LabView oktatási segédlet

TARTALOMJEGYZÉK

I. MÉRÉSI ADATOK FELDOLGOZÁSA

1 A LabView program készítésének alapjai

A LabView elindításakor 2 ablak jelenik meg: egy szürke és egy fehér hátterű. Az egyiken a virtuális műszerünk (VI=Virtual Instrument) előlapja – eredeti nevén PANEL – építhető fel, a másikon – amit a továbbiakban DIAGRAM-nak hívunk – a programunk által megvalósított műveletek láthatók. A PANEL-ablakon található kapcsolók, kijelzők, nyomógombok, forgatógombok a vezérlések, a DIAGRAM-on ezzel szemben matematikai, logikai, string-műveletek, ciklusok, stb. láthatók. A két ablak között egyszerű egérkattintásokkal lehet kapcsolgatni, vagy a Windows menü 'Show Panel/Diagram' menüpontjait választva. (Ezt kell tenni akkor is, ha véletlenül bezártuk a DIAGRAM ablakot, és ismét szükségünk lenne rá). Egyébként a menürendszer is ugyanaz mindkét ablakban, és teljesen mindegy, hogy melyikben választunk ki valamit, ugyanaz lesz a hatása. A két ablak egyetlen programot takar, melynek PANEL és DIAGRAM ablakai szorosan összetartoznak, egymástól elválaszthatatlanok.
Vezérlés vagy művelet elhelyezése: jobb egérgomb után a csoportot kell kiválasztani, amelybe a vezérlés vagy művelet tartozik, majd a csoporton belül a szükséges elemet elhelyezni.
A vezérlések és műveletek csoportjai logikusan vannak kialakítva, leginkább a kezelt adat típusa szerint érdemes kutatni. A csoportok gyors megismerésére érdemes egy kis időt fordítani, mert ezután nem az egyes elemek keresgélésével fognak eltelni a gyakorlásra szánt órák.

1.1 A legfontosabb vezérlések

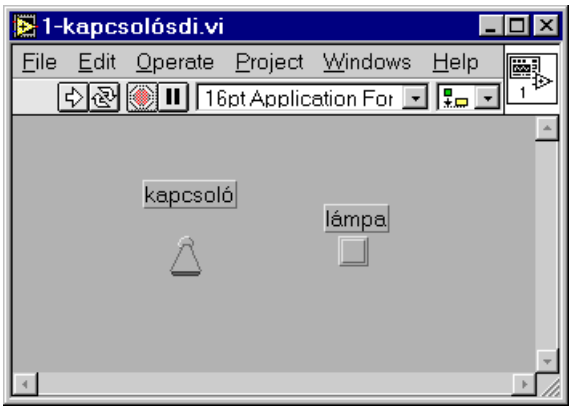
Numeric: tartalmazza az összes számokkal kapcsolatos vezérlést és kijelzőt. Valós és egész számokra ugyanazokat. A megjelenésüktől eltekintve teljesen hasonlóan funkcionálnak, persze alapvetően két csoportra bonthatók aszerint, hogy adat bevitelére vagy megjelenítésére szolgálnak.
Boolean: minden logikai típusú adat bevitelével és megjelenítésével kapcsolatos eszközök. Ha sikerül különbséget tennünk a kapcsolók és a lámpák között, akkor ezeken a csoportokon belül már szinte mindegy, hogy melyiket választjuk a feladatunk megoldásához.
String & Table: sztringek és táblázatok csoportja
List & Ring: legördülő listák és választómezők csoportja
Array & Cluster: tömbök és rekordok létrehozására és megjelenítésére használatos eszközök.
Graph: Grafikus megjelenítő eszközök
…
Decorations: Csak arra valók, hogy az előlapon elhelyezett objektumainkat esztétikusan kialakított csoportokba rendezzük, elválasszuk őket egymástól, stb.

1.2 A legfontosabb műveletek

Structures: A program szerkezetét, a futását alakító lehetőségek csoportja.
Numeric: Minden olyan művelet, amely számokkal végezhető. Mivel ezek száma nagy, további csoportokat alakítottak ki a trigonometrikus, logaritmikus, stb. függvények számára, és külön csoportban vannak az előre definiált matematikai és fizikai konstansok is.
Boolean: Az összes logikai függvény csoportja, az egyszerű invertálástól kezdve a bonyolultabb, tömbökön is elvégezhető műveletekig.
String: Alapvető string-függvények halmaza.
Array: Tömbök kezelését segítő függvények
Cluster: Rekordok kezelését segítő függvények
Comparison: Összehasonlításra szolgáló függvények
Time & Dialog: Idő lekérdezése, mérése, dialógusablak megjelenítése, stb.
File I/O: File-műveletek
…
Tutorial: A LabView programok készítésének elsajátításához hasznos segédeszközöket tartalmazza.

1.3 'Kapcsoló és lámpa' egyszerű programja

A legelső programjaink nagyon egyszerűek lesznek, inkább csak arra valók, hogy bemutassák a LabView objektum-orientáltságát, a legördülő menük használatát. Az első feladat csupán annyi, hogy a képernyőn megjelenő lámpa fénye jelezze egy szintén a képernyőn elhelyezett kapcsoló vagy nyomógomb állapotát.

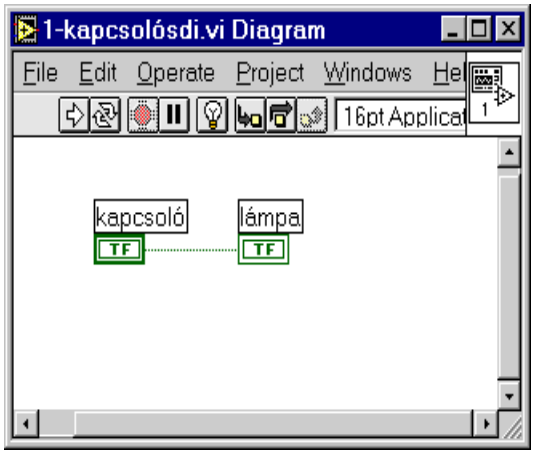


1.3.1 Kapcsoló elhelyezése és elnevezése

A kapcsolót vagy nyomógombot a szürke hátterű ablakban egy semleges területen elvégzett jobb egérgombbal lehet elhelyezni, természetesen a Boolean-csoportból lehet kikeresni a szimpatikus vezérlést. A képernyőre helyezve érdemes rögtön a nevét is begépelni (pl. 'kapcsoló'), ha ezt elmulasztjuk, a későbbi elnevezés egy kicsit bonyolultabb lesz.

1.3.2 Lámpa elhelyezése és elnevezése

Ugyanabból a csoportból, mint a kapcsolót. Ha a nevének begépelése helyett véletlenül valami mást kezdtünk csinálni, a következőket kell tenni: jobb egérgombbal a lámpára kattintva a legördülő menüből a Show->Label menüpontot választva pótolhatjuk a hiányzó nevet.



1.3.3 A kapcsolat létrehozása

A kapcsolat már a virtuális műszer belsejében kell, hogy létrejöjjön, ezért át kell kapcsolnunk a DIAGRAM ablakra, és ott megkeresnünk a két előlapra szerelt elem (=CONTROL) csatlakozási pontjait (=TERMINAL). Elég könnyű lesz megtalálni a két zöld téglalapot, főleg, ha nevük is van. Bonyolultabb helyzetben úgy lehet megtalálni egy vezérlés csatlakozási pontját ill. fordítva, hogy a jobb egérgombra legördülő menüből a 'Find Control/Terminal' menüpontot választjuk.
A két zöld színű téglalap összekötéséhez szükségünk lesz vezetékre, mely a Tools Palettán található. (Window menü -> Show Tools Palette.) Egy feltekert vezeték a jele, de emellett még más fontos eszközök is megtalálhatók itt.

1.3.4 Tools Palette-n található eszközök:

Működtető kéz: tulajdonképpen ezzel lehet változtatni az előlapra helyezett elemek állapotán (kapcsolni a kapcsolót, tekerni a forgatógombot, vagy léptetgetni egy állítható számkereket fölfelé és lefelé). Rögtön kapcsoljuk is át a kapcsolót a másik állapotába !
Szerkesztőnyíl: A Windows egyéb programjaiban már megszokott szerkesztési funkciók mindegyike ezzel valósítható meg. Kijelölés (kijelölés után lehet törölni a DEL billentyűvel), mozgatás, átméretezés, stb. A vezetékekre is ugyanúgy működik, mint minden másra, kivételt jelent az egyszeres, a kétszeres és a háromszoros kattintás. Egyszer rákattintva egy vezetékre annak egyetlen töréspont nélküli szegmensét tudjuk kijelölni, dupla kattintással egy teljes vezetékszakaszt (a legközelebbi elágazásig vagy TERMINAL-ig), háromszoros kattintással pedig minden vezetéket, amely a kiválasztottal kapcsolatban van. A kijelölés után a kurzor-nylakkal finoman, SHIFT gombbal és a kurzor-nyilakkal gyorsabban lehet mozgatni a vezetékszakaszokat.
Szöveg-beírás: Az 'A' betűre kapcsolva bárhová elhelyezhetünk feliratokat az előlapon, vagy megjegyzéseket a programban, és ezzel az eszközzel lehet módosítani az elnevezéseket és tartományok végértékeit is.
Vezeték: Ezt már használtuk a TERMINAL-ok összekötéséhez. Egy kattintás a villogó csatlakozón, a további kattintások a vezeték megtörését teszik lehetővé, végül egy utolsó kattintás a végpont TERMINAL-ján.
…
Színek állítása: A Tools Palette legalsó elemére kattintva lehetőség adódik az ablakok és minden objektum színeinek megváltoztatására. A két fedésben levő négyzet jelképezi az objektumok színeit (háttérszín is !), így minden objektumhoz állapotonként két szín rendelhető. Pl. egy nyomógomb lehet piros és zöld attól függően, hogy meg van-e nyomva, és mindez megvalósítható kék és sárga háttérszínnel. A színezés mindig arra az állapotra vonatkozik, amelyben a vezérlés éppen van ! (A működtető kézzel viszont bármikor átváltható a másik állapotba, hogy annak a színeit is állíthatóvá tegyük !)

A fenti funkciók a SPACE billentyűvel egyszerűbben válthatók (PANEL-ablakban a működtető kéz és a szerkesztőnyíl, DIAGRAM-ablakban az összekötő huzal és a szerkesztőnyíl gyors váltogatását teszi lehetővé a Tools Palettán való keresgélés nélkül)

1.3.5 A program futtatása

A futtatásra két gomb áll rendelkezésre, közvetlenül a menü alatt. Az egyik egy egyszeres futtatást hajt végre, a másik ciklikusan ismételgeti a program végrehajtását. Az egyszeres futtatás kipróbálásához állítsuk a kapcsolót a kívánt állásba, majd indítsuk el a programot. Remélhetőleg a másodperc töredéke alatt lefut, és a lámpa jelzi a kapcsoló állapotát. Minden kapcsolóállítást követnie kell az egyszeres futtatásnak, hogy az eredményt a lámpán lássuk.
A ciklikus futtatás ebben a feladatban látványosabb megoldást nyújt: rögtön látható a váltás eredménye. A leállítás csak a Stop gombbal lehetséges, ami ugyanabban a sorban helyezkedik el. DE A KÉSŐBBIEKBEN A CIKLIKUS FUTTATÁS HASZNÁLATÁT KERÜLJÜK !!! Inkább módosítsuk úgy a programot, hogy addig működjön egyszeres futtatással, amíg le nem állítjuk !

1.4 Folyamatos működés megvalósítása ciklussal

Egy egyszerű ciklusba foglaljuk az eddigi programunkat, és egy külön erre a célra elhelyezett 'STOP' gomb megnyomásától tesszük függővé a további futtatást.

1.4.1 Ciklusválasztás és elhelyezés

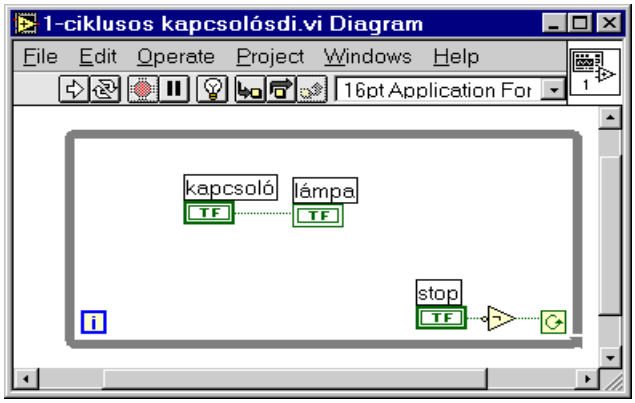
A ciklusszámot előre nem tudjuk, ezért a WHILE ciklust választjuk a DIAGRAM ablak 'Structures' csoportjában. A programot ezzel a ciklussal be kell keríteni, lenyomva tartott egérgomb mellett. Két dolog jelenik meg a ciklusban: 'I' a ciklusszámláló mindenkori értékét tartalmazza, ezzel egyenlőre nem foglalkozunk; a jobb alsó sarokban pedig megjelent a kilépési feltétel.

1.4.2 Stop-gomb elhelyezése

Az előlapra helyezzünk el egy STOP-gombot (a boolean csoportból).

1.4.3 Kilépési feltétel megvalósítása inverteren keresztül

A STOP gomb csatlakozást pontján akkor kapunk IGAZ-at, ha valaki megnyomta a gombot. A WHILE ciklus viszont akkor áll le, ha HAMIS értéket kap a kilépési feltételére. Ezért szükség lesz még egy inverterre (=NOT) a boolean függvények közül, amelyet a Stop gomb terminálja és a WHILE cilklus kilépési feltétele közé kell kapcsolnunk. Az inverter bekötésekor figyelni kell arra, hogy melyik oldala villog. Minden LabView függvény bal oldalán vannak a bemenetek, a jobb oldalán pedig a kimenetek. Emiatt a programok is általában úgy néznek ki, hogy a vezetékekben az adatok balról jobbra futnak.
Az összehuzalozás után már csak az egyszeres futtatás gombot használjuk !



1.5 Áttérés a valós számokra

A legelső programunk elkészült. Most úgy fogjuk alakítani, hogy valós számokkal dolgozzon.

1.5.1 A kapcsoló kicserélése forgatógombra

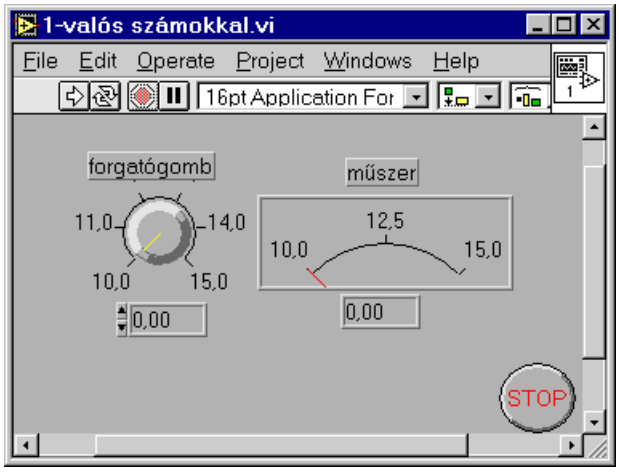
Megtehetnénk azt is, hogy szerkesztőnyíllal kijelölve a kapcsolót kitöröljük, és helyette egy teljesen új forgatógombot teszünk a képernyőre. Megspórolunk egy huzalozást, ha ehelyett a kapcsoló legördülő menüjéből a Replace menüpontot választva keressük ki a Numeric csoportból a forgatógombot.

1.5.2 Hasonlóan a lámpa kicserélése pl. egy mérőműszerre

A mérőműszer helyett persze választhatjuk a hőmérőt, a tartályt, vagy a sebességmérőt is.

1.5.3 Átnevezések az új helyzetnek megfelelően

Hogy az új vezérlések ne a 'kapcsoló' ill. a 'lámpa' nevet viseljék, gyakoroljuk a nevek módosítását ! (Tools Palette - 'A')



1.5.4 Méréstartományok átalakítása

Szintén a Tools Palette 'A' eszközét használva tudjuk módosítani a tartományokat is a 0…1-ről mondjuk 10-ről…15-re. A módosításhoz pontosan a számokra kell kattintani, különben egyszerű feliratként tudjuk csak elhelyezni a méréstartományt jelentő számokat !
A kipróbálás után érdemes megfigyelni, hogyan alakultak a vezetékek és a csatlakozási pontok színei. A Labview ugyanis a színekkel jelzi ezek adattípusát:
Zöld: Boolean
Narancs: Valós számok
Kék: Egész számok
Lila: Stringek

1.6 A szerepek (vezérlés és megjelenítés) felcserélése

Egy másik dolgot is érdemes megfigyelni: Azok a terminálok. amelyek adatot szolgáltatnak a program számára (pl. a felhasználó által állítható nyomógomb vagy potenciométer termináljai), vastag kerettel rendelkeznek ellentétben azokkal, amelyek a programban keletkezett adatokat megjelenítik (pl. hőmérséklet-kijelző, sebességmérő műszer, stb.). A vastag keretes, adatbevitelre szolgáló elemek a 'CONTROL' nevet, ezzel szemben a megjelenítő elemek az INDICATOR nevet viselik.
A LabView program futtathatóságának feltétele, hogy egy vezetéken maximálisan 1 db adatot szolgáltató elem legyen (de legalább 1 db feltétlenül legyen), és ezen kívül legalább 1, de nem korlátozott számú adatot feldolgozó elem. Ha ez a feltétel nem teljesül (pl. 2 CONTROL, vagy csak INDICATOR-ok helyezkednek el egy ilyen vezetéken), akkor a LabView ezt a vezeték feketére színezésével és a futtató nyíl összetörésével jelzi.
Az előbb említett két csoport közötti átalakítás persze megengedett, így a LabView-ban olyan dolgok is megvalósíthatók, amelyek a valóságban nehezen képzelhetők el. Pl. hogy egy műszer mutatóját mozgatva tekeredjen egy potenciométer. Ennek eléréséhez csupán mindkét elem legördülő menüjének első sorát kell választani: 'Change To Indicator/Control', mellyel az ellenkező típusúvá alakítjuk az elemet. A mérőműszerünk mostantól adatot szolgáltat, a potenciométer pedig megjelenít. A futtatás után valóban a mutatót kell mozgatnunk.
A működés másik feltétele, hogy ne legyen hurok az adatfolyamban. Ekkor ugyanis nincs kezdőpont, ahonnan a LabView el tud indulni a végrehajtáskor: az egész hurok fekete szaggatott vonallá válik.

2 Egyszerű hőmérséklet-regisztrálás és határérték-figyelés

A következőkben tovább alakítjuk a programunkat. A kézi adatbevitel – potenciométer tekergetése - helyett áttérünk egy a LabView-felhasználását tekintve sokkal jelentősebb alkalmazási területre: a valamilyen technológiai folyamatból érkező mérési adatok feldolgozására. A legkézenfekvőbb példa erre egy hőmérséklet-mérő berendezés adatainak beolvasása, és valamilyen egyszerű átlag, minimum vagy maximum kiszámítása.

2.1 A 'Tutorial'-ban található demo-hőmérő használata

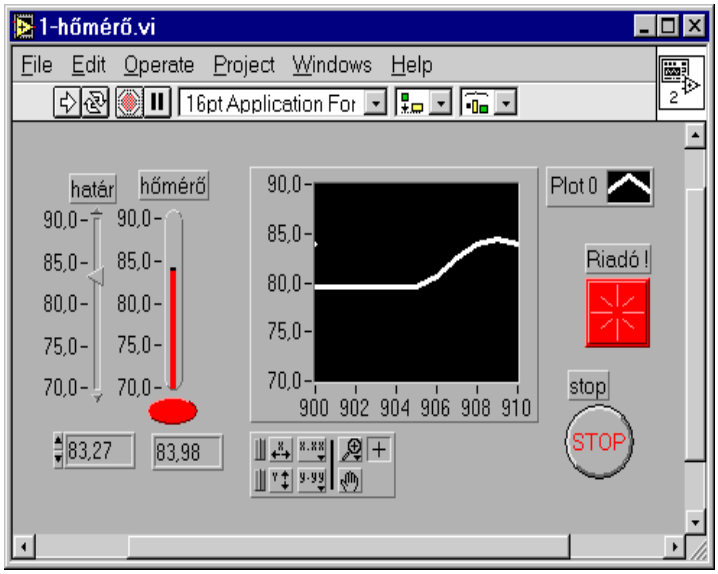
Kifejezetten gyakorlási célból a DIAGRAM ablak 'Tutorial' csoportjában megtalálható egy hőmérő-szimuláció függvény, melyet elhelyezve az ablakon minden egyes kiolvasás alkalmával más és más hőmérsékleteket kapunk (80 °C környékén). A függvény bal oldalára semmit nem kell kötnünk, csak egyszerűen a jobb oldalán megjelenő kimenetét rendszeres időközönként olvasgatnunk. Generátorként fog tehát működni, ezért nyugodtan letörölhetjük a legutóbb adatszolgáltatóként használt vezérlésünket a CONTROL ablakról (szerkesztőnyíllal kijelöljük, majd DELETE-gomb), és helyette elhelyezhetjük a 'Tutorial' csoport 'Thermometer.vi' függvényét. Ennek természetesen nincs képe az előlapon, hiszen ez egyfajta belső véletlenszám-generátorként működik.

Az előző feladatban kijelzőként használt vezérlést – az átalakított potenciométert – a 'Replace' utasítással cseréljük le egy hőmérőre (a folyadékhőmérőt a CONTROL panelen jobb gombot nyomva a 'Numeric' csoportban találhatjuk meg), majd méretezzük át ennek méréstartományát kb. 70…90 °C-ra. Ehhez a 'Tools Palette'-n kell kiválasztani az 'A' betűt, majd pontosan a módosítandó skálavég számjegyeire kattintva azokat átírni.
A futtatás alatt nagyon gyorsan változó hőmérsékleteket fogunk látni a képernyőn.

2.2 A program végrehajtásának lassítása időzítés elhelyezésével

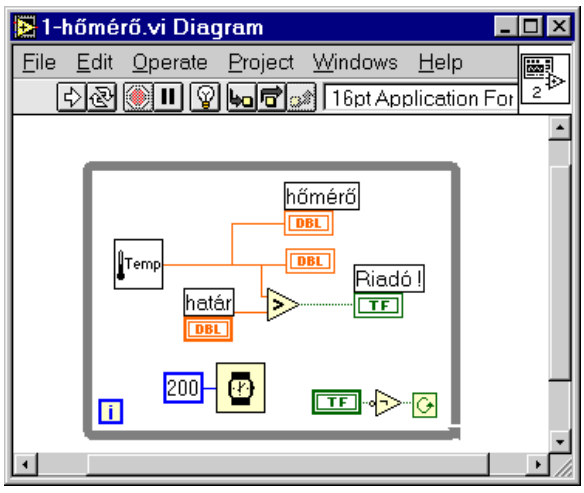
Hogy a mérések valóságosabbnak tűnjenek, és könnyebben követhetők legyenek, kissé le kell lassítani a program futását. Ezt a legegyszerűbben úgy érhetjük el, hogy a 'Time c Dialog' csoportból egy 'Wait [ms]' függvényt helyezünk még el a cikluson belül valahol. Ezzel a függvénnyel bal oldalán közölni kell, hogy hány ms ideig függessze fel a ciklusmag végrehajtását, és ezt a ciklus minden egyes lefutásakor meg fogja tenni. Célszerű tehát egy konstansban rögzíteni mondjuk 200-at, és így másodpercenként ötször hajtja majd végre a program a hőmérsékletmérést és kijelzést. A konstans elhelyezésére és bekötésére két nagyon egyszerű módszer van: az egyik, hogy a 'Numeric' függvényei között az alsó sorban megtaláljuk a kék színű konstans-ábrát, elhelyezve a képernyőn rögtön lehetőségünk van rögzíteni az értékét (mely később az 'A' betűvel módosítható), és ezután a huzalozással ugyanúgy beköthető az időzítő bal oldalára, mint bármi más. A másik módszer még gyorsabb: huzalozás üzemmódban a villogó kapcson jobb egérgombbal kiválasztjuk a 'Create Constant' menüpontot, majd beírjuk az értéket.

2.3 Idődiagramok készítése



Van néhány diagram-rajzoló vezérlés a LabView-ban, melyek közül egyenlőre csak egyet: a WaveForm Chart-ot fogjuk használni. Helyezzük el az előlapon, állítsuk be a méréstartományát szintén 70…90 °C-ra, majd kössük rá ezt is a hőmérsékletmérő függvény kimenetére ! A grafikon bár egymás után kapja a mérési eredményeket, képes megjegyezni a korábban kapott eredményeket, és azokat görgetve idődiagram-szerűen megjeleníteni. Néhány egyszerű, de hasznos lehetőség az X és Y irányú automatikus méréstartomány beállítás, a grafikon mozgatása, a megjelenítés módjának, színének, formájának, vastagságának átállítása.

2.4 Határérték-figyelés beépítése



Ha van még hely a cikluson belül, gond nélkül – egyéb esetekben a ciklus keretének egyszerű megnövelése után – további funkciókat építhetünk a programba. Ilyen pl. a határérték-figyelés, mely során egy egyenlőre konstansként beírt hőmérsékletnél magasabb értékekre egy felvillanó lámpával kell jeleznie a programnak. A határérték-figyelés csupán abból áll, hogy a mért értéket állandóan összehasonlítjuk mondjuk 82,5 °C-szal. Az összehasonlításra a 'Comparison'-csoport 'Greater than' függvényét használjuk (x>y ?), úgy, hogy annak felső (x) bemenetére a mért hőmérsékletet, alsó (y) bemenetére az általunk most 82,5 °C-ra beállított konstans hőmérsékletet kapcsoljuk. A függvény kimenete IGAZ/HAMIS típusú (zöld szín !), ezért egyszerűen megjeleníthető egy az előlapon elhelyezett lámpa segítségével.

2.5 Állítható szintű határérték-figyelés

Egy finomítása az előző programnak az lehet, ha a határérték már a felhasználó által beállítható. Ehhez helyezzünk el egy potenciométert az előlapon, méréstartományát igazítsuk a hőmérséklet-tartományhoz, és ezt kapcsoljuk az előző konstans helyére.

3 Mérési adatok feldolgozásának alapjai

Adott tehát egy mérési adatokat jól és látványosan megjelenítő program, ami egyenlőre nem tud számolni semmit. Ha nincs elég hely a DIAGRAM ablakon, töröljük le a határérték-figyeléssel kapcsolatos programrészeket !

3.1 Előre meghatározott számú mérés elvégzése

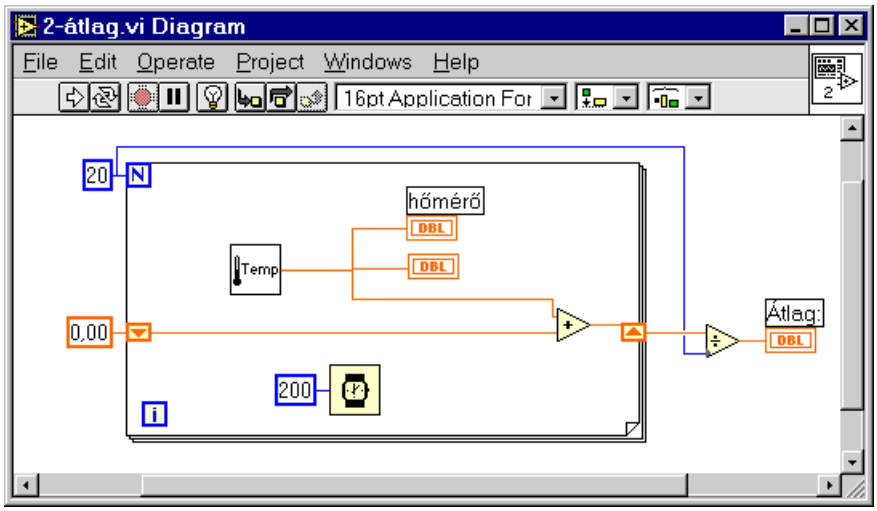
Töröljük le az előlapról a STOP gombot, és cseréljük le a WHILE-ciklust FOR ciklusra (’Replace’ a ciklus keretén nyomott jobb egérgomb után), így előre beállított számú mérés elvégzése után a programunk automatikusan le fog állni. Még annyit kell megtennünk, hogy a FOR ciklus bal felső sarkában található ’N’-be kívülről egy konstanssal beállítjuk a ciklusszámot (’Create Constant’, majd írjuk be pl.: 20).

3.2 Átlag kiszámolása

3.2.1 Shift-regiszter alkalmazása

Az átlag kiszámolásához szükség lesz egy olyan tárolóra, amelynek a ciklus minden egyes végrehajtásakor ki tudjuk olvasni a korábbi értékét, és rendelkezni tudunk a jövőbeli értékéről. Ebben gyűjtenénk a mért hőmérsékletek összegét, amelyet a program végén elosztva a ciklusok számával a számtani átlag adódna. Jelenlegi ismereteink szerint ezt nem tudjuk megoldani, hiszen rögtön zárt hurok képződne egy olyan összeadással, ahol a korábbi kimenetet ismét a bemenetre kapcsoljuk. Erre a LabView fekete szaggatott adatvezetékkel reagál.
A megoldás a ciklusoknál alkalmazható ’Shift-register’-nek nevezett tárolókban rejlik, melyekből egyet-egyet a ciklus szélén, jobb gombbal kérhetünk - ’Add Shift Register’. A ciklus bal oldalán feltűnő háromszöggel jelzett regiszterből a ciklus minden egyes lefutásakor ki tudjuk olvasni annak előző ciklusban beállított értékét, a jobb oldali regiszterben pedig ennek megfelelően rendelkezhetünk (rendelkeznünk kell !) arról, hogy milyen értéket vegyen fel majd a következő végrehajtáskor. A legelső végrehajtás előtti értékét (=kezdeti érték) balról kívülről kell általában egy konstanssal beállítanunk (FONTOS, hogy beállítsuk, mert a LabView nem nullázza a Shift regisztereket a következő futtatáskor !!!), a legutolsó végrehajtáskor beállított értékét pedig szintén cikluson kívülről, a jobb oldali jelből lehet kiolvasni.
Az átlagszámításhoz kérjünk egy ilyen Shift-regisztert, állítsuk a kezdeti értékét nullára (a ’Numeric’-ból elhelyezett kék konstanssal), és a cikluson belül az ’előző érték’-éhez mindig adjuk hozzá az akkori mérési eredményt, az összeget pedig vezessük a ’következő érték’-be. Az összeadás függvényét a ’Numeric’ csoportban találjuk.

3.2.2 Átlagszámítás



A FOR ciklus kilépésekor már csak el kell osztanunk a kapott összeget (=a Shift regiszter utolsó értéke jobbról kívülről kiolvasva) a ciklusok számával (vagyis ugyanazzal a konstanssal, amit az ’N’-be kötöttünk – célszerű onnan elhozni egy vezetékkel). Az osztás végeredményét egy ’Numeric Indicator’-ral megjeleníthetjük, de a leggyorsabb megoldás, ha ezt is a LabView-tól kérjük: Amikor villog az osztás függvényének kimenete a huzalozás közben, jobb

gomb után válasszuk a 'Create Indicator' menüpontot. Tipikus hiba szokott lenni az osztási művelet (x/y) fordított bekötése, ezért figyeljünk arra, hogy a felső (x) bemenetre kapcsoljuk az összeget, és az alsóra (y) a ciklusok számát !
Ha nem a cikluson kívül jelenítjük meg a hányadost, hanem az osztást a cikluson belül végrehajtjuk az addigi összeggel és az addigi ciklusszámmal, akkor folyamatosan meg tudjuk jeleníteni az addigi mérések átlagát is. Ez sem sokkal bonyolultabb, csak arra kell figyelni, hogy a ciklus bal alsó sarkában kiolvasott 'I' ciklusváltozó 0-ról indul, és N-1-ig fut (tehát eggyel meg kell növelni az értékét, mielőtt osztóként bevezetjük az osztás függvénybe)

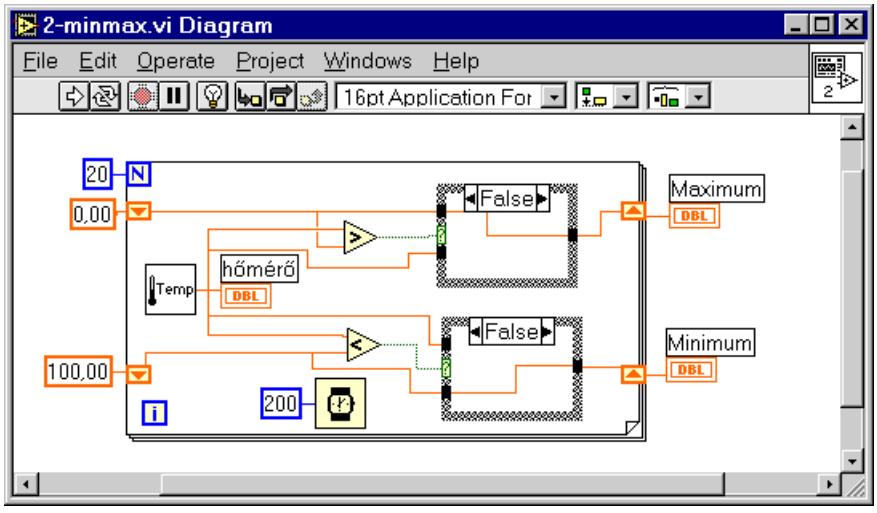
3.3 Minimum és maximum figyelése

3.3.1 Shift-regiszterek elhelyezése

A mérések alatt előforduló legkisebb és legnagyobb értékek tárolása sem komolyabb feladat az átlagszámításnál, de egy további struktúra-elemet, az esetszétválasztást ('CASE') kell hozzá megismernünk. Az algoritmus lényege, hogy egy shift-regiszterben tároljuk az eddigi legkisebb, és egy másik shift-regiszterben az eddigi legnagyobb értékeket. Ha a mérések során az eddigi legkisebb értéknél kisebbet mérünk, vagy az eddigi legnagyobbnál nagyobbat, akkor a megfelelő shift-regiszter eddig tárolt értékét lecseréljük a jelenleg mért hőmérsékletre.
Kérjünk tehát két shift-regisztert, egyiket fent (maximum), egyiket len (minimum) helyezzük el a ciklus szélein. Adjunk a minimumnak mondjuk 100-as, a maximumnak 0-ás kezdeti értékeket (a minimumnak nagyobb kezdeti érték kell, mint a várható legnagyobb mérési eredmény !).

3.3.2 Esetszétválasztás

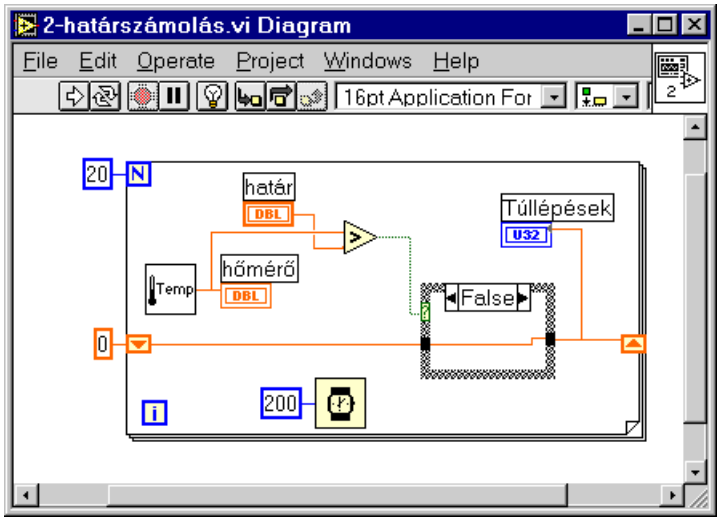
helyezzünk el két 'CASE' struktúrát a ciklus jobb felső és jobb alsó sarkában. Ezek a más programozási nyelvekben megszokott IF…THEN…ELSE… struktúrának felelnek meg a leginkább: a zöld kérdőjellel ábrázolt feltétel-bemenetre IGAZ/HAMIS típusú adatot kell juttatnunk, ami alapján a 'CASE' struktúra igaz vagy hamis ágán (helyesebben lapján) megrajzolt programrészek fognak lefutni. A két lap tulajdonképpen egymás mögött helyezkedik el, egyszerre mindig csak egy látható, és a felső ki nyilakkal lehet váltogatni közöttük. (True=igaz; False=hamis)
Mi történjen tehát, ha a mért érték (x) nagyobb, mint az eddigi maximum (y). A felhelyezett 'Greater' függvény (x>y?) kimenetét rákötjük a felső CASE struktúra feltétel bemenetére, és megrajzoljuk annak True lapját: Legyen a Shift regiszter új értéke a pillanatnyi mért eredmény. Vagyis a mért eredményt behozzuk a Case struktúra lapjára (kattintsunk a vezetékkel legalább egyet a struktúrán belül !), majd továbbvisszük jobbra ki, és rákötjük a Shift regiszterre. A CASE struktúrába be- és kilépve két speciális jel keletkezett. Ezek az alagutak biztosítják a struktúra kommunikációját a külvilággal. A bemeneti alagút fekete színe jelzi, hogy itt minden rendben, a kimeneti fehér színű alagúttal viszont még gond van. A LabView ugyanis még nem tudja, hogy a másik ágon/lapon milyen értéket adjon ennek az alagútnak. Itt tehát nem csak a THEN ágban kell rendelkezni a shift-regiszter jövőjéről, hanem az ELSE (False) ágban is kell valamit csinálni: bár nem is kell változtatni a shift-regiszter értékén, a korábbi értékét be kell vezetni egy harmadik alagúton, és a False lapon a fehér színű kimeneti alagútra rákapcsolni.



Az alagutak ugyanúgy mozgathatók és törölhetők, mint bármi más, a keletkezésük automatikusan megtörténik, ha egy vezetéket átviszünk a struktúra falán. Tipikus hiba szokott lenni, hogy a szerkesztgetések miatt több alagút keletkezik, mint amennyi szükséges, és ezek még ráadásul egymás alatt fedésben is el tudnak helyezkedni, így szinte észrevehetetlenek.
A minimum figyelése hasonló az előzőekben

leírtakhoz, persze most a 'kisebb, mint' függvényt kell majd használni a másik CASE struktúra feltételének meghatározásában.
Végül a két shift-regiszterben legutoljára kialakuló értékeket meg kell jeleníteni egy-egy indicatorban. (Kívülről kiolvasva a jobb oldali regiszter-jeleket). Vagy folyamatosan is megjeleníthetjük az addigi legnagyobb és legkisebb értékeket, ha mindezt a cikluson belül tesszük.

3.4 Határértéket túllépő értékek előfordulásának számolása



Nem bonyolult feladat az sem, hogy egy adott határérték alatti vagy feletti, esetleg egy adott tartományba eső mérési eredmények előfordulását megszámoljuk egy ilyen mérési sorozat alkalmával. Ehhez is mindössze egy shift-regiszter kell (ez lesz a számláló), amelyet minden programindításkor lenullázunk (kívülről balról 0-át kapcsolunk rá), a cikluson belül pedig egy CASE struktúrában eggyel növeljük az értékét, ha szükséges (a True lapon a shift regisztert legalább egy belső kattintással átvezetve közbeiktatunk egy '+1'-es függvényt), ill. megtartjuk a korábbi értékét, ha nem teljesült a feltétel (False lapon egyszerűen átkötjük a bemeneti alagutat a kimenetire, hogy az érték ne változzon). A '+1'-es függvény a legegyszerűbb lehetőség a növelésre, természetesen megoldható lenne a hagyományos összeadással és egy konstans '1'-essel is. Úgy célszerű beszúrni egy már létező vezetékbe, hogy a vezetékre jobb gombbal kattintva az 'Insert' menüpontot választjuk, majd a megjelenő 'Numeric' csoportból a kívánt függvényt.
Attól függően, hogy a shift-regisztert a cikluson belül vagy azon kívül jelenítjük meg, a program vagy folyamatosan jelzi futás közben, hogy hányadik előfordulásnál tart, vagy csak a futtatás végén írja ki az esetek számát.

4 Egyéb egyszerűen használható adatforrások

A hőmérsékletet szolgáltató 'Thermometer.vi'-n kívül van még számos lehetőség a mérendő jel előállítására:

4.1 Jelalak-generátor

Ennek a VI-nek egyetlen bemeneti paramétereként a ciklusváltozót célszerű megadni, mely minden ciklusfajtában a bal alsó sarokban található kék színű 'I'-ből olvasható ki. Egy eléggé speciális, látványos kimeneti jelalakot szolgáltat ez esetben, melyet ugyanúgy fel lehet használni az előző feladatokban átlagoláshoz, minimum és maximum meghatározásához.

4.2 Feszültséggenerátor

A feszültséggenerátor működését tekintve teljesen megegyezik a hőmérsékletmérővel, (egyébként ez van beépítve a hőmérsékletmérő programjába is forrásként), kiemenetén különböző feszültségként értelmezhető számok jelennek meg

4.3 Véletlenszám-generátor

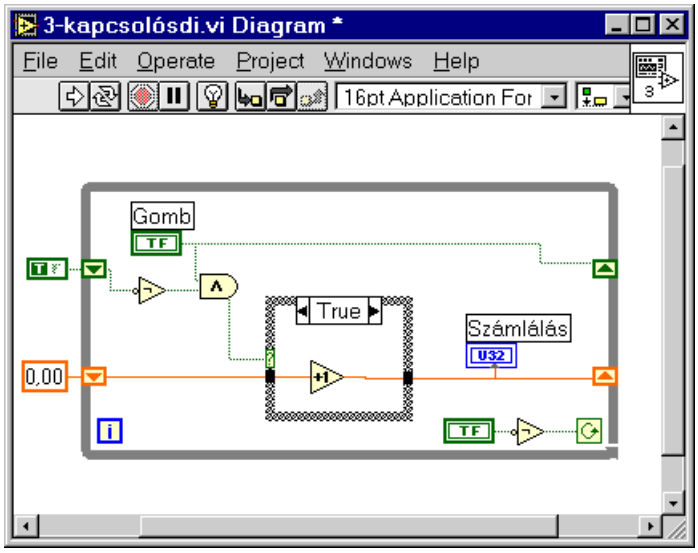
A véletlenszám-generátor már nem a 'Tutorial' csoportban, hanem a 'Numeric'-ban található meg (egy dobókockát kell keresni), és minden kiolvasásra egy 0 és 1 közé eső véletlen számot állít elő. Kiválóan alkalmas az ismertetett szimulációs programjaink tesztelésére, azzal a kiegészítéssel, hogy a feladatban meghatározzuk, milyen tartományba eső véletlen számot kell előállítanunk.
A 0…1 tartományt át kell alakítanunk a megadott tartományra, mégpedig úgy, hogy annak teljes szélességével megszorozzuk a véletlen számunkat, majd a 0-át a megadott tartomány kezdeti pontjába toljuk egy összeadás vagy kivonás segítségével. Pl. ha –5 és +10 közé eső véletlen számot kell generálnunk, a dobókockából kapott értéket megszorozzuk 15-tel, majd 5-öt kivonunk belőle.

II. FREKVENCIAMÉRŐ KÉSZÍTÉSE

1 Nyomógomb megnyomásának számolása

A frekvenciamérés programjának megvalósításához feltétlenül szükséges, hogy meg tudjuk számolni, hányszor kapcsoltak fel pl. egy kapcsolót vagy hányszor nyomtak meg egy nyomógombot. A számlálást eddigi ismereteink alapján már meg tudjuk csinálni: egy WHILE ciklust helyezünk el a DIAGRAM ablakon, kérünk hozzá egy Shift-regisztert, beállítjuk hozzá a ’0’ kezdeti értéket, majd a cikluson belül elhelyezünk egy CASE-struktúrát, amelynek ’True’ lapján a shift-regiszter korábbi értékét eggyel növelve vezetjük át a jobb oldalra, a shift-regiszter következő értékébe. A shift-regiszternek akkor is meg kell határozni a jövőbeli értékét, ha éppen a ’False’ lapot hajtja majd végre a LabView, ezért ezen a lapon egyszerűen, változtatás nélkül át kell kötnünk a CASE struktúra bemeneti alagútján bejövő számlálóértéket a kimeneti alagútra (és pontosan arra !) A kérdés csak az, hogy mi legyen az a feltétel, amely alapján növeljük a számlálót vagy sem ?

1.1 A megnyomás tényének megállapítása



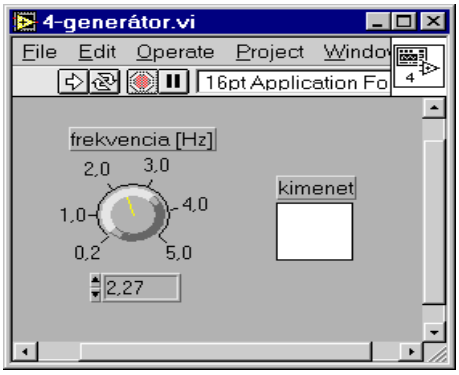
Ha a kapcsoló terminal-ját egyszerűen a CASE struktúra kérdőjelére vezetjük, a program nem a felkapcsolások számát fogja számolgatni, hanem pörgeti a számlálót, amíg a kapcsoló fel van kapcsolva, és leállítja, amíg a kapcsolót alapállapotban tartjuk. A felkapcsolás tényét úgy tudjuk megállapítani, hogy programban megvalósítjuk a következő gondolatmenetet: Ha a kapcsoló az előző ciklusban még ki volt kapcsolva, ÉS most ebben a ciklusban már be van kapcsolva, akkor minden bizonnyal történt egy felkapcsolás. Semelyik más esetben a számlálót nem szabad növelni.

1.2 Számlálás

Az előző ciklusban tapasztalt kapcsolóállást csak úgy tudjuk felhasználni a logikai függvényünkben, ha egy másik shift-regiszterben tároljuk azt. Kérjünk tehát még egy shift-regisztert, és a kapcsoló mindenkori állását vezessük be a jobb oldalon található ’jövőbeli érték’-ébe. Így bármely ciklusban kiolvasható a múltbeli érték – balról – , és könnyen összehasonlítható a jelenlegi értékkel (=a kapcsoló terminal-ja). A függvény elkészítése után még az újonnan felvett shift-regiszter kezdeti értékét kellene beállítanunk. Hogy tényleg csak a felkapcsolásokat számoltassuk a LabView-val, logikai IGAZ-at kell beállítanunk. (A logikai konstans a ’Boolean’ csoportban szintén az alsó sorban található meg, és a működtető kézzel állítható a két állapot között)

2 Frekvencia-generátor elkészítése

A virtuális frekvenciamérő műszerünk egy szintén a LabView-ban elkészített, de a mérő programtól teljesen függetlenül működő frekvencia-generátor kimeneti változóját fogja vizsgálni, és a változás sebességéből a beállított frekvenciához remélhetőleg minél közelebbi értéket mérni.



A pontos négyszögjel generálásához egy olyan végtelen ciklust kell elkészíteni, amely egy változója értékét (célszerűen egy shift-regiszter értékét) bizonyos időközönként az ellenkezőjére fordítja. A logikai IGAZ állapot felel majd meg a magas szintnek, a logikai HAMIS az alacsonynak. Az adó program tehát végtelenül egyszerű: egy db WHILE ciklus, kilépési feltételként állandó IGAZ-at kapcsolva, egy db shift-regiszter, mely múltbeli és jövőbeli értéke között egyetlen inverter (=NOT) helyezkedik el, és egy időzítő, amelyet már használtunk (=WAIT[ms]) – nem ez lesz a legjobb, de egyenlőre ezzel ki tudjuk próbálni a működést. Bemenetére az elérni kívánt

periódusidő felét kell adni ms-ban, hiszen 2 állapotváltás okoz majd egy teljes periódust. A számítógép teljesítményétől függ, hogy milyen frekvenciát tudunk elérni a programunkkal, és milyen pontosan tudjuk majd megmérni, de kezdetben 1 Hz körüli villogást próbáljunk majd beállítani a képernyőn.

2.1 A pontosabb időzítés lehetősége

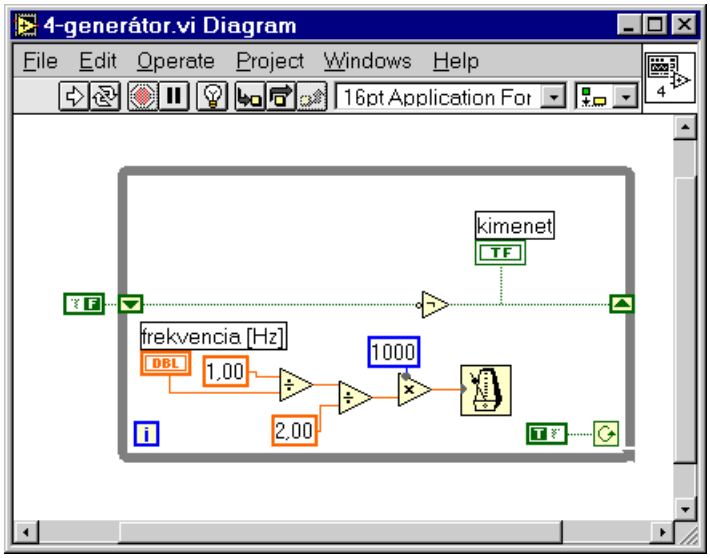
2.1.1 Az eddig megismert késleltető művelet

A hőmérséklet-mérő ciklusunk lassítására használt Wait időzítő valóban nem ad pontos frekvenciájú jelet, és ennek az az oka, hogy a ciklusban az időzítés végrehajtásán kívül sok egyébbel is kell foglalkoznia a programnak. Pl. lámpa átszínezése (ha ilyet egyáltalán elhelyeztünk a képernyőn), ciklusváltozó növelése, kilépési feltétel kiértékelése, shift-regiszter kiolvasása, invertálása majd visszaírása, időzítő bemeneti paraméterének kiértékelése, stb. Csupa olyan dolog, amelynek végrehajtása ha nem is számottevően sok, de mérhető időbe telik, és ezek az idők hozzáadódnak a késleltetésben beállított fél-periódusidőkhöz, a kimeneti „négyszögjelünk" periódusideje nagyobb, frekvenciája pedig kisebb lesz, mint a beállított. A 'WAIT' függvény a rendszeróra számlálóját olvasgatva várakozik annyi impulzusig, amennyit bemeneti paraméterként kap tőlünk.

2.1.2 Egy újabb késleltetési lehetőség

Az ehhez hasonló időzítési feladatok megoldására a Wait melletti 'Wait Until Next Multiple' függvényt célszerű használni: Ez az előzőtől eltérően a paraméter többszörösénél befejezi a várakozást, akárhol is tartson a rendszeridő a függvény meghívásakor. Így tulajdonképpen az egyéb feladatok végrehajtására fordított időt automatikusan levonja a várakozási időből, és óramű-pontosságú kimenő jelet tudunk vele elérni. (Amíg az egyéb feladatokra szánt idő kevesebb, mint a beállított késleltetés, tényleg kiválóan alkalmazható, egyéb esetekben baj van !). Cseréljük le a késleltetést erre !

2.2 Állítható periódusidejű/frekvenciájú négyszögjel-generátor



Eddig egy konstanssal állítottuk be az időzítők késleltetéseit, most tegyük ezt meg úgy, hogy a felhasználó állíthassa be a program indítása előtt. Olyan potenciométert kell tehát elhelyeznünk az előlapon, amelynek a tartományát a program által valószínűleg gond nélkül előállítható és megmérhető frekvenciákra ill. periódusidőkre korlátozzuk. A kísérletezések előtt állítsunk be pl. egy 0,5…10 Hz-es megengedhető tartományt !

2.3 Időmérés a LabView-ban

Szükség lesz arra is, hogy két felfutó él között eltelt időt másodpercben meg tudjuk mérni. Ehhez a harmadik eszközt kell majd használnunk a 'Time & Dialog' csoportban-> a rendszeridőt 'Tick Count'. A számítógép bekapcsolásától eltelt időt tudjuk bármikor kiolvasni belőle, ms-ban. Időmérésre úgy használható, hogy az időmérés végén kiolvasott értékből levonjuk az időmérés kezdetén kiolvasott – és remélhetőleg tárolt – értéket, majd a különbséget 1000-rel osztjuk, hogy másodpercben kapjuk az eltelt időt.

3 Frekvencia mérésének módszerei

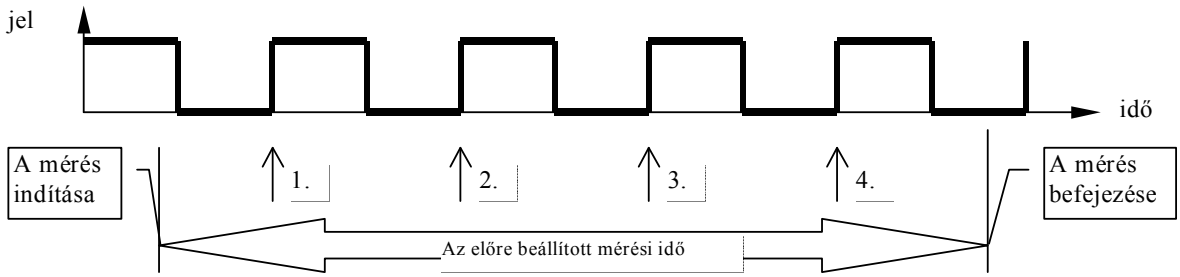
Alapvetően kétfajta módszerrel fogjuk megvalósítani a mérőműszerünket. Mindkettőben előre meghatározott sorrendben kell majd végrehajtanunk bizonyos feladatokat (pl. először kiolvasni a rendszeridőt, majd számolni a felfutó éleket, végül még egyszer kiolvasni a rendszeridőt), és ennek a sorrendnek a rögzítése és feltétlen

betartása érdekében egy eddig nem ismertetett struktúrát fogunk használni: a SEQUENCE-et. Ha egy ilyet elhelyezünk a DIAGRAM ablakon, egy filmhez hasonló széllel jelölt, egyenlőre egyetlen lapot tartalmazó keretet kapunk, mely ebben a formájában még nem használható sorrendi feladatok megoldására. Ha viszont kérünk még néhány lapot (jobb egérgomb a filmcsík szélén, majd 'Add Frame Before/After'), akkor ezeket a LabView megszámozza, és a 0-diktól az utolsóig egymás után végrehajtja a lapon elhelyezett feladatokat.

3.1 Adott ideig történő periódus-számlálással

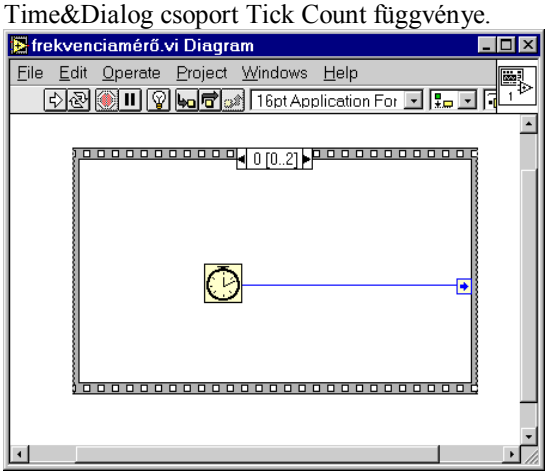
3.1.1 Az alapelv

Előre meghatározott – célszerűen a felhasználó által előre megadott – ideig fut a mérőprogram, és a felfutó éleket számolja. Az idő leteltével az összeszámolt teljes periódusok számát osztja a beállított idővel, és a kapott frekvenciát megjeleníti.



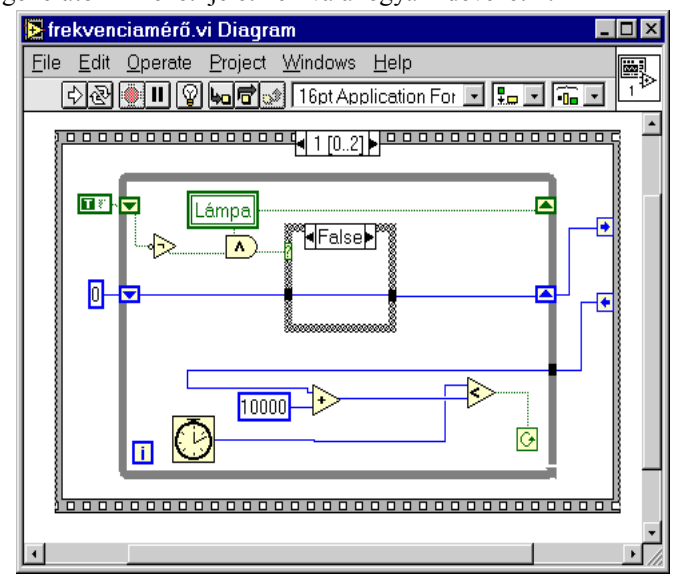
3.1.2 A megvalósítás

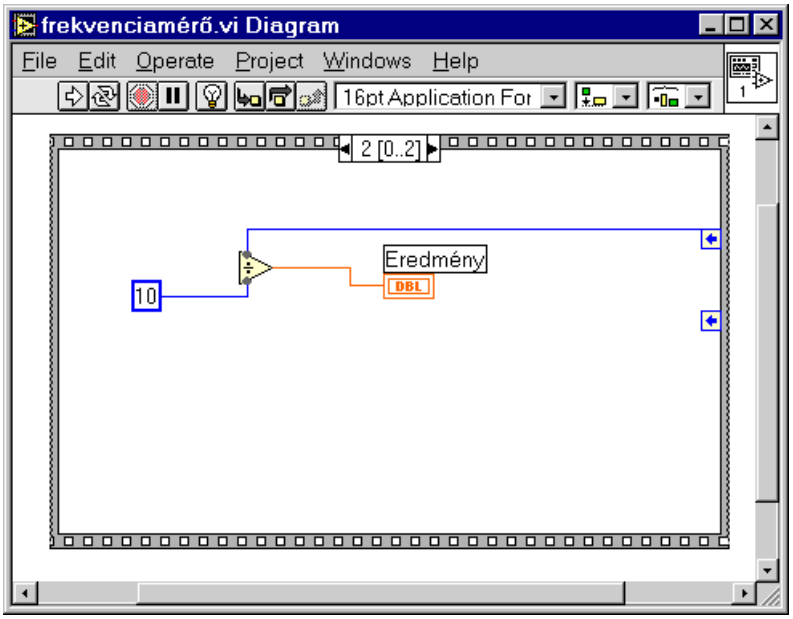
Megvalósításához egy SEQUENCE-t helyezünk el a DIAGRAM ablakon, melynek legelső (=0.!) lapján először kiolvassuk a rendszeridőt, hogy legyen miből meghatároznunk, mennyi idő telt el a mérés közben – Time&Dialog csoport Tick Count függvénye.



Mivel ott egy lámpa villogása jelezte a „vezeték" állapotát, célszerűnek látszik ezt a változót lemásoltatni a LabView-val. Ilyen feladatra a struktúrák csoport 'LOCAL' eleme szolgál. Ha elhelyeztük, jobb gombbal a 'Select Item' menüpontra kattintva kiválaszthatjuk a listáról a lámpánkat – feltéve, hogy annakidején adtunk neki nevet. Nincs már más teendőnk, mint vastag keretesség varázsolni ezt a lokális változót, hogy ő most adatot szolgáltasson nekünk a lámpa állapotának megfelelően (jobb gomb után 'Change To Control'). A WHILE ciklus továbbfutásának feltétele, hogy az előre megadott idő még ne teljen le, tehát a cikluson belül rendszeresen olvasgatni kell a rendszeridőt, és megvizsgálni, hogy kisebb-e, mint az induláskor kiolvasott idő és a mérési idő összege.

Erre a számra viszont a következő lapon lesz majd szükségünk, ezért egy speciálisan ilyen célra való 'Sequence Local' változón keresztül majd át kell vezetnünk a következő lapra. (Jobb gomb a SEQUENCE szélén, majd 'Add Sequence Local'). Vezessük be tehát az induláskor kiolvasott rendszeridőt egy ilyen átmeneti változóba, és folytassuk a munkát a következő lap megtervezésével. A számlálást egy WHILE ciklusban fogjuk végezni, mely nagyon hasonló lesz a bevezető feladatban említett kapcsoló felkapcsolását számolgató programrészhez. Az egyik különbség, hogy itt már szó sincs kapcsolóról. Helyette a mérendő jelet, vagyis a már megvalósított frekvencia-generátor kimeneti jelét kell valahogyan idevezetni.



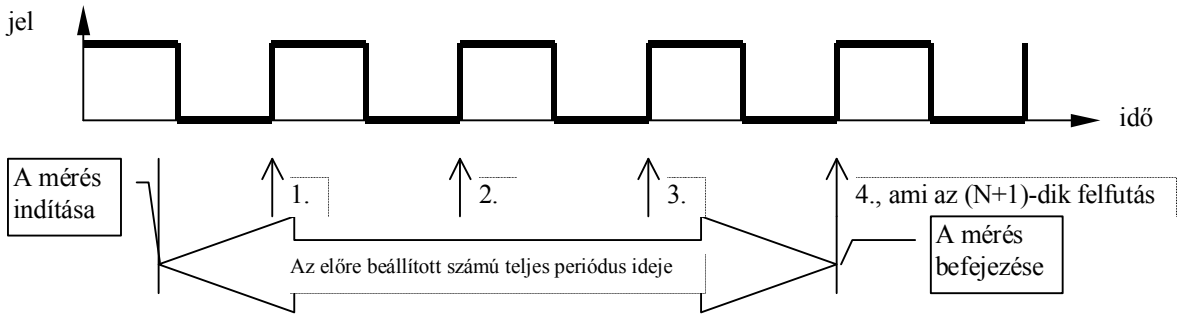


Végül ha letelt a mérési idő, kilépünk a WHILE-ciklusból, és az áttekinthetőség kedvéért egy másik 'Sequence Local'-ban tároljuk, hogy meddig jutottunk a számlálásban, végül a harmadik lapon kiszámoljuk és megjelenítjük a megszámolt felfutó élek és a mérési idő hányadosát.

3.2 Adott számú teljes periódus idejének megmérésével

3.2.1 Az alapelv

Előre meghatározott – célszerűen a felhasználó által előre megadott – számú periódus alatt méri az eltelt időt. A beállított számú periódus után leállítja az időmérést, és a kettő hányadosaként a mért frekvenciát megadja. A következő ábra ezt szemlélteti N=3 esetén:



3.2.2 A megvalósítás

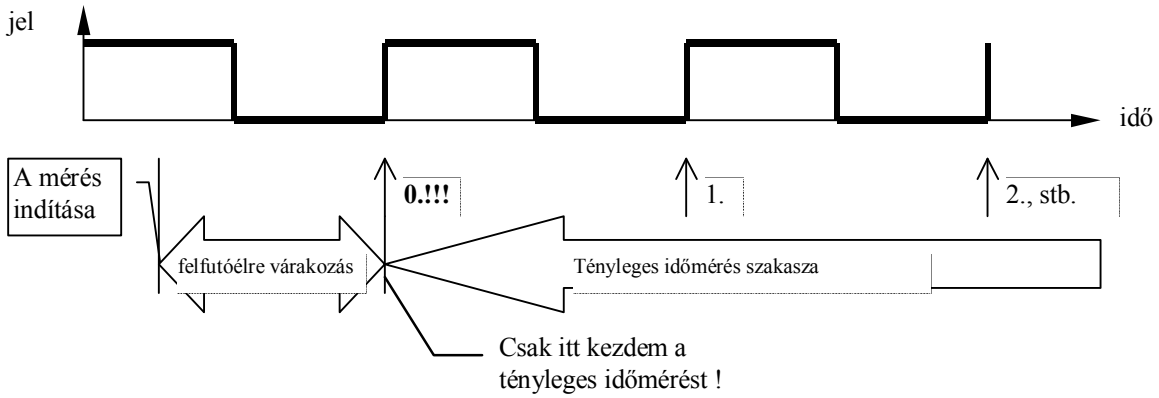
Itt a SEQUENCE első lapján ismét a rendszeridő kiolvasásának kell megtörténnie, a következő lapon szintén egy WHILE ciklus, amely a felfutó éleket számolja, teljesen az előző megoldáshoz hasonlóan. Különbség, hogy most nem kell olvasgatni a rendszeridőt, csak a felfutó éleket számolgatni, és a kilépési feltételt is ehhez kell kapcsolni. Ha ugyanis elértük a szükséges számú felfutó élet, nincs más teendőnk, mint a következő lapon ismét kiolvasni a rendszeridőt, és a számláló, valamint a két idő különbségének hányadosaként meghatározni és megjeleníteni a mért frekvenciát.

4 Kiegészítések a pontosság növelése érdekében

Eddigi mindkét megoldásunk a program indításakor rögtön időmérésbe kezdett, és az első felfutó élnél megindult a ciklusok számolása. Nyilván a program indításának pillanatában egy valóságos helyzetben lehetünk egy periódus elején, közepén vagy akár a legvégén is, és az indítás után tapasztalt első felfutó élig a már ketyegő óra miatt eltelt időt komoly mérési hibákat okozhat, különösen kevés számú periódus vizsgálatakor vagy rövid idejű mérés beállításakor. Hasonló a helyzet a mérés leállításakor is a megvalósítás első variációjánál: Lehet, hogy a felhasználó által megadott mérési idő pontosan egy már megkezdett következő periódus elején, közepén vagy vége felé jár le, és ezt a mérőprogram nem veszi figyelembe.

4.1 Első felfutó él megvárása az időmérés indítása előtt

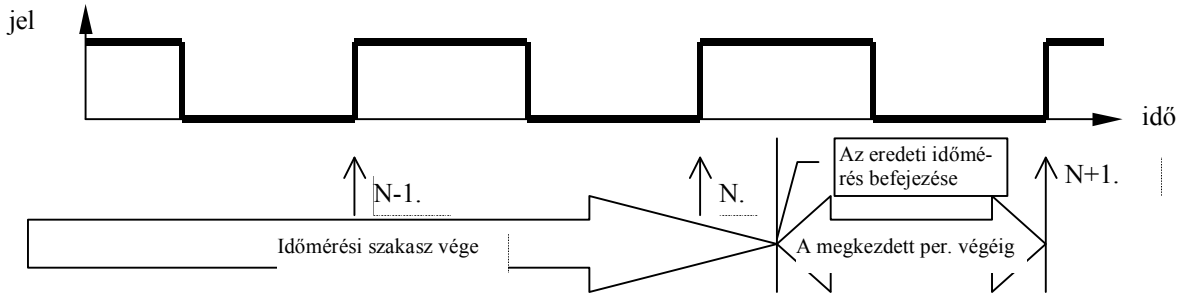
Az első problémára nagyon egyszerű a megoldás:



Mindössze kérnünk kell a SEQUENCE struktúrától még egy lapot, az eddigi legelsőt megelőzően (0. lapra kapcsolva 'Add Frame Before'), és az immáron 0. sorszámot viselő üres lapra egy elfutó élre várakozó ciklust kell elhelyezni. A program többi része változatlan maradt: az időmérés indítása átkerült az 1. lapra, és minden további feladat sorszáma eggyel nőtt.

4.2 Következő felfutó élig időmérés, és ennek beszámítása

A második probléma nyilván nem jelentkezik akkor, ha a második variációban készítjük el a mérőprogramot, hiszen itt pont az utolsó felfutó él detektálását követően állítjuk le az időmérést. Az első esetben viszont a letelt idő után valahogy meg kellene mérni, hogy meddig jutottunk az utolsó megkezdett periódusban.



Ezt pedig úgy tehetjük meg, hogy elindítunk még egy időmérést, és az utolsó, megkezdett periódusból még hátralévő időt (Δt) megmérjük vele (vagyis megvárunk még egy felfutó élet – a megkezdett periódusból hátralevő idővel meghosszabbítjuk a mérést a pontosság növelése érdekében). A megszámolt ciklusok számát eggyel növelve – hiszen eggyel több ciklust vizsgáltunk – elosztjuk a felhasználó által megadott és a ráadásként megmért ráadás idő összegével, hogy minél pontosabb frekvenciát kapjunk.