Készítette: Nagy István buenos@freemail.hu BME

NATIONAL INSTRUMENTS™ LabVIEW™ 6.1 -Használati segédlet

A Labview-val virtuális m szereket (VI) lehet készíteni, melyekkel számítóógépr l lehet a géphez csatlakoztatott (RS-232, USB, Bluetooth, GPIB buszokon keresztül) intelligens m szereket, vagy egyéb processzoros készülékeket, áramköröket tesztelni, vezérelni, tesztelést, mérést automatizálni.

A vizuális programozás elve: A VI-k vizuálisan leírt programok, melyekkel blokkvázlat-szer en írjukj le a programunk m ködését, szöveges kódolás helyett. A blokkvázlatban hagyományos programozási elemek (pl. ciklusok…) és m szerkezel panelek (DRIVERek) vannak. A m szerkezel panelek is VI-k, valakik megírták alapelemekb l. Be lehet illeszteni készen beszerzett VI-ket is, vagy saját, vagy a Labview-hoz adják, vagy a m szerek gyártói mellékelik. (pl. www.ni.com) A VI-k egymásba ágyazhatóak.

A VI-k szerkesztése:

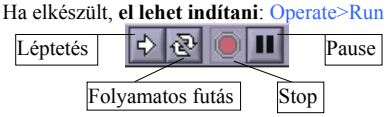
Ez 2 szinten folyik: Blokkdiagram-nézet, és a kezel felület-nézet. Minden VI-nek, és al-VI-nek. .

A kezel felület-nézet: szürke hátter ablak (hacsak valaki át nem állítja). Ezen keresztül kezeljük a VI-nket. Ez a kész project arca. Kezel f. És kijelzés.

A Blokkdiagram-nézet: Fehér hátter ablak. A fejlécbe van írva, hogy diagram.(A másik ablakba nincs). Ebben a nézetben adjuk meg a VI bels m ködését

VI-készítés lépései:

1) Meg kell bizonyosodni, hogy a megfelel kommunikációs **driverek** telepítve vannak. (soros/párh-port default, többi nem)
2) A LabVIEW-ot indítva lehet választani: **new** VI/open VI. Megjelenik mindkét nézet.
3) Le lehet **pakolni**, ami teccik. Jobb gombbal kattintva, mindkét ablakban megjelenik, az oda tartozó menü: blokkdiagram-nézet: Functions, a kezel felület nézeten: Controls. Ezekb l lehet dobozokat lepakolni. Van még egy menü: Tools. Ezek a menük kis ikonokból állnak, és minden elemük almenüket nyitnak meg, amik funkció-csoportok. Ha lerakunk egy elemet az egyik nézetben, az a másikban is megjelenik, az ott megfelel formában.
4) **Összekötöz**getjük az elemeket. Mozgatáshoz a tools menüben a kis nyíl legyen kijelölve! Összekötözéshez a cérna ikon legyen kijelölve! A Tools menü mindkét nézetre használható.
5) **Mentés**: Bármelyik nézetb l: File>save as
6) Ha elkészült, **el lehet indítani**: Operate>Run

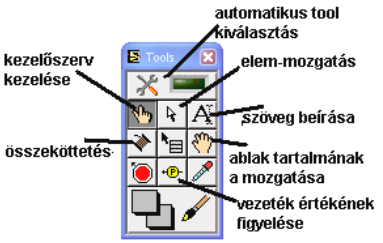


Ekkor már üzemkész a VI, ez a normál használat is. Ez a kezel -nézetbeli menü. A másik nézetben van még pár gomb a teszteléshez:

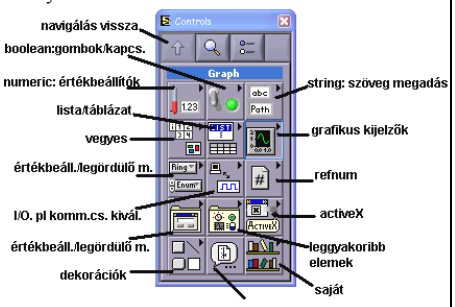
7) **Tesztelés**: a  -ikonnal láthatóvá lehet tenni a m ködési folyamatokat. A  -ikonnal pedíg lassú lépegetésre lehet bírni a programot. Ezek a blokkdiagram nézetben vannak.
8) A már kész VI-nkre, ha rákattintunk, (*.vi) megnyílik, de csak a kontroll panel. A blokkdiag-ram nézet nem, hogy használat közben ne zavarjon. Window>Show Diagram

Menük:

1) Tools: (mindkét nézetre, szerkesztési mód kiv.)

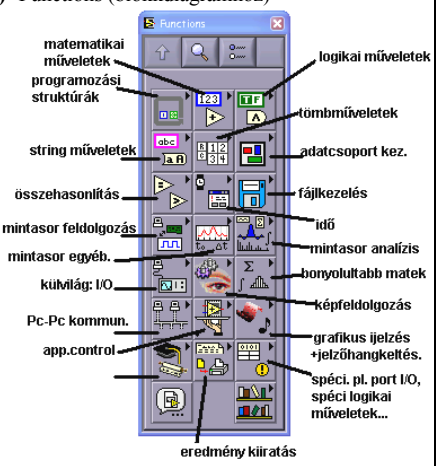


2) Controls: (kezel felület nézetre.) Almenükre nyílik minden ikon.



A saját: pl letöltött, vagy el re elkészített VI-k.

3) Functions (blokkdiagramhoz)



4) Egyéb beállítások:
File>VI Properties : pl. ablakméret…beállítása. A Window menüben állítható, hogy melyik ablakok, menük láccódjanak, láccódjon-e a bd.

Context Help: Help>ShowContextHelp: amikor egy dobozt be akarunk kötni, rámutatunk a dobozra, és a help kijelzi, hogy melyik terminálja melyik.

Beépíthet elemek:

-Összeköttetések: (változók): Csak az azonos adattípusú elemeket lehet összekötni. Összeköttetéseket az elemek TERMINÁL-jain keresztül létesítünk. A terminál megjelenik, ha rámutatunk a kurzorral az adott dobozra: Rámutatás el tt után.

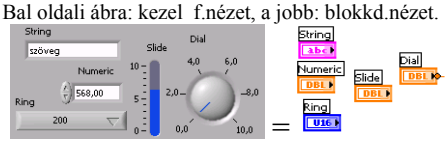
Adattípusok: logikai ( ), tömb (array int: , double: , integer( ), float, double ( ), string ( ), cluster (lista vagy), waveform (tömb szer). Ezek az összeköttetéseken terjednek. Más vonaltípussal is jelzi ezeket.

Az egész VI-t egy jelfolyamba foglaljuk. **Olyan, mint egy digitális kapcsolási rajz.** De a vezetékeken bármilyen c-ben megszokott adattípus lehet, akár tömb is. Lehetnek benne sorrendi elemek is. Az els elemmel indul, pl start-gomb, a vége pedíg CLOSE-gomb. Lehet benne végtelen ciklus, mint a c programokban. Pl.:start gombot feltételnek kötjük be egy ciklusba, így az addíg nem indul, amíg a gomb=0. Stop gombra pedíg kilép. A ciklus vagy egyéb is akkor ad a kimenetén 1-est, ha akar, addíg meg a következ nem indul.

Vannak állandó m ködés elemek (pl. led), és alkalmi m ködés ek, amik akkor hajtanak végre valamit, ha kapnak egy indító impulzust a bemenetükön (pl. ciklusok).

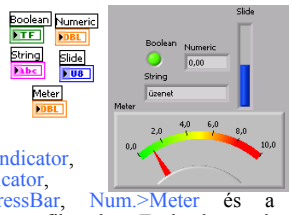
-Konstansok: A Numeric>Num.Constant, vagy a Boolean>TrueConstant, vagy false. A blokkdiagram nézetben. Ennek csak kimenete van.  A logikai konst. kattintással átbillenthet 1-b l 0-ba. A numerikusnál: Jobbcl.>representation :szám típus.

-Paraméter megadása: String&Path>bármelyik: szöveg, amit a VI felhasznál, pl kiküldi porton. Számparaméter: Numeric>bármelyik. Ezek között lehet léptethet , vagy potméter, vagy csuszka alakú. Ring&Enum>MenuRing: Ez a legördül menü. Ehhez értékek megadása: Jobb_cl.>Add_Item_After, de csak a kezel -nézetben. A kimeneti vezetéken a kiválasztott (el lapon) értéket fogja adni. Minden elemnek nevet lehet adni, de default nevet is kapnak. Bal oldali ábra: kezel f.nézet, a jobb: blokkd.nézet.



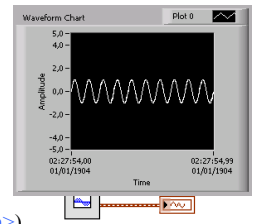
-Kezel szervek Bármely nézetben fel lehet tenni. Akár, a Boolean, akár a ClassicControls>Boolean menük elemei között. Ezek kimenete 1/0, van nyomógomb, és kapcsoló. Az el z ek (paraméter megadók) is kezel szervek. Lehet úgy is, hogy konstanst teszünk le, majd jobbcl.>change to control. Ilyen még a DialogControls>checkbox is.

-Kijelz elemek: Vannak az egyszer bbek, amik az indikátorok:



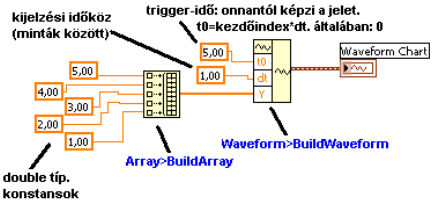
Boolean>RoundLed, String&Path>StringIndicator, Numeric>DigitalIndicator, Numeric>Num.ProgressBar, Num.>Meter és a bonyolultabbak, a grafikusok. Ezeknek csak bemenetük van. A grafikusok: A Graph menüben vannak:

Waveform chart: Ezzel lehet hullámformákat ábrázolni, mint az oszcillosz-kóppal. A bemenete hullámforma típusú. Direktbe rá lehet kötni hullámforma generátort (Waveform>Wav.f.Gen>).

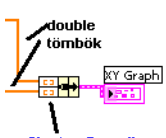


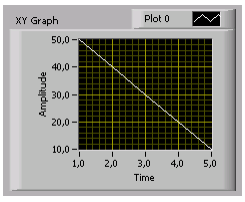
NEM KELL MINDEN KIVEZETÉST BEKÖTNI! Paraméterek állítása (ahol lehet): duplakatt a dobozra. Kijelzési határok: a kezel lapon átírható.

Vagy el állíthatunk neki hullámformát saját adatokból: A Wavef.>BuildWaveform segítségével. Ehhez double-tömb kell. Tömb el állítás: Array>BuildArray. Trigger id : honnan kezd dik.(0) Persze konstansok helyett más blokkok kimenetei is lehetnek. A BuildArray, és a másik letétel után kíván méretre húzható az alján, aszerint hogy hány paramétert akarunk megadni. Tömböt is ábrázol!!!



Graph>XY Graph: Ennek 2db 1D (double típ.) tömbb l el állított lista



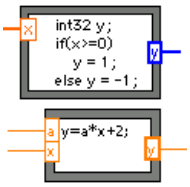


(cluster) a bemenete. A listát a Cluster>Bundle segítségével állítjuk el . A Bundle-t méretre (2bem) húzzuk, az alján. Fels listaelem: X tengely, alsó: Y.

Graph>CTLs>PolarPlot: Polár ábrát rajzol. Bemen adata: 1D tömb, melynek elemei: 2elem listák. A listák: (r,).Általában az r-ek tömbben adottak. Ebb l a megfelel bemenet el állítása: ForCiklus+Cluster.Boundle.

A Prog.files>N.I.>Labview>Examp.>Pict>demos.llb –ben van példafájl rá. Azt ki lehet másolni,és felh. Egyéb spéci: Radarkép, Smith diagram, 3D.

-Matematikai: A Numeric –menüben. Összeadás, kivonás, szorzás…, numerikus konstans, adott konstansok, függvények, komplex m veletek… A ContextHelp megmutatja milyen adattípusokkal m ködnek. (Help>ShowContextHelp)

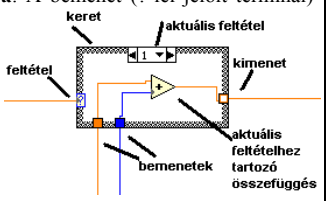


A Mathematics –menüben bonyolultabb függvények vannak, és megadhatók saját függvények is: >Formula. Ezt letesszük a blokk-diagramba, méretre húzzuk, beleírjuk a függvényt (mint c-ben.), majd ki/bemeneteket adunk hozzá: JobbCl.>AddInput vagy AddOutput. Ezután a létrejött terminálokba beírjuk a hozzárendelt változó nevét. Itt lehet megvalósítani az IF-es szerkezeteket is. A kimenet típusát deklarálni kell, ha más típusút akarunk, mint a bemenet.

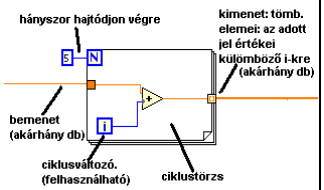
-Nagy dobozok: Programozási modulok. Keretként lehet megrajzolni, a bels m ködésüket a megszokott módon lehet dobozokkal kialakítani.

Változók: Globális/lokális lehet. A lokális csak a kereten belül érvényesek. Iránya: Kimeneti/bemeneti kiválasztható (JobbCl.>ChangeRead –bemeneti vagy ChangeWrite -kimeneti)

Case srtuktúra: A bemenet (?-lel jelölt terminál) értékét l függ en lehet kimenetet generálni. A vizsgált esetek default: 1 vagy 0 boolean típusú értékek. Lehet új eseteket hozzáadni: JobbCl.>AddCaseAfter. A klikk a legördül feltétel-menüre kell. Ezután beírhatjuk a vizsgálni kívánt értékeket. Végrehajtás: Kiválasztunk egy értéket, majd berajzoljuk a kimenet generálását. Kiválasztunk egy másik értéket, megrajzoljuk ahhoz is. Mindegyikhez. Amikor az els t megrajzoljuk, át lehet húzni vezetékeket a kereten, így használva/módosítva küls jeleket. Így a kereten magától keletkezik egy terminál. Akárhány db lehet.



For ciklus: Hasonlóan az el z höz. Itt a kimenetet el -állítjuk minden i-re, és ez tömbként elérhet . A ciklusváltozót maga generálja. 0…(N-1)-ig megy.



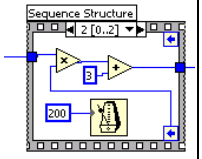
A célja: egy tömb el állítása.

Lassítás: ciklustörzsbe késleltetést kell tenni. Elvileg ms-ban kéri, de nekem kb 10szer lassabb. Jó a whilehoz is. Az egész VI-t lassítja: lép egyet, vár, lép, vár…

Lefutás után újraindul. A While is ugyanígy.

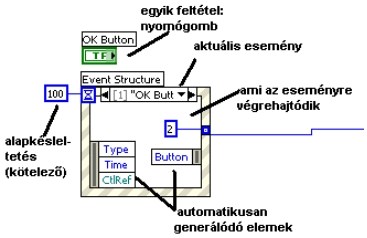
While ciklus: (feltételre várás) i-vel iterál. Van kilépési feltétel: (boolean) Beállítható, hogy ha a feltétel igaz, akkor folytat, vagy kilépjen. Feltételt lehet kreálni összehasonlítókkal: Comparison –menü. A feltételt bekötjük a kis dobozkába: vagy . A kivezetett jelek értéke a lefutás utáni, pl ha valamit iterációval kapunk meg. Közbüls jelek (részeredmény) csak belül használhatók fel.

Sequence: Egymás után végrehajtandó programrészek megadása. Az egyes filmkockákban (frame) létrehozhatók az egyes egymás utáni program-részek. Szerkesztéshez a fels legördül menüvel kiválasztható egy kocka. Hasonló a case-hez.



Minden bemenet felhasználható minden frame-ben, de kimenetet csak egy frame generálhat. Viszont átvihet adat egyik frame-b l a következ kre: Keret szélére JobbCl.>AddSequenceLocal: Ezzel az adott frame-nek lehet kimenetet generálni a következ frame-ek számára. (Ahol kim: , ahol bem: , bekötés után látszik így)

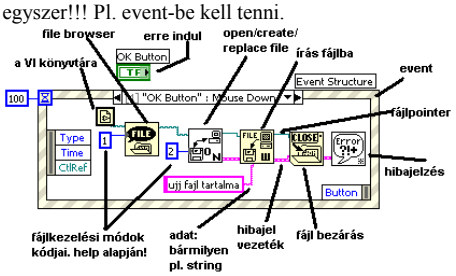
Event Structure: Adott eseményre (**élvezérelt**) bekövetkez program-részlet. Ez csak egyszer hajtódik végre, míg a ciklusok,++egyéb újra és újra.. Események: gombnyomás (mousedown), billenty (keydown), változó értékváltozása. Alapértelmezett végrehajtás is van. Ehhez tartozik a timeout-id zít . Defaultból 1 esemény van, ez a timeout. Új esemény hozzáadása: JobbCl.>Add-EventCase. Ekkor a menüben kiválasztjuk a forrást (el lapi kezel k/kijelz k. Pl. led), és hogy mire érzékeny: lenyom/felenged/értékvált…



-Fájlkezelés:

A FileI/O-menüben vannak az elemek. Fájltípusok: SpreadSheet (számokat ír, vagy hozzáf z ASCII karakterek formályában fájlba. Bemenete:tömb), karakteres (stringeket kezel), I16(integereket kezel, tömb formában. Ez bináris fájl). Vagy egyszer en általános fájlként is kezelhetjük.

Fájlkezelés menete: Megnyit/létrehoz, ír/olvas, lezár. Csak eseményre hajtódjon végre, és csak egyszer!!! Pl. event-be kell tenni.



A file írása-dobozt le lehet cserélni bármire: FileI/O> ReadFile, WriteFile, WriteToSpreadSheetFile… bármelyik. Hibajelz.:Time&Dialog>

Kikerülhet a brows-olás, ha a FileI/O>File-Constants –menüben lév dolgokat használjuk.

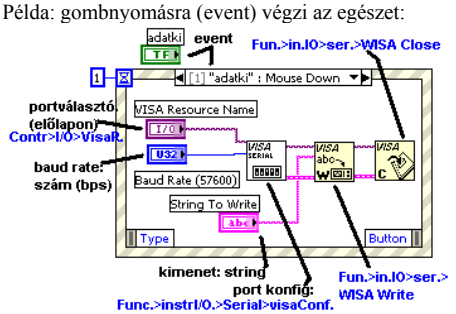
El lapra is kitehetjük: Controls>StringAnd-Path>FilePathControl. A korábbi megoldás akkor kérdez rá, amikor el akarja menteni, ez meg el re.

-I/O-elemek:

InstumentI/O> GPIB, Serial, VISA (bármilyen telepített), vagy kész m szer-driverek.

Sorosport:

Részfeladatok: Konfigurálás, írás/olvasás, lezárás. Példa: gombnyomásra (event) végzi az egészet:



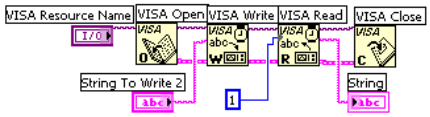
A portválasztó lehet Contr.>I/O>VisaResource, vagy Functions>Instr.I/O>VisaResourceNameConstant. Itt láthatók a gépre telepített VISA-kommunikációs driverek (Hardware Resource Driver). Az Agilent IOSuite programmal lehet könnyen kommunikációs csatornákat létrehozni/telepíteni. Visa Session: egy konkrét csatorna, (VisaResourceName-vezeték-lánc)

GPIB:

Gyári mér m szerek kezelésére való buszrendszer. Vezérl kártyán keresztül érhet k el. Karakteres parancsokkal (rövid stringek, pl: CLR) megy a kommunikáció, vezérlés+mérési eredmények betöltése, ezért szükség lehet karakter-szám-konverziókra (Funct.>String>String/Num.Conv.). Az egyes m szerekhez szoktak driver-t mellékelni a gyártók. VISA-SESSION-néven vannak. Ezeket *.lib fájlként bemásoljuk a LabVIEW\…\InstrLib\ -be. A m szereknek egyedi azonosítójuk van.

Össze lehet rakni saját m szer-drivert: A Func.>InstrI/O>Visa menüben vannak hozzá elemek: >VisaAdvanced>Interf.>VisaGpibCommand, és persze a VisaRead, VisaWrite. A szükséges parancsokat, és adatokat a gépkönyvek írják le.

Hozzá kell rendelni egy VISA resource name-et. (komm.csatorna: GPIB)



Egyéb: pl. USB, Bluetooth. Ezekhez kell drivereket telepíteni.

-Jelfeldolgozás:

waveform típusú adatok kezelése. Ez a 3 menü elemei. Itt lehet pl fájlba menteni: Wavef.>Wavef.FileI/O. Jelfeldolgozás: Wavef.>Wavef.Measurements (átlagolás, FFT, felfutási id , ampl. Torzítás, Sinad…). Wavef.>Wavef.Operations: további feldolgozás. Az Analyze menüben vannak még: Ablakolás, határolás, trigger generálás, generálás, sz r k (FIR, IIR), interpoláció, polár-dékárt konv., várható-érték, szórás, konvolúció, mátrix m veletek, PointByP.>array (a*x+b , skálázás).

Adat analízis: pl. sorosportról stringben kapunk adatokat (bájtok), majd konvertáljuk tömbbé (Func.>String>St.Array.Conv>St.ToByteArray), ezután a Wavef.>BuildWf. Segítségével hullám-formává alakítjuk. Ez már mérhet , kijelezhet , feldolgozható.

Programszerkezet:

Lefutás: Minden általában újra és újra végrehajtódik, ha az alapra tesszük. Persze csak ha a RUN CONTINUOUS-sal indítottuk. Ha event-ben van, akkor csak egyszer, ha sequence-ben, akkor minden frame-ben meg lehet mondani, hogy lefusson –e. Ciklusban iterációnként egyszer hajtódik végre minden. De lehet máshogy is: Pl. ApplicationsControl>Stop: ezzel a VI leáll, ha akarjuk. Vannak menük, Labv. kilépés...

Ha sima RUN módban használjuk, akkor az alap csak egyszer fut le. Viszont ha az alapra teszünk egy while ciklust, benne a kilépési feltétel mindig false, akkor így állandóan fut a VI.

Legjobb: A File>VI_Properties –nél beállíthatók a >Window_Apperance>Customize –nál, hogy: futás közben ne jelenjen meg scroll, menük… mert feleslegesek. Viszon ekkor kell a VI-nkbe stop/kilépési feltétel, gombhoz kötve. Érdemes még: …prop.>Execution –nál hogy Run when Load.

Saját VI-k használhatóvá tétele: SaveAs>NewVILibrary, majd a LabVVIEW\…\User.Lib\ -be mentjük.

Kijelzés id belisége: Ha kijelz elemet ciklusból kivezeetett jelre kötünk, csak a lefutás utáni értékeket jeleníti meg. Ha event-b l vezetünk ki jelet kijelzéshez, az csak a default-hoz tarozó értéket fogja jelezni, de ha az esemény-ablakba tesszük a kijelz t, akkor az az eseménykor kijelez, majd a kijelz n meg is marad az eredmény. Ugyanez van a cese-nél is. Esetleg a LocalVariable –lel lehet mást.

Sorosporti eszközkezelés: Gomb által indított event-struktúrában inicializáljuk a sorosportot, annak az inic.doboznak a kimenete egy sequence struktúrába megy, amiben az egyes kommunikációs fázisok vannak, az utolsóban a port-lezárás. Port-olvasási frame: addíg nem megy tovább, amíg az igényelt számú karakter be nem érkezett.

Tools>DataAcquisition>Daq solution wizard: Ezzel lehet a leggyorsabban adatfeldolgozó, kiértékel VI-ket létrehozni. Magától generál.

EXE készítés: csak Professional version esetében van ilyen opció: Tools>Build_Application_or… . Érdemes a beállításokat elmenteni, majd újrafelhasználni. Save as, és a Load.