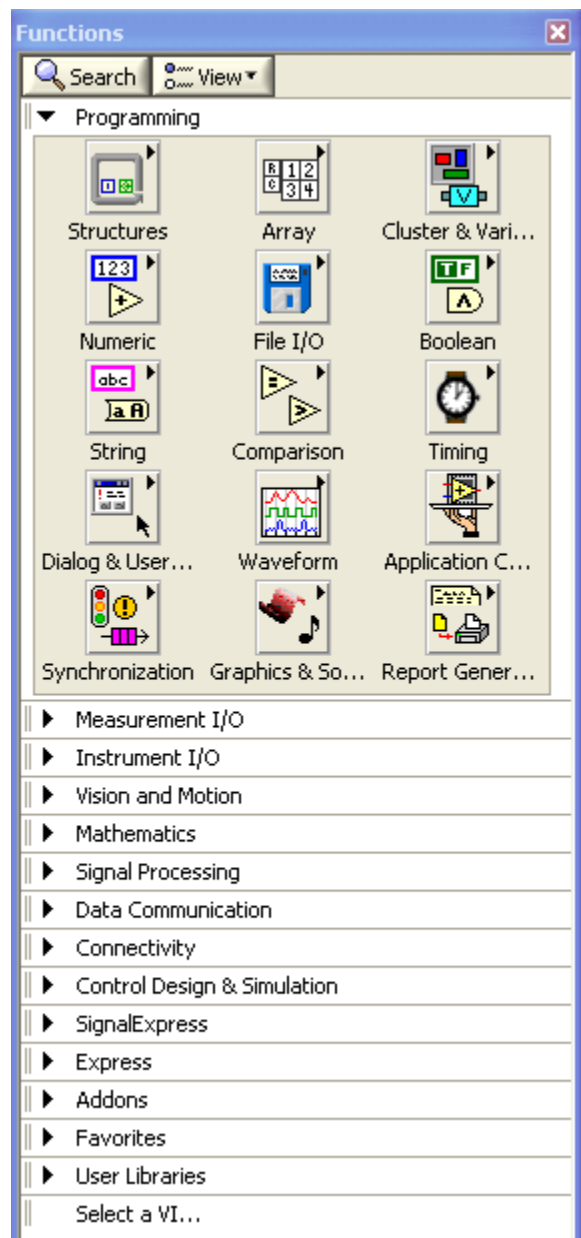
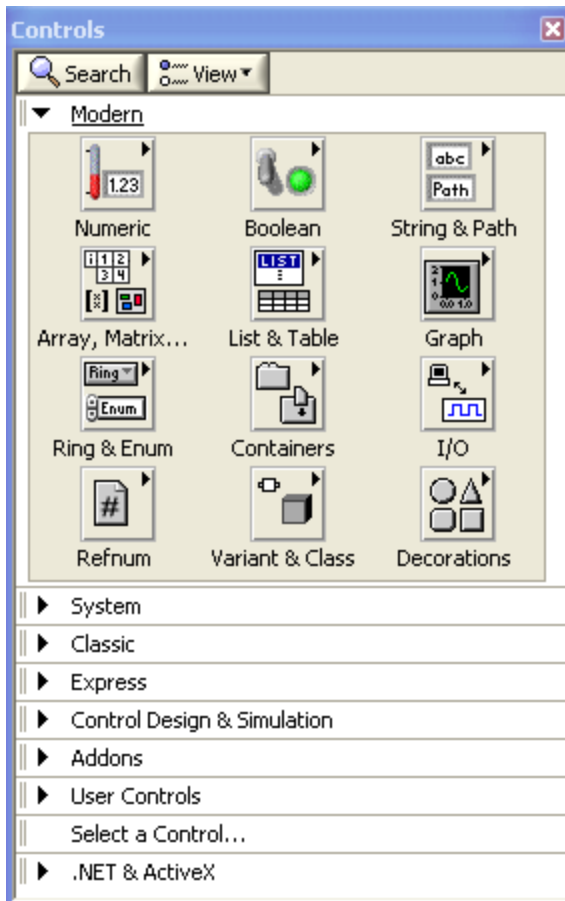
Objektum átméretezése (Resize Objects)

2.3 Egy egyszerű példaprogram létrehozása

A legalapvetőbb fogalmak megismerése után egy egyszerű példaprogram írásán keresztül mutatjuk be a LabVIEW további tulajdonságait. Létrehozzuk a Celsius–Fahrenheit átalakítás párját, a Fahrenheit– Celsius konvertert. Az algoritmus nagyon egyszerű:

A Fahrenheitben adott értékből kivonunk 32-t, majd elosztjuk 1.8-al!

- Új VI megnyitása A LabVIEW indítása után válasszuk ki a New csoportból a Blank VI (üres VI) sort.
- Megjelenik egy üres Front Panel és a szintén üres Block Diagram (ha nem jelenne meg, hívjuk elő a CTRL+E leütésével).
- Paletták A VI-k írásához, futtatásához, hibakereséséhez, javításához szükséges eszközöket ún. palettákba szervezik (a paletták kinézete verzió és konfigurációfüggő, de az alap blokkok megegyeznek).
- Vezérlők (Controls) Az Front Panel-hez tartozó segédeszközöket a **Controls** paletta tartalmazza. Ezt úgy hívhatjuk elő, ha a Front Panel-re a jobb egérgombbal kattintunk.
- Funkciók (Functions) A Block Diagram eszközei a **Functions** palettában vannak. Ezt úgy hívhatjuk elő, ha a Block Diagramra kattintunk a jobb egérgombbal.



A VI-nk előlapja

Először állítsuk elő a speciális virtuális műszerünk Front Panel-jét! Egy vezérlőnk (bemenetünk: Fahrenheit fok) és egy megjelenítőnk (kimenetünk: Celsius fok) van.

Változófogalom

Hagyományos programozás esetén két változóról beszélünk, az egyik tartalmazná a Fahrenheit, a másik pedig a Celsius értéket. Eddig szándékosan kerültük a változó elnevezést, mivel a LabVIEW-ban alapvetően nem változókkal, hanem adatforrásokkal (vezérlők, Controls) és megjelenítőkkal (Indicators), valamint adatfolyamokkal dolgozunk. (A LabVIEW-ban is vannak lokális és globális változók, de ezeket csak akkor használjuk, ha az eredeti adatfolyam-konceptióval nem tudjuk megoldani a feladatot.)

Control létrehozása

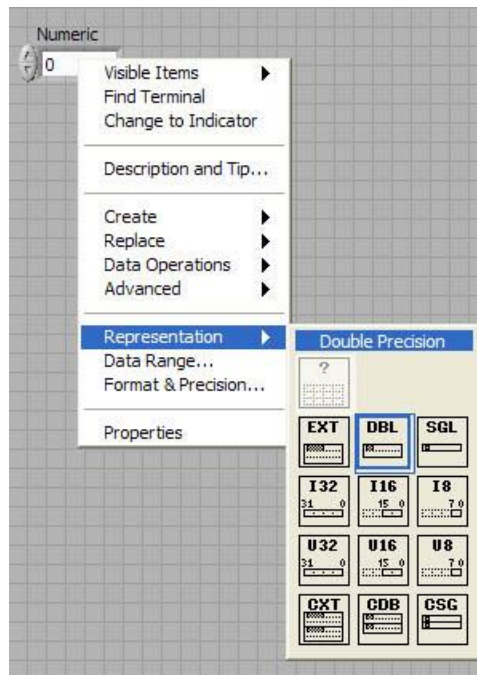
Az Front Panel Controls palettáján az egérrel kattintsunk a *Num Ctrl* alpalettára (bal felső ikon), majd válasszuk ki a Num Ctrl ikont!



Kattintsunk a blokkdiagram ablakra, és nézzük meg, hogy az előlapra elhelyezett Numeric blokkunkhoz milyen szimbólum tartozik (Numeric, DBL)!

Adattípus

Váltunk vissza a Front Panel-re, kattintsunk a jobb egérgombbal a Numeric ikonra, és válasszuk ki a legördülő menüből a Representation pontot! Itt késsel bekeretezve megjelenik a vezérlőnk (lényegében változónk) típusa (defaultként DBL, vagyis Double Precision). Ha szükséges, itt tudjuk megváltoztatni a változó típusát. (Most ne tegyük!)



Indicator létrehozása



A Numeric Control mintájára helyezzük el az előlapon az indikátorunkat (Numeric Indic...)! Ez egy Numeric 2 feliratú lesz. Ezzel megvan az adatfolyamunk eleje és vége, most a közbülső részeket kell létrehoznunk.

Nevek megváltoztatása





A Numeric és a Numeric 2 semmitmondó neveket a blokkdiagramon változtassuk meg beszédesebbekre: az előbbit írjuk át Fahrenheit-re, az utóbbit pedig Celsius-ra!

Kivonás

Az algoritmus szerint a kezdeti értékből először ki kell vonni 32-t. Ehhez kell egy kivonó egység és egy 32 értékű konstans.





A Functions paletta Numeric alpalettájából válasszuk ki a kivonó (Subtract) egységet, és helyezzük el a blokkdiagramon a bemeneti és kimeneti egység közé!

Konstansok	 <p>Ugyanerről a Numeric palettáról tegyük egy konstans értéket reprezentáló ikont a blokkdiagramra (Numeric Cons..., bal alsó sarok), és a default nulla értéket írjuk át 32-re!</p> <p><i>Ugyanezt egyszerűbben úgy tudjuk megtenni, hogyha a kivonó ikon adott bemenetére mutatunk az egérrel, majd a jobb gomb segítségével előhívott ablakból kiválasztjuk a Create/Constans menüpontot. Ez azért fontos, mert a későbbiekben így tudunk bonyolult konstansokat, de akár indikátorokat vagy controllokat is létrehozni egy mozdulattal. Ennek az az előnye, hogy ilyenkor mindig a megfelelő konstans jön létre, nem nekünk kell kiválasztani és létrehozni a típust (sok VI elég bonyolult struktúrákat vár bemenetként).</i></p>
Osztás	 <p>A Numeric alpalettáról tegyük egy osztás (Divide) műveleti egységet a blokkdiagram megfelelő helyére!</p>
Törött nyíl	  <p>Mielőtt nekilátnánk a huzalozásnak, figyeljük meg a blokkdiagram ablak státus/eszközsorában a bal oldali ikont (Run nyíl)! Egy kettétört, szürke nyilat fogunk látni. Ez jelzi, hogy a VI (program) szintaktikailag nem alkalmas futásra. Ha a program futásra kész, az ikon automatikusan átvált ép, világos nyílra.</p>
Futtatás	<p>A rendszer összehuzalozása után kattintsunk a Run nyílra! Egy rövid villanás, és megjelenik a Celsius ablakban a konvertált érték (A program egyetlen egyszer fut le).</p>
Mentés	<p>Mentsük el a programunkat Convert_F_to_C.vi néven!</p>

2.4 Vezérlési szerkezetek (Structures) és adattípusok

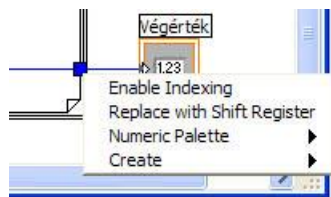
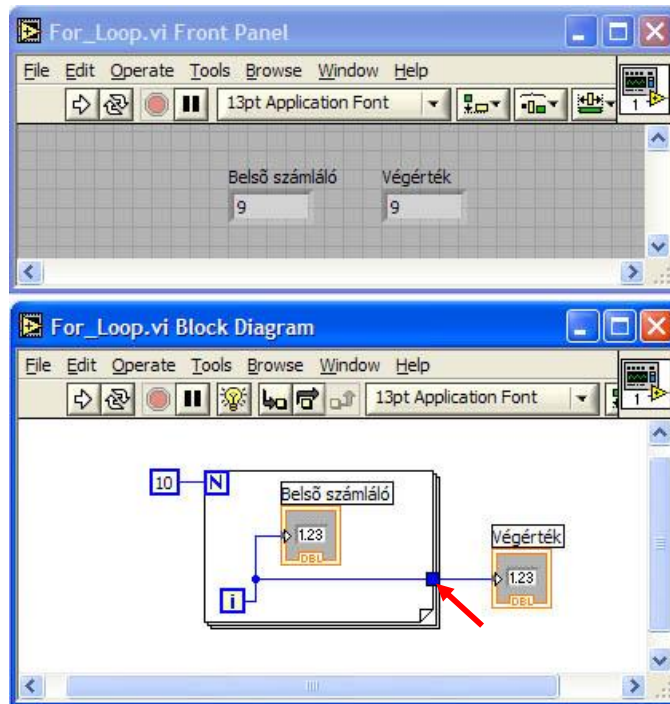
2.4.1 Ciklusok

A Functions paletta Structures palettáján található a vezérlési szerkezetek. Ezek közül először a ciklusokkal foglalkozunk.

FOR ciklus   Használatához ki kell választani a **For loop** ikont, majd körbe kell venni a ciklusba zárandó kódrészt. A For ciklusnak van ciklusszámlálója (i, Iteration Terminal), a ciklusok számát pedig az N bemeneti változó (Count Terminal) értéke határozza meg. A ciklus 0-tól N-1-ig fut.

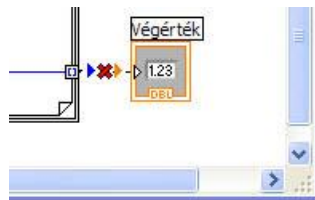
Tekintsük az alábbi mintaprogramot (FOR_Loop.vi):

A piros nyíllal jelölt kék négyzetnek az a feladata, hogy „kivezesse” az i ciklusváltozó végértéket a ciklusból. Ha a jobb egérgombbal rákattintunk a kék négyzetre, előugrik egy legördülő menü: Enable Indexing stb. menüpontokkal. Az Enable Indexing arra is utal, hogy most Disable állapotban van a kék négyzet (le van tiltva az automatikus indexelés, lásd alább), vagyis ebből az állapotból Enable-be tudjuk tenni. Auto index letiltva

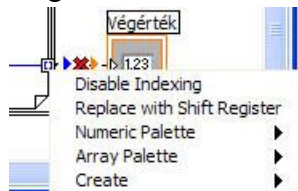


Auto index engedélyezve

Válasszuk ki az Enable Indexing sort! A következőt tapasztaljuk:



Megváltozik a négyzet kitöltése, és a Végérték-hez menő adatfolyam megszakad. Kattintsunk ismét a négyzetre!



Az Enable/Disable Indexing most már elárulja, hogy mi történik. A ciklus kétféleképpen vezethetjük ki az *i*-t: automatikus indexelés letiltásával vagy automatikus indexeléssel. Az előbbi esetben a kimenő adat egyetlen érték, az *i* végértéke lesz, míg az utóbbiban egy tömb, a ciklusváltozó ciklusonkénti értékével. A fenti példában a kapcsolat

azért szakadt meg, mert egy tömböt nem lehet közvetlenül összekötni egy skalárral.

Ez az automatikus indexelés (vagy annak letiltása) általánosan is igaz, nemcsak a FOR ciklusra, és nemcsak a ciklusváltozóra működik.

Adattípusok

A FOR_Loop.vi az adattípusok szemléltetésére is alkalmas. A LabVIEW-ban jelentősége van a színeknek. A kék szín egész (integer) típust jelent (lásd i és N, valamint a 10 konstans), a barna dupla pontosságú lebegőpontost (double). Itt automatikus típuskonverzióra is látunk példát: az i-t probléma nélkül összekapcsolhatjuk a DBL változóval, a LabVIEW automatikusan elvégzi a szükséges konverziót. Természetesen, az a jobb megoldás, ha azonos típusokat kötünk össze, és mi végezzük el a típusdefiníciókat (Representation: DBL, I32, I16 stb.).

Tömbök (arrays)

A normál programozási nyelvekhez hasonlóan a LabVIEW-ban is lehetőségünk van tömböket létrehozni és kezelni.

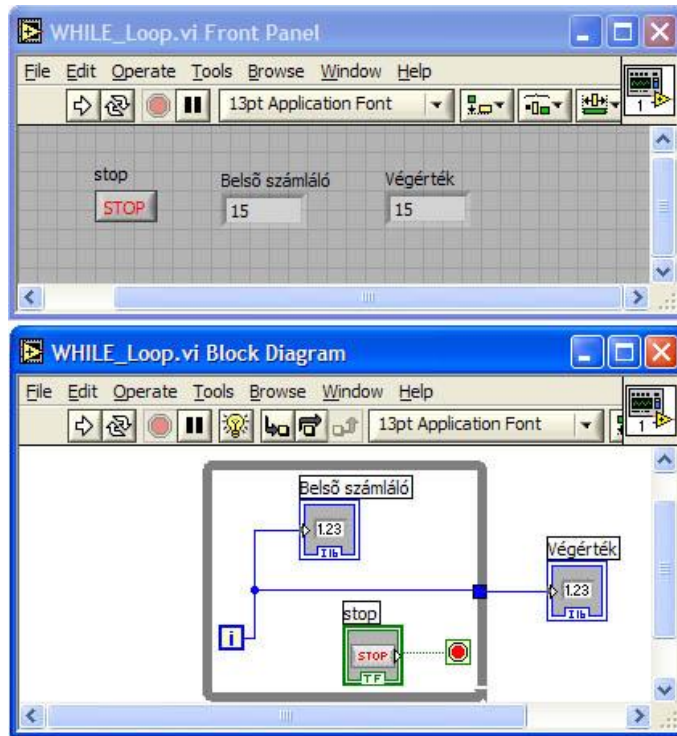
Fürtök (Clusters)

A fürt vagy köteg (cluster) adatokat összefogó adatstruktúra. Az adatok különböző típusúak is lehetnek. Hasonlít a C nyelv struct-jához. A gyakorlatban legtöbbször bonyolultabb VI-k konfiguráló konstansaiként fogjuk használni.

WHILE ciklus

Van ciklusszámlálója (i, Iteration Terminal). Legalább egyszer lefut. A ciklusok számát a ciklusfeltétel (Conditional Terminal) szabja meg. Az alábbi WHILE_Loop.vi mintapéldában a ciklus mindaddig végrehajtódik, amíg rá nem kattintunk a STOP gombra az előlapon. (Természetesen, minden program leállítható a státuszsorban levő Stop gombbal.)

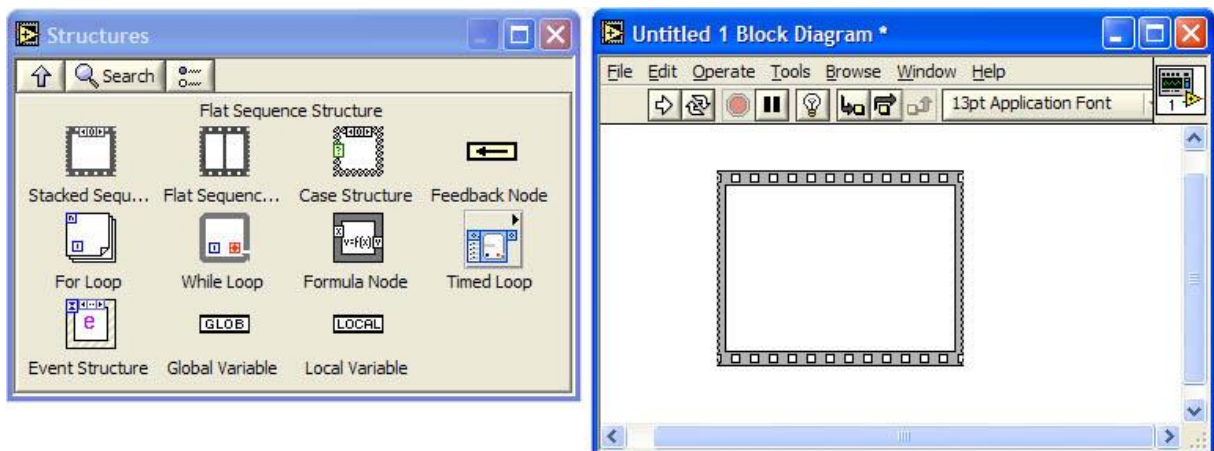
A while ciklus leállítására sokféle lehetőség van. A jobb egérgombbal a piros stop jelre kattintva a legördülő menüből kiválaszthatjuk a számunkra legmegfelelőbbet.



2.4.2 Sorrendi struktúrák

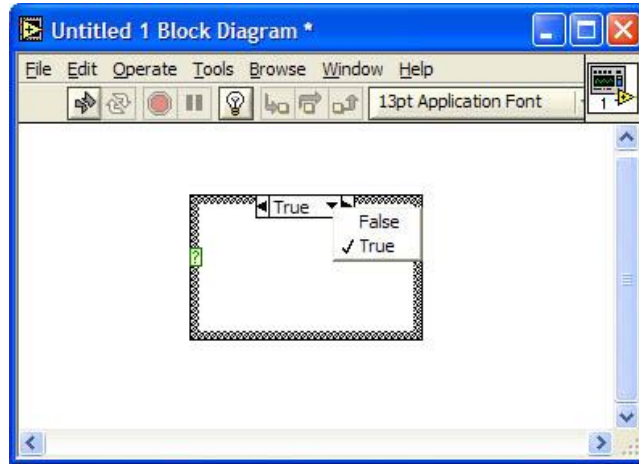
A sorrendi struktúrák alapesete a **Flat Sequence**. A Functions paletta Structures alpalettáján található, kinézete egy filmszalaghoz hasonló. Ikonjának kiválasztása után a blokkdiagrammon körbe kell venni a szekvenciálisan végrehajtandó programrészt, vagy a keret kijelölése után lehet belevonszolni a blokkokat!

A blokkokat szekvenciálisan, mint a filmszalag képkockáit, egymás után hajtja végre.



2.4.3 Választás (Case) struktúra

A szokásos switch–case struktúra stack-szerű megjelenítése. A keretek egymás mögé illeszkednek, akár a kártyalapok. Egyszerre csak egy „lap” látszik. Helye: **Functions / Structures / Case Structure**.

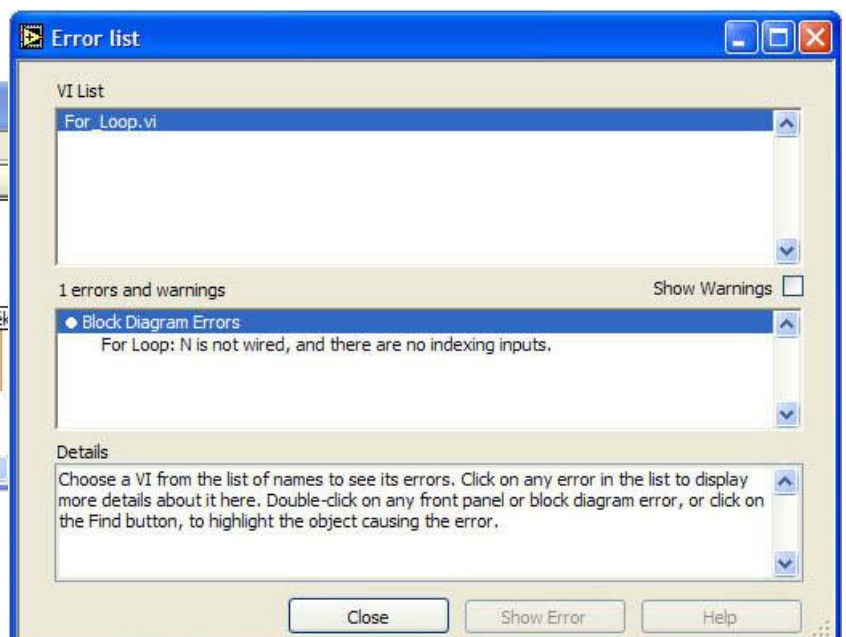
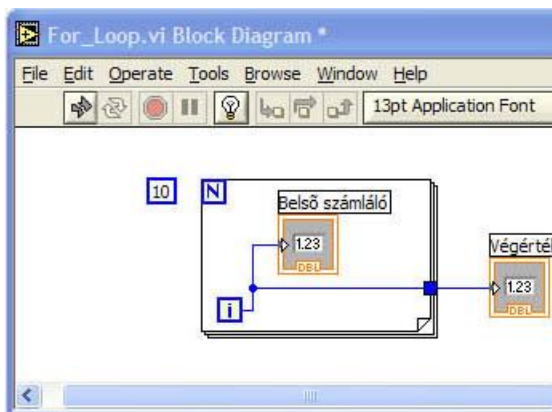


A keretre történő jobb-kattintással előjön a struktúrához tartozó legördülő menü. A legördülő menüben adhatunk új case eseteket a struktúrához, illetve távolíthatunk el nem használt eseteket. Fontos megjegyezni, hogy a Case struktúra által mutatott esetek (Case-ek) függenek a kiválasztó bemenet típusától. Alap esetben a Case struktúra logikai kiválasztó bemenettel rendelkezik, így két eset van definiálva a True és a False. Amennyiben a kiválasztó bemenet típusát megváltoztatjuk (pl. integer vagy string típusra), akkor az esetek is megváltozhatnak.

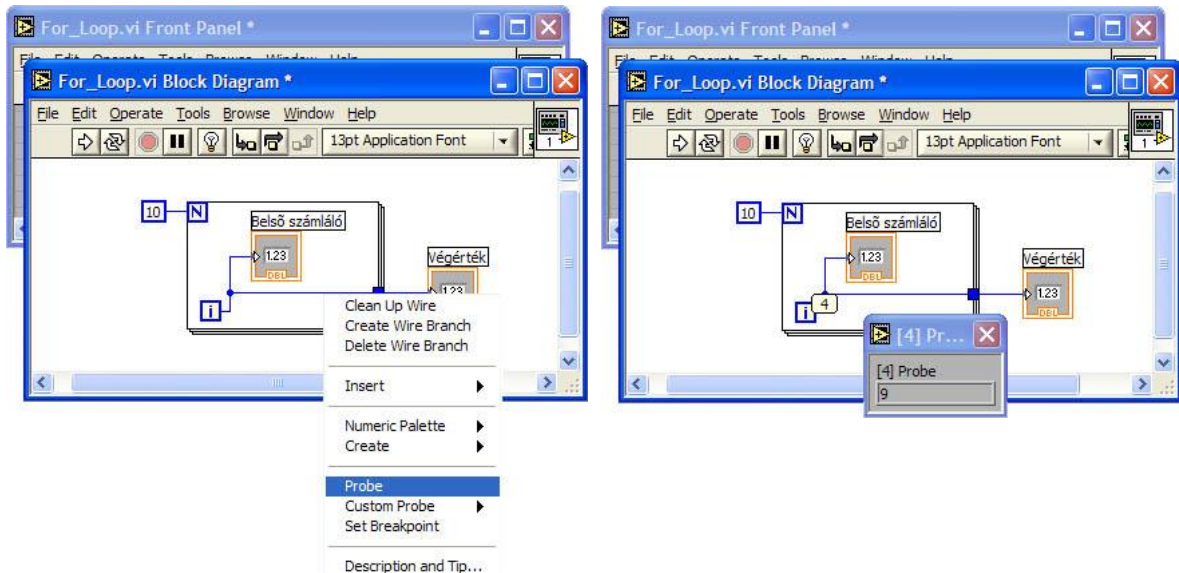
2.5 Egyéb fontos LabVIEW elemek


Hibakeresés

Kattintsunk a törött nyíl gombra! A megjelenő ablak kiírja a hibá(ka)t.



Mintavevő (Probe) Jobbkattintás a huzalra: megjelenik egy legördülő menü, kiválasztjuk a Probe-ot, amely futás közben kijelzi a huzalon „átáramló” adatot. (Kiválaszthatjuk a Mintavevő (Probe) eszközt is az Eszközök (Tools) palettából, majd kattintsunk a huzalra!)



Kontextusfüggő sűgő  Nagyon sokat segít a kontextusfüggő sűgő (**Context Help**). Ha rávisszük a kurzort valamelyik elemre, a hozzá tartozó rövid magyarázat azonnal megjelenik.

subVI A LabVIEW-ban a VI-k nem csak főprogramként használhatóak, hanem lehetőségünk van egy VI-ban egy másik VI-t, mint egyetlen blokkot futtatni. Tehát minden VI-ból készíthetünk egy olyan egységet, amelyet egy másik VI-ba szubrutinszerűen, egyetlen blokkként beilleszthetünk. Ezeket nevezzük subVI-nak. A subVI-k esetében a bemenetek és kimenetek helyettesítik a Front panel kezelőfelületét. A subVI-k létrehozását egyszerűen kipróbálhatjuk úgy, hogy a blokk diagramban a subVI-ba zárni szándékozott területet kijelöljük, majd az Edit / Create Sub VI menüponttal létrehozunk belőle egy subVI-t. A subVI-kből könyvtárakat állíthatunk össze. A subVI-k használata egyszerűsíti és átláthatóbbá teszi a programjainkat, gyakorlatilag a LabVIEW könyvtári elemek többsége is mint subVI jelenik meg a számunkra.

3. ISMERKEDÉS A VIRTUÁLIS MŰSZEREKEL

3.1 A National Instruments műszerkínálatának áttekintése

Adatgyűjtő kártyák A legegyszerűbb módja, hogy egy PC-ből műszert készítsünk az, hogy valamilyen adatgyűjtő interfésszel egészítjük ki. Ez az adatgyűjtő interfész lehet a legegyszerűbb, minden gépen megtalálható hangkártya, vagy annál lényegesen több funkcionalitást nyújtó PCI, PCI Express buszra csatlakozó adatgyűjtő modul. A National Instruments igen széles eszközválasztékkal rendelkezik ilyen modulokból. Az általános célú adatgyűjtő eszközök 4-80 analóg bemenettel, 0-4 analóg kimenettel, 16 bites felbontással és 250 kHz-től 1,25 MHz-es mintavételi frekvenciával rendelkeznek, de kaphatóak speciális modulok is, amelyek vagy a mintavételi frekvenciában, vagy más paraméterben lényegesen felülmúlják ezeket az általános eszközöket.



Az analóg adatgyűjtő kártyákon kívül más, speciális funkciókat ellátó modulok is találhatóak a National Instruments termékpalettájában (motor meghajtók, digitális I/O-k stb).

Műszerek illesztése Amennyiben valamilyen oknál fogva a PC-s adatgyűjtő kártyák nem lennének megfelelőek, vagy elérhetőek akkor általában valamilyen külső műszert kell a PC-hez csatolni. A hozzácsolás módját mindig az illesztendő eszköz határozza meg azzal, hogy milyen kommunikációs interfészt nyújt. Ez a legtöbb esetben RS232, RS485, CAN vagy Ethernet-et jelent, de sok műszer egy hagyományos műszerillesztő felületet a GPIB támogatja.

A GPIB busz Mivel a GPIB-ről az eddigi tanulmányok során nem igen esett szó, itt röviden bemutatjuk, bár a laborban nem fogjuk használni. A GPIB kommunikációs felületet az IEEE 488-as szabvány specifikálja és több, mint 30 éve használják az iparban. Sok műszer még manapság is támogatja ezt a felületet, például az alaplabor mérések során használt Agilent műszerek mindegyike rendelkezik GPIB interfésszel (Ez már azért sem meglepő, mert a szabvány alapját 1965-ben a HP fejlesztette ki HP-IB néven). A GPIB busz sebessége 1 MB/sec ezt 8 adatvonalra és számtalan kiegészítő handshake vonal segítségével tudja elérni. A National Instruments egy kiegészítést készített ehhez a szabványhoz HS-488 néven, ami 8 MB/s sebességre képes.

PXI

Sok ipari környezetben nem megengedett PC-k kihelyezése, ezért a szükség lehet egy kompaktabb ipari kivitelű rendszerre. A National Instruments ezekre a szituációkra fejlesztette ki a PXI rendszerét. A PXI (PCI eXtension for Instrumentation) gyakorlatilag egy az ipari követelményeknek megfelelő PC bázisú rendszer. A PCI busz ipari kiegészítése pedig a szinkronizációs problémák megoldását szolgálja. A kiegészítő jelek között szerepel közös órajel és triggerjelek. Szoftver tekintetében nincs lényegi eltérés a PXI rendszerek és a normál PC-s rendszerek között, mivel mindkettő PC-s alapon működik.



A PXI rendszerek előnye a szinkronizációban és az ipari kivitelben keresendő. Mint az ábrán is látható ugyanúgy modulárisan, adatgyűjtő kártyák beillesztésével lehet a PXI eszközökből mérőrendszert megvalósítani, mint az asztali PC-s esetben, csak itt a kivitel kompaktabb és jobban megfelel az ipari követelményeknek. A PXI jelenleg talán a legszélesebb körben használt ipari adatgyűjtő, vezérlő platform.

CompactDAQ

A National Instruments palettáján az USB egyre szélesebb körű elterjedésének köszönhetően a 2000-es évek közepén jelent meg a CompactDAQ sorozat, amely a PXI-os moduláris műszerépítési koncepciónak egy olcsó PC központú verziója.



Az USB-re csatlakozó alapsínhez itt is többfajta mérő beavatkozó modul illeszthető, így viszonylag olcsón (a PXI-hoz képest) létrehozva egy moduláris mérő, beavatkozó rendszert.

CompactRIO

Az USBs sorozat sikerét követően készült el az egy fokkal komplexebb feladatokra képes, a PXI-al majdnem megegyező funkcionalitást nyújtani tudó CompactRIO (RIO: Reconfigurable I/O) sorozat, ahol a

mérő modulokat befoglaló sín Real-Time PC-t és FPGA támogatást is tartalmazhat.



A CompactRIO sorozat tulajdonságait és alkalmazását későbbiekben részletesen bemutatjuk.

LabVIEW támogatás A fent bemutatott hardver megoldások csak úgy váltak jól használhatóvá, hogy a National Instruments igen magas szintű támogatást nyújt hozzájuk a LabVIEW illetve LabWindows/CVI környezetben. Ez a támogatás gyakorlatilag azt jelenti, hogy a cég által forgalmazott hardverek VI szinten elérhetőek, konfigurálhatóak és nagyon egyszerűen alkalmazhatóak, például *Data Acquisition Assistant (DAQ Assistant)* szoftvercsomag segítségével.

Alkalmazási terület A bemutatott rendszerek alkalmazási területe elsősorban ott keresendő, ahol a mérési funkciók változatosak, és egy standard műszer nem tudja kielégíteni azokat. Illetve még inkább azokon a területeken, ahol valamilyen együttműködés szükséges a különböző műszerek, érzékelők beavatkozók között. Tipikus alkalmazási terület valamilyen ipari automatizálási folyamat, például gyártósorok automatizálása, ahol minden lépésnél tesztelő méréseket kell végezni, és ezeket a méréseket folyamatosan dokumentálni kell az ISO-9000-es szabvány követelményei miatt. Ezekre a problématerületekre nem igen létezik más ennyire komplett eszközkészletű megoldás, így nagy valószínűséggel állíthatjuk, hogyha felkeresünk egy gyárat, akkor ott találkozni fogunk ezekkel a technológiákkal és eszközökkel.

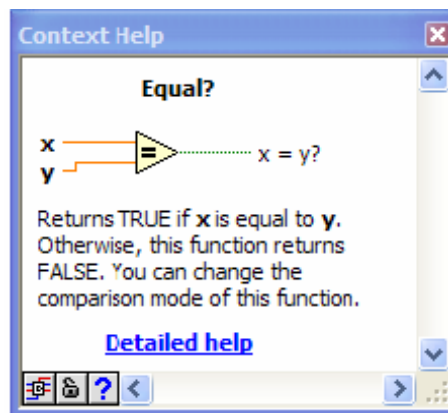
4. LV0 MÉRÉSI FELADATOK

4.1 Fahrenheitről Celsiusra konvertáló VI készítése

Hozzon létre egy üres VI-t (File menü / New / Blank VI), majd készítse el a Fahrenheit–Celsius konvertáló programot! (Emlékeztetőül: a Fahrenheitben adott értékből vonjunk ki 32-t, majd osszuk el 1,8-cal hogy megkapjuk a Celsius értéket.)

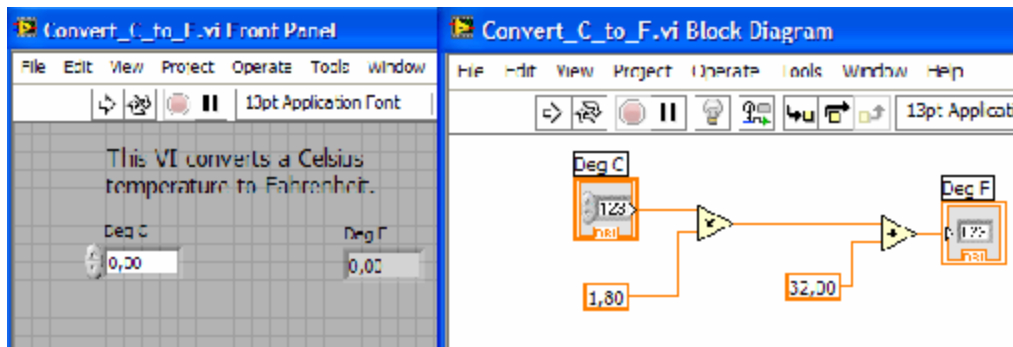
Segítség a programozáshoz

- Kontextusfüggő sűgő (Context Help: **Ctrl+h** billentyűkombinációval érhető el) információkat ír ki az egér pozíciójában lévő LabVIEW grafikus elemről. A sűgő választ ad a „Mit is csinál ez az elem?” kérdésre. Egyenlőségvizsgálatnál pl. így néz ki:




Kontextusfüggő sűgő

- Bal kattintással kijelölhetjük az adott elemet, amely tetszőlegesen mozgatható (*drag and drop*). A különböző mezőket dupla kattintással szerkeszthetjük.
- A VI-k szerkesztésekor alapértelmezésben az egér mutatója folyamatosan változik attól függően, hogy mire mutat. Például, ha egy elem kimenetére mutatunk, akkor automatikusan átváltozik a vezetékezési lehetőséget mutató „spulni”-ra. Ekkor az egérgomb lenyomásával lehet **huzalozni**, azaz összekötni a VI-k be- és kimeneteit.
- A LabVIEW-nak két felülete van: *Front Panel*, ez a felhasználói felület, illetve a *Block Diagram* panel, ahol a grafikus program készíthető el. Gyors váltás az ablakok között: **Ctrl+e**.
Ha az egér a *Block Diagram*-ra mutat, akkor a *Functions* paletta jön elő a jobb-klikk hatására, és a LabVIEW kód grafikus elemeit helyezhetjük el a panelon (lényegében így írjuk a programot). Ha a *Front Panel*-re kattintunk jobb-egérgombbal, akkor pedig a *Controls* paletta elemeit, jellemzően beviteli mezőt és kimeneti értéket megjelenítő mezőket helyezhetünk el.



LabVIEW Front Panel (balra) és Block Diagram (jobbra)

- Ha a *Block Diagram*-on lehelyezett ikonra vagy annak egy be- vagy kimenetére mutatóval kattintunk a jobb-egérgombbal, akkor a feljövő menüben nagyon hasznos a *Create* menüpont, erre kattintva ugyanis közvetlenül be lehet illeszteni *Constant* (csak a *Block Diagram*-on jelenik meg, felhasználó számára nem elérhető), *Control* (*Front Panel*-en is megjelenő beviteli mező) vagy *Indicator* (*Front Panel*-en is megjelenő kijelzés mező) elemeket. Ez sokkal egyszerűbb, mint ugyanezeket kikeresni a *Functions* palettáról.
- Ha egy vezeték (*wire*) nem szerencsésen van behuzalozva a blokkdiagrammon, vagy nem egyértelmű, hogy hová van bekötve, akkor hasznos, ha a jobb-egérgombbal rákattintunk, és kiválasztjuk a *Clean Up wire* funkciót. Ez automatikusan átrajzolja a vezetéket egy tetszetősebb, egyértelműbb kinézetre. Az esetek 90%-ban jól működik.
- Lehetőség van a teljes grafikus kód megjelenítésének optimalizálására a *Block Diagram* felület felső menüsorában található *Clean Up Diagram*  gombra kattintva.
- Nem LabVIEW-specifikus, de hasznos: Alt+Print Screen csak az aktív ablakot teszi a vágólapra, nem pedig az egész képernyőt.
- A blokkdiagram státuszsorában kapcsolja be az izzólámpát (*Highlight Execution*), amely animált módon megjeleníti az egyes vezetékeken „lévő” adatokat, értékeket!

4.2 Ciklusok vizsgálata

Módosítsa az előző VI-t, vegye körül egy *for* ciklussal a programot, és írassa ki, hogy hányadik ciklusban vagyunk, majd gyűjtse tömbbe az iterációk sorszámát! Cserélje le a *for* ciklust egy *while* ciklusra, és ellenőrizze a program helyes működését!

4.3 Virtuális függvénygenerátor

Készítsen függvénygenerátort, amely PC képernyőjén megjeleníti a kiválasztott hullámformákat!

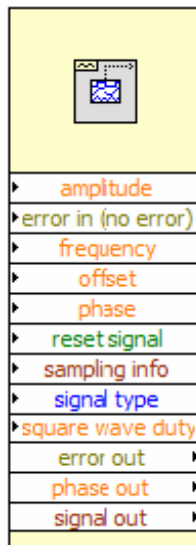
A függvénygenerátor fő funkciói a következők:

- Szinuszjel, négyszögjel, háromszögjel közül legyen kiválasztható a hullámforma!
- A négyszögjelnél legyen megadható a kitöltési tényező!
- legyen állítható a frekvencia és az amplitúdó!
- A kiadott hullámformát jelenítse meg a képernyőn!

- Az előlapon igényesen rendezze el a kezelőszerveket!

Segítség a feladat megoldásához: használja a **Functions** paletta (*Block Diagram* ablakon jobb-klikk) **Signal Processing** menüjéből a **Wfm Generation** almenüben található **Basic FuncGen** VI-t! Ezt már csak a megfelelő kezelőszervekkel kell ellátnia. (Használja a Ctrl+h billentyűkombinációt a kontextusfüggő sűgő eléréséhez!) Ezen kívül érdemes a hullámforma-generátorhoz tartozó elemeket egy *while* ciklusba helyezni, hogy ne csak egyszer fusson le a jelgenerálás.

Lehetőség van a grafikus elem be- és kimeneteinek átláthatóbb megjelenítésére a következő módon: jobb-klikk az elemre, majd a **View as icon** tulajdonság kiválasztását meg kell szüntetni. Ekkor a 3. ábrának megfelelően fog kinézni az ikon:



A Basic FuncGen függvény „kibontott” grafikus megjelenése

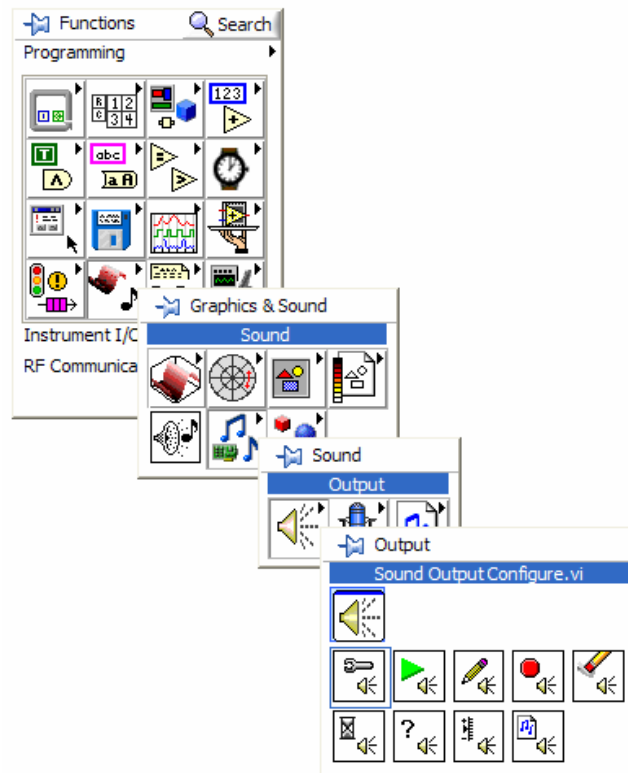
Megjegyzés: Nagyon fontos ismerni azt, hogy mi a kapcsolat a jelfrekvencia (*f-frequency*), a mintavételi frekvencia (*Fs-sampling frequency*), és a minták száma (*#s – number of samples*) között. Például egy $f=10$ Hz-es jelfrekvenciájú szinuszos jelet mintavételeztek $F_s=100$ Hz-cel, és $\#s=10$ mintát rajzolok ki. Akkor 1 teljes periódust fogok látni, mivel $F_s/f=10$, vagyis 1 periódus 10 mintát tartalmaz. Gondolja át az előbbieket!

4.4 Függvénygenerátor megvalósítása a PC hangkártyájával

Készítsen függvénygenerátort, amely PC képernyőjén megjeleníti a kiválasztott hullámformákat az alábbi módon:

- Bővítse ki az előző feladatban elkészített függvénygenerátort, úgy hogy a jeleket kiadja a PC hangkártyája segítségével!
- A kiadott hullámformát jelenítse meg a képernyőn is!
- Csatlakoztasson egy Jack – RCA kábelt a hangkártya kimenetéhez, majd az RCA – BNC átalakító segítségével csatlakoztassa a kábelt az oszcilloszkóphoz! Ellenőrizze a függvénygenerátor működését az oszcilloszkópon!

Segítség a feladat megoldásához: használja a **Functions** paletta **Graphics & Sound / Sound / Output / Configure**, **Write** és **Clear** elemeket. A grafikus elemek elérésében segítséget jelent a 4. ábra. Figyeljen arra, hogy a **Configure** és **Clear** elemeket a **while** cikluson kívülre kell helyezni, hiszen azokat csak egyszer kell futtatni. A **Write** elemet a cikluson belülré kell helyezni, hiszen folyamatosan küldjük ki a generált hullámformát (annak egy tárolt mintafüggvényét ismételjük). Ügyeljen arra, hogy a **Configure** elem sound format bemenetén ugyanaz a mintavételi frekvencia legyen beállítva, mint a **Basic Function Generator sampling info** bemenetén! Gondolja át, hogy miért!



A Functions paletta függvényei

4.5 Opcionális feladat: reakcióidő-mérő

Egy gomb lenyomásától számított 1-5 másodpercen belül egy LED véletlenszerűen kigyullad, mire a felhasználónak ismét le kell nyomnia a gombot. A LED felvillanása és a gomb másodszeri megnyomása közötti időt mérjük ms-ban.

A feladatot állapotgépes módszerrel oldja meg *case* struktúrával.