ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ
ՀԱՄԱԼՍԱՐԱՆ

Տիգրան Աբրահամյան
Հենրիկ Պարսամյան

Գիտափորձի ավտոմատացում LabVIEW միջավայրում

NATIONAL INSTRUMENTS™

ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Աբրահամյան Տ. Ա., Պարսամյան Հ. Ա.

ԳԻՏԱՓՈՐՁԻ ԱՎՏՈՄԱՏԱՑՈՒՄ *LabVIEW* ՄԻՋԱՎԱՅՐՈՒՄ

(ուսումնամեթոդական ձեռնարկ)

ԵՐԵՎԱՆ
ԵՊՀ ՀՐԱՏԱՐԱԿՉՈՒԹՅՈՒՆ
2023

ՀՏԴ 004(07)
ԳՄԴ 32.97գ7
Ա 161

Հրատարակության է երաշխավորել
ԵՊՀ գիտական խորհուրդը:

Աբրահամյան Տ. Ա., Պարսամյան Հ. Ա.
Ա 161 Գիտափորձի ավտոմատացում *LabVIEW* միջավայրում (ուսումնամեթոդական ձեռնարկ) / Տ. Ա. Աբրահամյան, Հ. Ա. Պարսամյան.- Եր.: ԵՊՀ հրատ., 2023, 93 էջ:

«Գիտափորձի ավտոմատացում LabVIEW միջավայրում» ուսումնամեթոդական ձեռնարկը նվիրված է մի շարք լաբորատոր աշխատանքների նկարագրություններին, որոնք Ֆիզիկայի ինստիտուտի ուսումնական պլանով նախատեսված են «Գիտափորձի ավտոմատացում» և «Գիտափորձի ավտոմատացման պրակտիկում» առարկաների լաբորատոր պարապմունքների շրջանակներում:

Ներկայացված են Լաբվյու (*LabVIEW*) ծրագրային միջավայրի գործիքակազմը և լաբորատոր աշխատանքներում կիրառվող հանգույցների հետ աշխատելու առանձնահատկությունները: Լաբորատոր աշխատանքների շարադրման ընթացքում տրվում է դրանց իրականացման համար նախատեսված սարքավորումների նկարագրությունն ու փորձի կատարման ընթացքը:

Ձեռնարկում ներկայացված են տվյալների՝ համակարգիչ ներմուծման և մշակման, համակարգչից դրանց արտածման և սարք-սարքավորումների կառավարման, փորձերի իրականացման գործընթացի ավտոմատացման առանձնահատկությունները *LabVIEW* ծրագրային միջավայրի օգնությամբ:

ՀՏԴ 004(07)
ԳՄԴ 32.97գ7

ISBN 978-5-8084-2613-9
https://doi.org/10.46991/YSUPH/9785808426139

ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

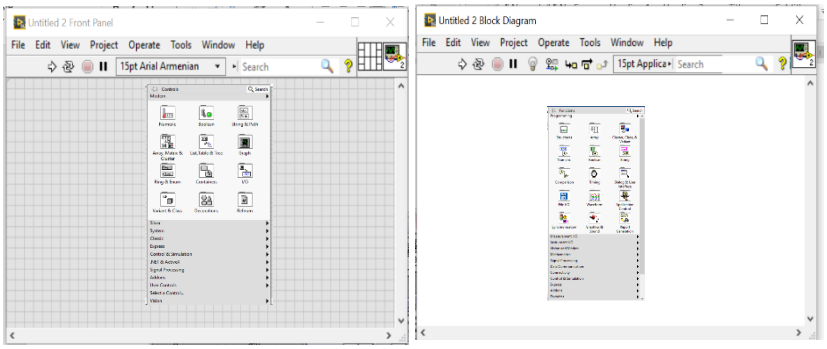
Ներածություն

Գիտության և տեխնոլոգիաների զարգացմանը զուգընթաց որոշ գործընթացների ղեկավարման կամ վերահսկման համար նոր մոտեցումներ անհրաժեշտություն է առաջանում, և մարդուն փոխարինելու են գալիս ռոբոտները, որոնք նույն աշխատանքը կատարում են ավելի արագ, ճշգրիտ և մատչելի: Այս տեսանկյունից ներկայումս ավտոմատացումը լայն կիրառություն է գտել տարբեր ոլորտներում, ինչպիսիք են՝ մեքենաշինությունը, ավիաշինությունը, բժշկությունը և այլն: Բնականաբար, գիտությունը նույնպես անմասն չի մնացել այս գործընթացից, և այսօր ավտոմատացումը հետազոտություններին հաղորդում է նոր որակ՝ հնարավորություն տալով կատարել ավելի ճշգրիտ չափումներ, հեռահար ղեկավարել գիտափորձի ընթացքը, մշակել ստացված արդյունքները և այլն: Գիտափորձերի ավտոմատացում իրականացնելու համար կան բազմաթիվ համակարգչային լեզուներ, դրանցից մեկը *LabVIEW* ծրագիրն է:

LabVIEW-ն համակարգչային ծրագրավորման լեզու է, որը նախագծվել է հանրահայտ *National Instruments* ընկերության կողմից: *LabVIEW* ծրագիրը, որի հապավումը նշանակում է Laboratory Virtual Instrument Engineering Workbench, համարվում է գրաֆիկական ծրագրավորման լեզու, որտեղ տեքստային տողերի փոխարեն օգտագործվում են որոշակի ֆունկցիոնալություն կատարող պատկերներ՝ համապատասխան մուտքային և ելքային հանգույցներով: Վերջին 35 տարում *LabVIEW*-ն օգտագործվում է ինժեներների և գիտնականների կողմից՝ պարզագույն լարման չափումներից մինչև մեծածավալ տվյալների հավաքագրման, սարքերի ղեկավարման և արդյունաբերական ավտոմատացման համար: Սկսած 1986 թվականից, երբ *LabVIEW*-ն առաջին անգամ ներկայացվեց ինժեներական հանրությանը, այն շարունակաբար զարգացել ու բարելավվել է: Այն աշխատում է *Microsoft Windows*, *Macintosh* և *Linux* օպերացիոն համակարգերում:

1. *LabVIEW* ծրագրի հետ աշխատելու սկզբունքներ

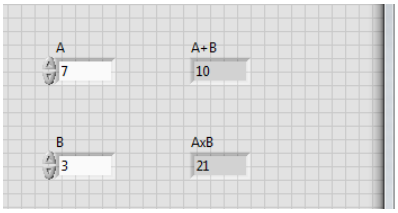
Նկ. 1.1-ում պատկերված է *LabVIEW* ծրագրային միջավայրի գրաֆիկական տեսքը, որը բաղկացած է երկու պատուհաններից՝ *Front Panel*-ից և *Block Diagram*-ից: Գրաֆիկական պատկերների ծրագրավորումը իրականացվում է Block Diagram-ում, իսկ արձագանքը երևում է *Front Panel*-ում: Front Panel-ը՝ դիմային վահանակը, նախատեսված է տարատեսակ տվյալների ներմուծման և արտածման հանգույցների տեղակայման համար: Յուրաքանչյուր պատուհանի վրա սեղմելով մկնիկի աջ կոճակը՝ էկրանին կհայտնվի համապատասխան գործիքակազմի վահանակը:

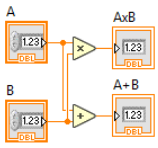


Նկ. 1.1. *LabVIEW* ծրագրային միջավայրի գրաֆիկական տեսքը

Նկ. 1.2-ում պատկերված է արդեն ծրագրավորված միջավայրը, որտեղ գրաֆիկական տարրերը միմյանց միացված են լարերով:

Ծրագիրը մեկ անգամ գործարկելու համար անհրաժեշտ է սեղմել կոճակը  , անընդհատի դեպքում՝  , իսկ դադարի համար՝  :

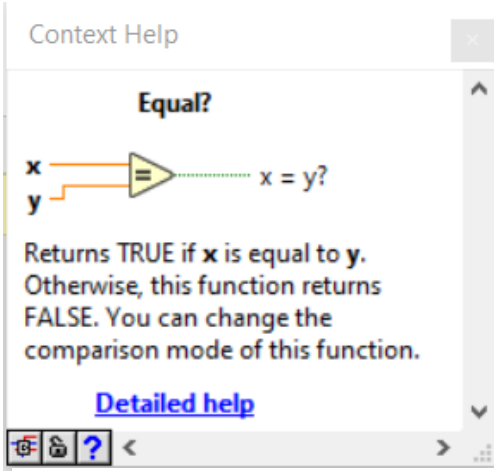




Նկ. 1.2. *LabVIEW* ծրագրավորված միջավայր

LabVIEW ծրագրի գլխավոր առանձնահատկություններից է գրաֆիկական պատկերների միջոցով առավել պարզեցված ծրագրավորման հնարավորությունը: Այսպիսով, դժվար չէ կռահել, որ Նկ. 1.2-ում պատկերված ծրագիրը միաժամանակ հաշվում է a և b թվերի գումարն ու արտադրյալը:

Ծրագիրը հնարավորություն է տալիս տեսնել յուրաքանչյուր ֆունկցիոնալ տարրի մասին բացատրություն, որի համար անհրաժեշտ է նախ *Block Diagram*-ում կամ *Front Panel*-ում հորիզոնական մենյուից ընտրել Help, այնուհետև Show Context Help կամ պարզապես սեղմել «Ctrl+H» ստեղները: Արդյունքում էկրանի անկյունում կհայտնվի Նկ. 1.3-ում պատկերված *Context Help* պատուհանը:

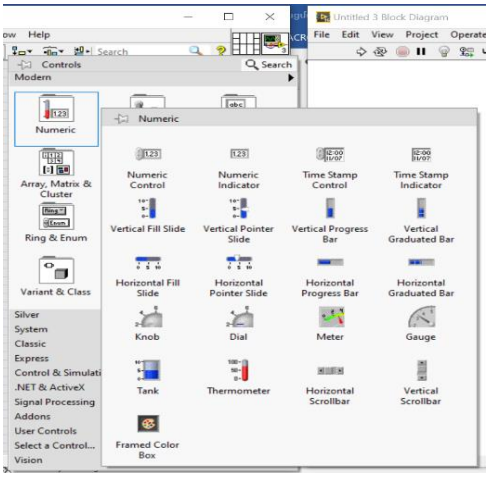


Նկ. 1.3

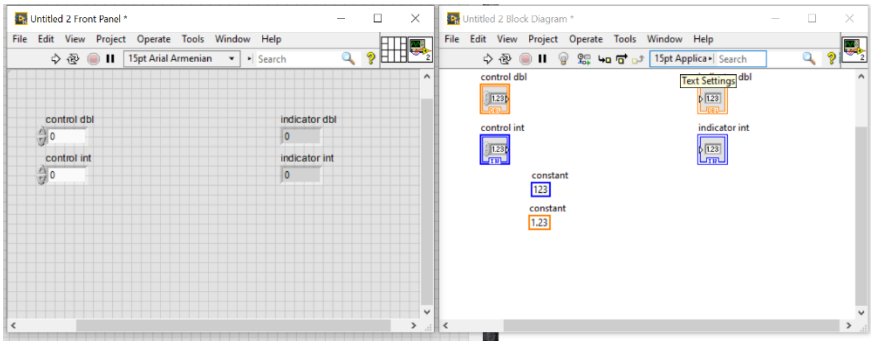
Block Diagram-ում մկնիկի սլաքը պահելով տվյալ ֆունկցիոնալ տարրի վրա՝ *Context Help* պատուհանում կհայտնվի տվյալ տարրի համառոտ բացատրությունը: Այդ պատուհանում սեղմելով Detailed help հղման վրա՝ կհայտնվի տարրի մասին ընդգրկուն ինֆորմացիա և բացատրություն:

2. Constants, Controls և Indicators: Տարբեր տիպի թվերի ներկայացումը *LabVIEW* միջավայրում

Սեղմելով մկնիկի աջ կոճակը ձախ պատուհանի վրա՝ *Numeric* բաժնից ընտրում ենք Control-ները և Indicator-ները: Մենք կսահմանափակվենք այս երկուսով. մնացած գործիքների հետ ծանոթացումը թողնում ենք ընթերցողին (Տե՛ս Նկ. 2.1):



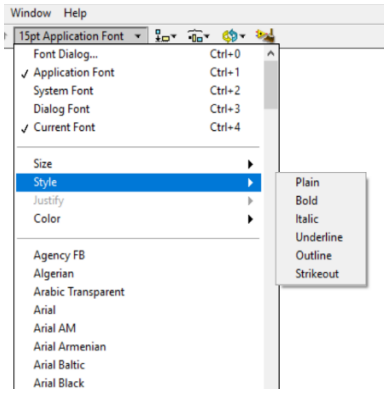
Նկ. 2.1. Numeric բաժնի գործիքները



Նկ. 2.2. *LabVIEW* միջավայրում Constant-ի, Control-ի և Indicator-ի տեսքերը

Նկ. 2.2-ում ցույց է տրված *LabVIEW* միջավայրում Constant-ի, Control-ի և Indicator-ի տեսքերը Front Panel-ում և Block Diagram-ում: Աջ պատու-

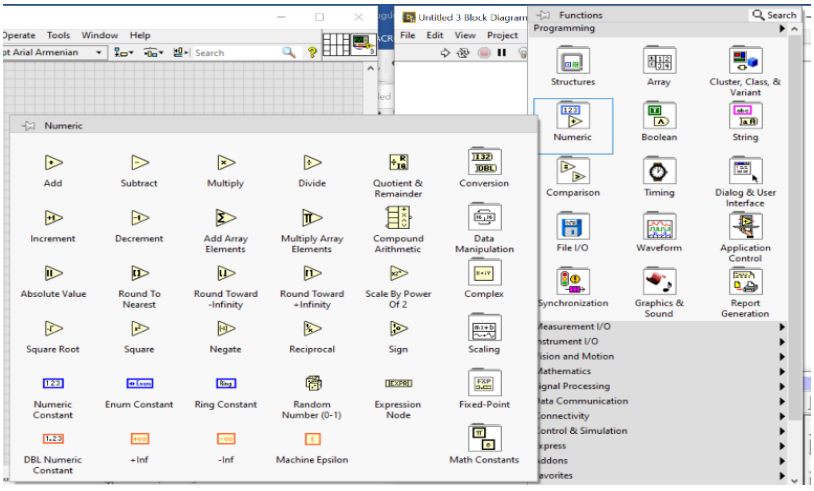
հանում Control-ը Indicator-ից կարելի է տարբերակել արտաքին ընդգծված շրջանակի միջոցով և լարի միացման ուղղությամբ. Control-ը միանում է աջից, իսկ Indicator-ը՝ ձախից: Վերջիններս ձախ պատուհանում տարբերվում են արտաքին տեսքով:



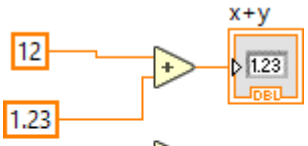
Նկ . 2.3

Այստեղ թվերի ներկայացման տեսակները կարելի է տարբերակել գույներով, օրինակ՝ նարնջագույնը՝ DBL(double), կապույտը՝ INT(integer): Եթե անհրաժեշտ է փոխել թվի բիթային տեսքը (Օրինակ՝ I8, I32, U64, DBL և այլն), ապա սեղմում ենք մկնիկի աջ կոճակը և ընտրում *Representation* բաժինը: Սեղմելով Control-ի և Indicator-ի վրա՝ կարելի է փոփոխել դրանց անուններն ու իրականացնել արտաքին տեսքի ձևափոխություններ (Նկ. 2.3):

Constant-ները կարելի է ընտրել Block Diagram-ի նույն Numeric բաժնից: Ինչպես նկատելի է Նկ. 2.4-ից, այս բաժնում կան բավականին շատ գործիքներ, որոնց արտաքին տեսքից կարելի է կռահել, թե դրանք ինչ գործողություններ են իրականացնում: Նկ. 2.5-ում բերված են դրանցից մի քանիսի բացատրությունները:



Նկ. 2.4



Գումար

Տարբերություն

Բազմապատկում

Երկու թվերի հարաբերության ամբողջ և մնացորդ մասերի հաշվում

Թվի մոդուլի հաշվում

Կլորացում դեպի մոտակա ամբողջ թիվը

0-1պատահական թվերի գեներացում

Քառակուսի արմատ

Թվի քառակուսի բարձրացում

Զանգվածի տարրերի արտադրյալ

Զանգվածի տարրերի գումար

Նկ. 2.5. *Numeric* բաժնի որոշ գործիքների նկարագրությունը

3. Բուլյան տարրեր: Տրամաբանական օպերատորներ: Համեմատման գործիքներ

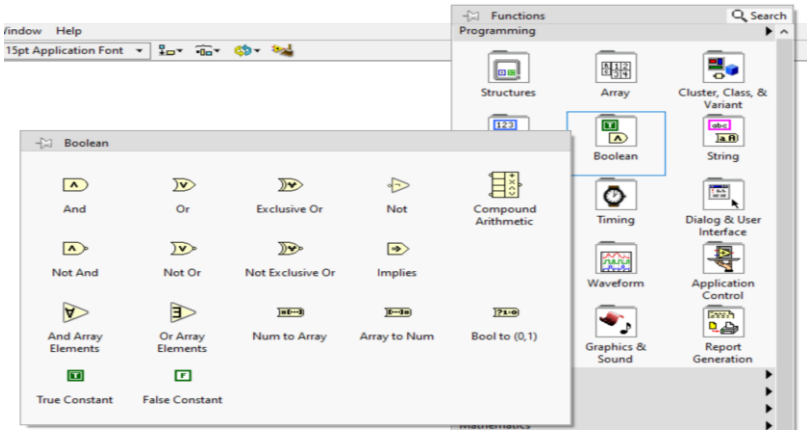
Բուլյան հանրահաշիվը ստեղծվել է 1854թ. Ջորջ Բուլի կողմից, իսկ «Բուլյան հանրահաշիվ» եզրույթն առաջին անգամ առաջարկվել է Շեֆֆերի կողմից 1913թ.: 1930-ական թթ. Քլոուդ Շենոնը, ուսումնասիրելով անջատիչներով կառավարվող էլեկտրական շղթաները, նկատեց նմանություններ բուլյան հանրահաշվի գործողությունների հետ և հանրահաշվորեն նկարագրեց դրանց աշխատանքը տրամաբանական փականների միջոցով:

Բուլյան հանրահաշվում փոփոխականների արժեքները համարվում են «ճշմարիտ» կամ «սխալ» ասույթներ, և սովորաբար դրանք արտահայտվում են 1 ու 0 թվերով: Ի տարբերություն տարրական հանրահաշվի, որտեղ փոփոխականների արժեքները հիմնականում թվեր են, և իրենց հիմնական գործողություններն են գումարումը և բազմապատկումը, Բուլյան հանրահաշվում հիմնական գործողություններ են համարվում կոնյունկցիան՝ «և» տրամաբանական շաղկապով, որը արտահայտվում է ∧ նշանով, դիզյունկցիան՝ «կամ» շաղկապի կիրառմամբ, որը արտահայտվում է ∨ նշանով, և հերքումը, որը ժխտում է դատողությունը՝ ցույց տալով, որ հակառակն է ճիշտ, և նշանակվում է ¬ նշանով:

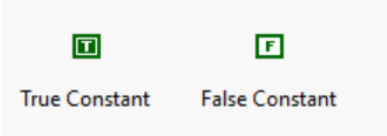
Բուլյան հանրահաշիվը հիմնային դեր ունի թվային էլեկտրոնիկայի զարգացման մեջ:

Այժմ *LabVIEW* միջավայրում ծանոթանանք հետևյալ գործիքներին.

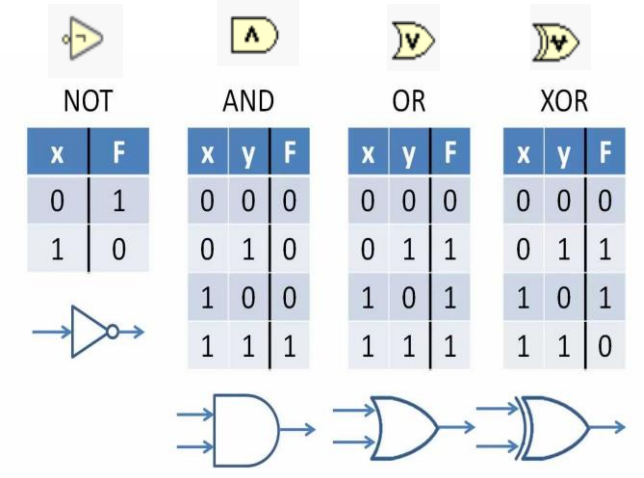
ա) *Block Diagram*-ի *Boolean* բաժնի գործիքներ (Նկ. 3.1):



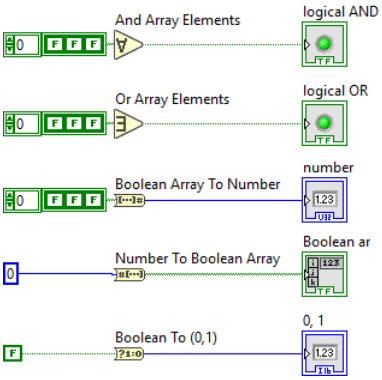
Նկ. 3.1 *Block Diagram*-ի *Boolean* պատուհանի գործիքները



Բուլյան հաստատուններ



Նկ. 3.2. *LabVIEW* միջավայրում օպերատորների տեսքերը և աշխատանքները բնութագրող իսկության աղյուսակները



True, երբ բուլյան զանգվածի բոլոր էլեմենտները True են, մնացած դեպքերում՝ False:

False, երբ բուլյան զանգվածի բոլոր էլեմենտները False են, մնացած դեպքերում՝ True:

Բուլյան զանգվածի ձևափոխումը թվային տեսքի

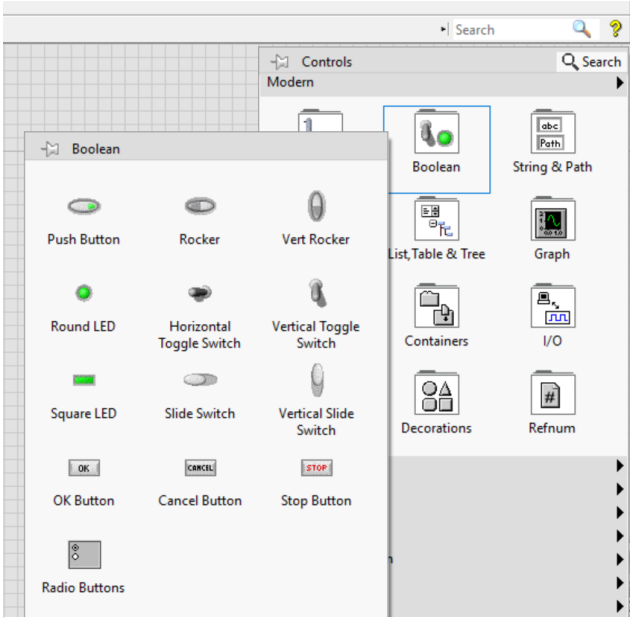
Թվային տեքստի ձևափոխումը բուլյան զանգվածի

Բուլյան արժեքի ձևափոխումը թվային 0 կամ 1-ի

Նկ . 3.3. Բուլյան օպերատորների աշխատանքի նկարագրությունը

բ) Front Panel-ի Boolean բաժնի գործիքները (Նկ. 3.4):

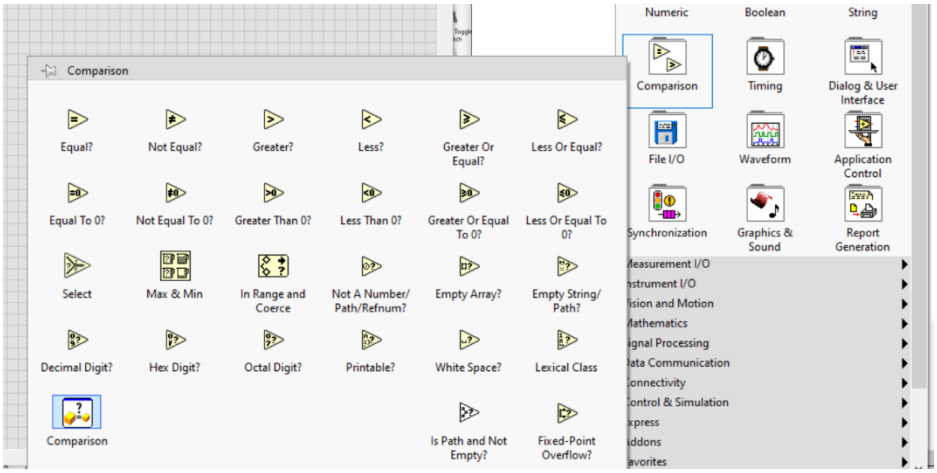
Այստեղ հիմնականում տարբեր տեսքի *Control* կոճակներ են՝ տարբեր մեխանիկական միացումներով, իսկ կլոր և ուղղանկյունաձև *LED* լույսերը *Indicator*-ներ են: Նշենք, որ դրանց արտաքին տեսքերը նույնպես կարելի է ենթարկել ձևափոխությունների:



Նկ. 3.4. *Front Panel*-ի *Boolean* բաժնի գործիքները

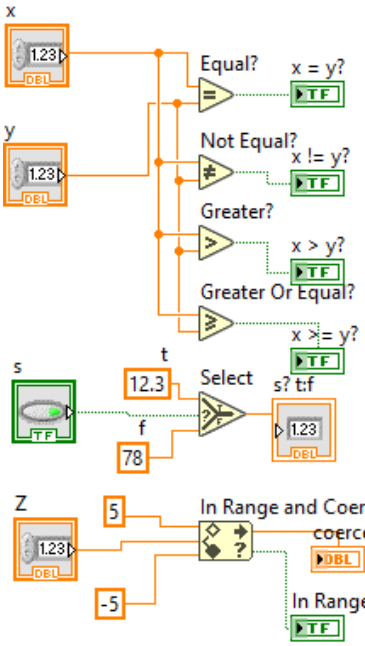
Համեմատության գործիքներ

Մկնիկի աջ կոճակի միջոցով *Block Diagram*-ում ընտրելով *Comparison* բաժինը՝ կտեսնենք բազմաթիվ գործիքներ (Տե՛ս Նկ. 3.5):



Նկ. 3.5. *Block Diagram*-ի Comparison բաժնի գործիքները

Անդրադառնանք դրանցից մի քանիսին:



Եթե x = y, ապա՝ True:

Եթե x ≠ y, ապա՝ True:

Եթե x > y, ապա՝ True:

Եթե x >= y, ապա՝ True:

Եթե s=True, ապա՝ t, իսկ հակառակ դեպքում՝ f:

Եթե Z>5, ապա՝ 5, Z<-5, ապա՝ -5:

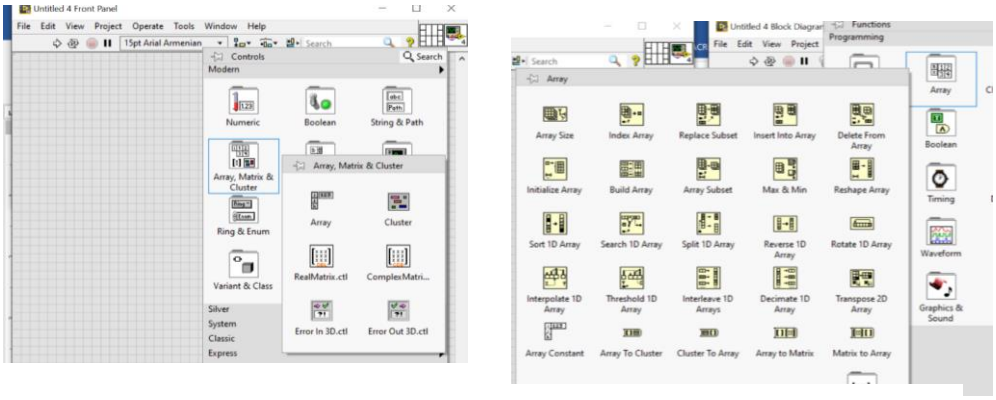
Եթե Z-ը՝ [-5:5], ապա՝ True:

Նկ. 3.6. *Comparison* բաժնի որոշ գործիքների նկարագրությունը

4. Զանգվածներ (Arrays): Կլաստերներ (Clusters): Սթրինգներ (Strings)

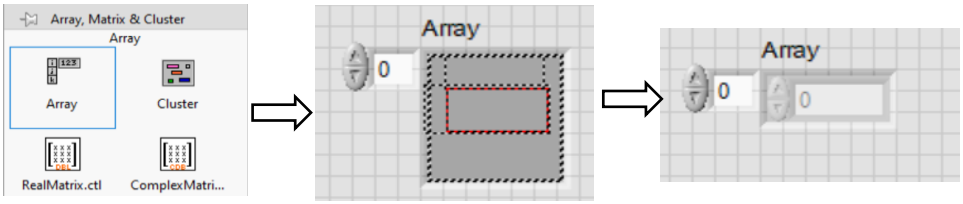
Զանգվածներ

Զանգվածները միևնույն տիպի տարրերից կազմված շարքեր են, որոնք ծրագրավորման մեջ կարևոր և հաճախակի օգտագործվող մասերից են: Հիմնականում լայն կիրառություն են գտել 1 և 2 չափանի (1D, 2D) զանգվածները: Նկ. 4.1-ում պատկերված է *Front Panel*-ի և *Block Diagram*-ի զանգվածների բաժինների գործիքակազմը *LabVIEW* միջավայրում:



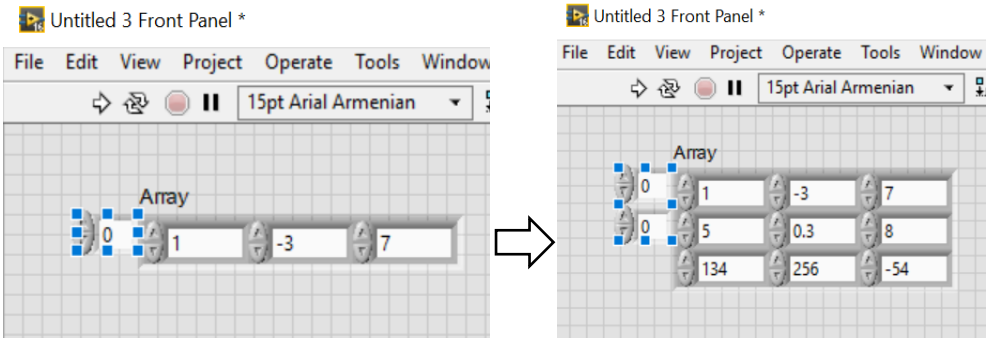
Նկ. 4.1. Front Panel-ում և Block Diagram-ում զանգվածների գործիքակազմը

Այժմ Front Panel-ում ստեղծենք 1D ղեկավարվող թվային զանգված (Numeric Control Array): Մկնիկի օգնությամբ անհրաժեշտ է ընտրել Array, Matrix & Cluster բաժինները և Նկ. 4.2-ում պատկերված հերթականությամբ կառուցել այն:



Նկ. 4.2. Numeric Control Array-ի ստեղծման հերթականությունը
(Front Panel-ում ստեղծած Array-ի մեջ տեղադրում ենք Numeric Control-ը):

2D և ավել չափանի զանգվածներ ստանալու համար անհրաժեշտ է ձգել զանգվածի ինդեքսը (index [0:N]) դեպի ներքև (Նկ. 4.3): Կարելի է անել նաև զանգվածի արտաքին տեսքի ձևափոխություններ (Տե՛ս Նկ. 2.3):



Նկ. 4.3. 1D զանգվածից 2D-ի ձևափոխություն

Ստորև ներկայացված են *Block Diagram*-ում *Array* բաժնի հաճախ օգտագործվող մի քանի ֆունկցիաներ (Նկ. 4.4):

Ցույց է տալիս զանգվածի տարրերի քանակը:

Զանգվածից տարանջատում է համապատասխան ինդեքսով էլեմենտը:

Զանգվածում ավելացնում է էլեմենտ:

Զանգվածից էլեմենտի հեռացում

Զանգվածի կառուցում

Զանգվածի էլեմենտի փոխատեղում

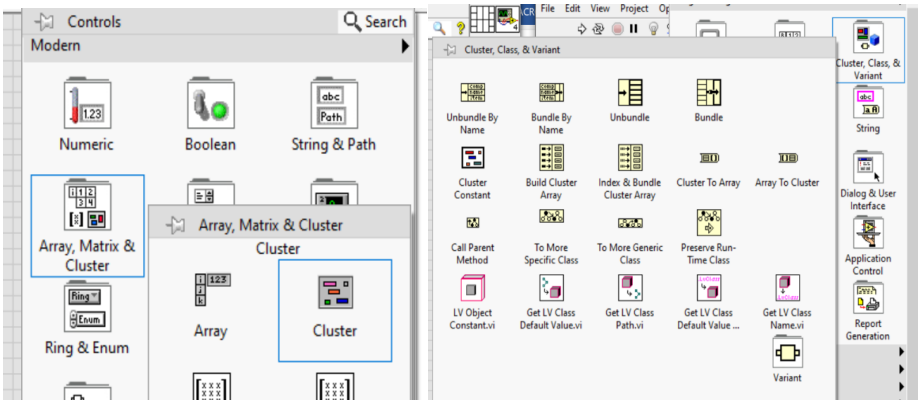
Զանգվածում առավելագույն և նվազագույն արժեքների որոշում՝ իրենց համապատասխան ինդեքսներով

Նկ. 4.4. Array բաժնի մի քանի ֆունկցիաների նկարագրությունները

Հետագայում՝ ցիկլերի ներկայացման ժամանակ, զանգվածներին կրկին կանդրադառնանք:

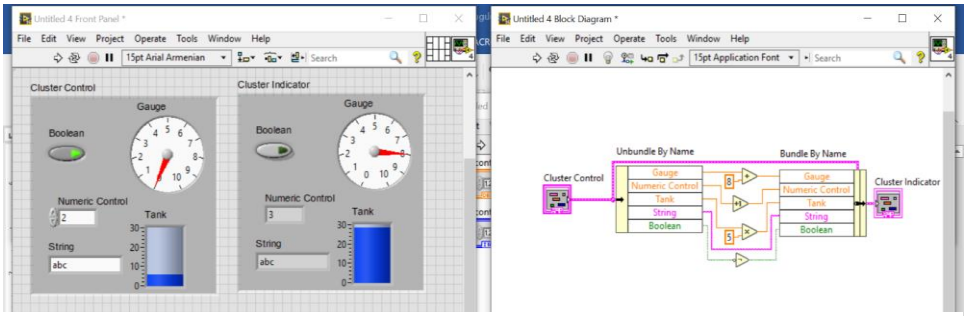
Կլաստերներ

Կլաստերները (Clusters) տարբեր տիպի էլեմենտներից կազմված համակարգներ են: Նկատելի է, որ *Front Panel*-ում կա միայն մեկ ֆունկցիա, իսկ *Block Diagram*-ում դրանք շատ են (Նկ. 4.5):



Նկ. 4.5. Front Panel-ում (ձախ) և Block Diagram-ում (աջ) կլաստերների գործիքները

Ստեղծենք տարբեր տիպի տվյալներով կլաստերներ և դրանց հետ կատարենք որոշակի գործողություններ: Նկ. 4.6-ի ձախ մասում պատկերված են *Cluster Control*-ը և *Cluster Indicator*-ը, իսկ աջ մասում պատկերված է, թե ինչպես են տվյալները դուրս բերվում կլաստերից: Այնուհետև կատարվում են որոշակի փոփոխություններ, և կրկին հետ ներմուծվում կլաստեր:

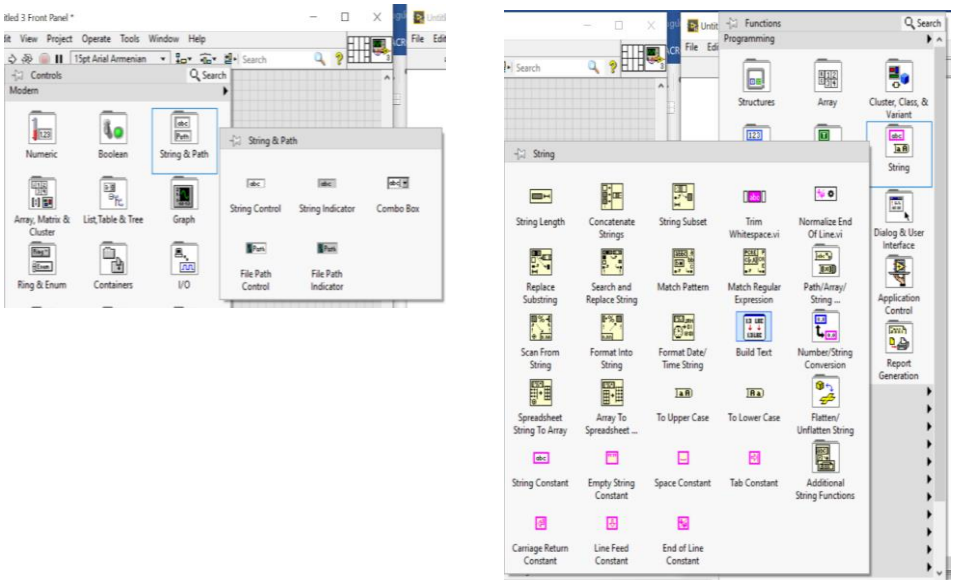


Նկ. 4.6. Կլաստերներում տվյալների փոփոխության իրականացումը

Ձախ կողմի դիմային վահանակի (Front Panel) Cluster Indicator–ի վրա երևում է, թե ինչ փոփոխություններ են իրականացվել:

Տողային տիպ (Strings)

Սթրինգը (String) ցուցադրվող կամ չցուցադրվող ASCII (American standard code for information interchange) նիշերի (Character) հաջորդականություն է: *Front Panel*-ում տողային տիպի փոփոխականները գտնվում են String&Path, իսկ Block Diagram-ում՝ String ենթաբաժնում (Նկ. 4.7):



Նկ. 4.7. Սթրինգների գործիքները Front Panel-ում և Block Diagram-ում

Առաջինում դրանք երկուսն են՝ String Control և String Indicator, իսկ երկրորդում գործիքակազմը բավական հարուստ է: Անդրադառնանք դրանցից մի քանիսին (Տե՛ս Նկ. 4.8):

Հաստատուն սթրինգ

Դատարկ հաստատուն սթրինգ

Սթրինգի երկարության որոշում

Սթրինգների միավորում

Ենթասթրինգի դուրս բերում սթրինգից

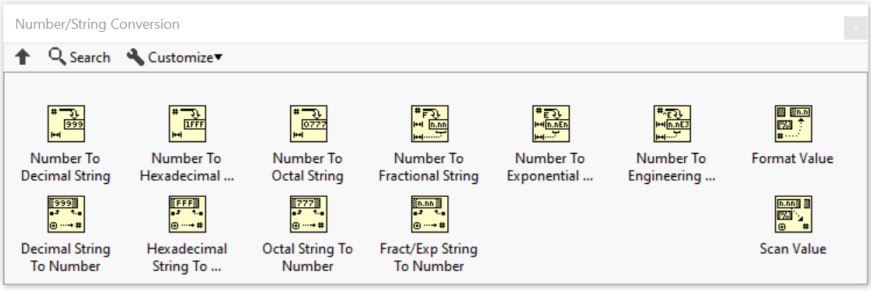
Սթրինգի փնտրում և փոխարինում

Նմուշի համընկնում

Ջննում սթրինգում

Նկ. 4.8. Սթրինգների բաժնի որոշ ֆունկցիաների նկարագրությունները

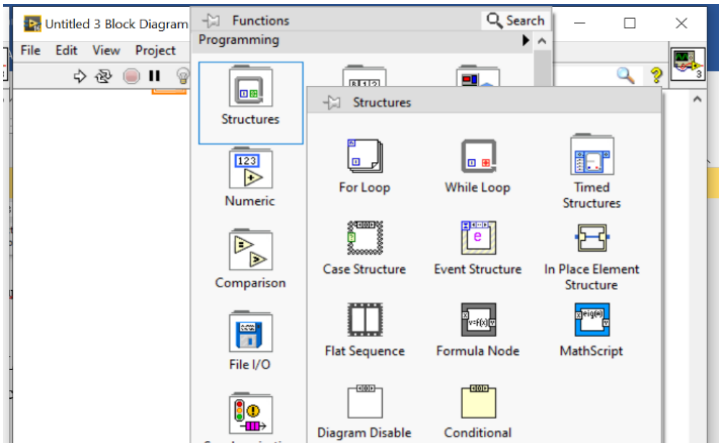
Տողային տիպի թվերով հանրահաշվական գործողություններ կատարելու համար անհրաժեշտ է տողային տիպը ձևափոխել թվային տիպի տվյալների (2-ական, 8-ական, 16-ական և այլն): Այսպիսի գործողությունների իրականացման համար անհրաժեշտ է օգտագործել *Number/String Conversion* բաժնի ֆունկցիաները (Տե՛ս Նկ. 4.9):



Նկ. 4.9. Սթրինգը-թվերի և թվերը-ստրինգների ձևափոխման գործիքները

5. Ցիկլեր (For Loop, While Loop, Shift Reg), պայմանի կառուցվածք (Case Structure), Flat Sequence, Local և Global Variable

Եթե անհրաժեշտություն է առաջում տրված գործողությունը մեկից ավելի անգամ կատարելու, ապա օգտագործում են ցիկլեր: Ցիկլերն առհասարակ ծրագրավորման մեջ շատ կարևոր դեր ունեն: Ցիկլերը գտնվում են *Block Diagram*-ի Structures բաժնում (Տե՛ս Նկ.5.1):

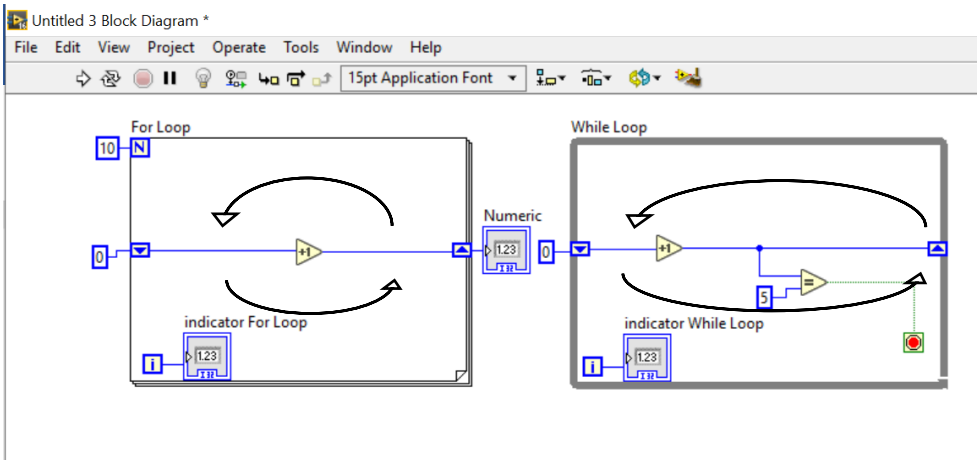


Նկ. 5.1. Ցիկլեր՝ Block Diagram-ի Structures բաժնում For Loop և While Loop

While Loop-ը աշխատում է այնքան ժամանակ, քանի դեռ չի իրականացել ցիկլի դադարեցման պայմանը (True կամ False): For Loop-ի դեպքում ցիկլերի քանակը սահմանվում է նախապես: Այն ունի նաև պայմանի ավելացման հնարավորություն, որը թույլ է տալիս ժամանակի ցանկացած պահին դադարեցնել ցիկլի աշխատանքը:

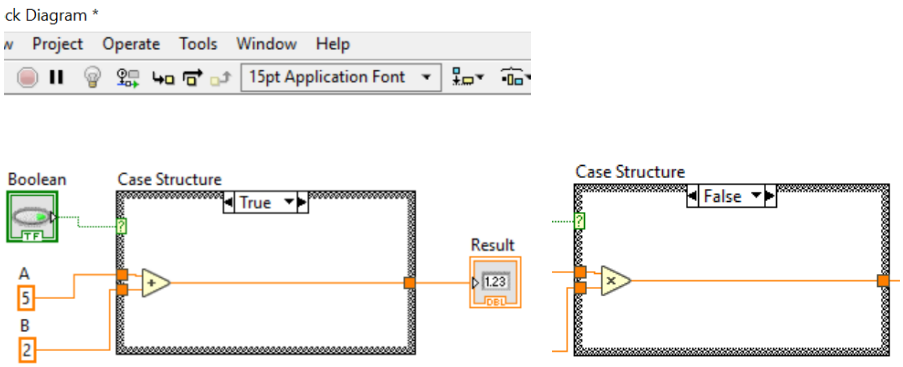
Նկ. 5.2-ում պատկերված են Block Diagram-ում For Loop-ի և While Loop-ի տեսքերը՝ իրենց *Shift Register*-ներով: *For Loop* ցիկլի գործողությունների քանակը 10 է, իսկ *While Loop*-ինը՝ 5: *Indicator For Loop*-ը և Indicator While Loop-ը ցույց են տալիս ընթացիկ ցիկլիկ գործողության հաշիվը: Shift Register-ը ավելացնելու համար ցիկլի ուղղահայաց կողմերից մեկի վրա անհրաժեշտ է սեղմել մկնիկի աջ կոճակը և ընտրել Add Shift Register ֆունկցիան: Սլաքներով ներկայացված է Shift Register-ի աշխատանքի սկզբունքը: Այն հիշողության բջիջ է, որը հնարավորություն

է տալիս i-րդ քայլում ստացված որևէ գործողության արդյունք օգտագործել i+1 քայլում:



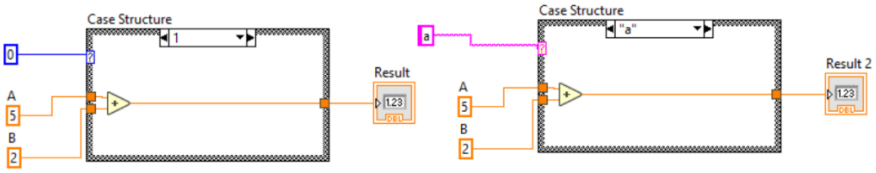
Նկ. 5.2. Block Diagram-ում For Loop-ի և While Loop-ի աշխատանքի սկզբունքը

Case Structure-ը տառային ծրագրավորման մեջ համարժեք է ԵԹԵ(if)-ին: Այստեղ ունենք երկու էջ՝ True և False: Առաջինում կատարվում է A և B հաստատուն թվերի գումարման գործողություն, իսկ երկրորդում՝ բազմապատկման: Boolean կոճակի օգնությամբ ընտրում ենք համապատասխան էջը կամ գործողությունը (Տե՛ս Նկ. 5.3):



Նկ. 5.3

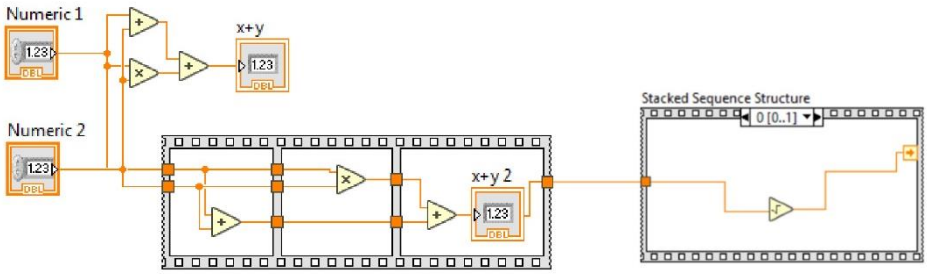
Եթե անհրաժեշտ է էջերի քանակը դարձնել երկուսից ավել, ապա բուլյան պայմանը փոխարինվում է թվերի կամ սթրինգների (Նկ 5.4)։ Էջերի անունները թվերի դեպքում ընտրվում է նույնը, իսկ սթրինգների դեպքում՝ չակերտների մեջ։



Նկ. 5.4

Flat Sequence

Գործողությունների/տվյալների հոսքի հերթականության կառավարման համար կարելի է օգտագործել Flat Sequence Structure գործիքը։ Նկ. 5.5-ում վերջինիս բերված գրաֆիկական տեսքն է. այն նման է ֆոտոժապավենի։ Ձախ կողմում կադրերը ներկայացված են շարքի տեսքով, իսկ աջ մասում այն համարակալված է։ Աշխատանքի սկզբունքը հետևյալն է. յուրաքանչյուր կադրում գործողության ավարտից հետո անցում է կատարվում հաջորդ կադրին և այսպես շարունակ, մինչև վերջին կադրը։ Կադրերը ավելացնելու կամ հեռացնելու համար կողային եզրերից որևէ մեկի վրա մկնիկի աջ կոճակի օգնությամբ ընտրում ենք համապատասխան գործողությունը։



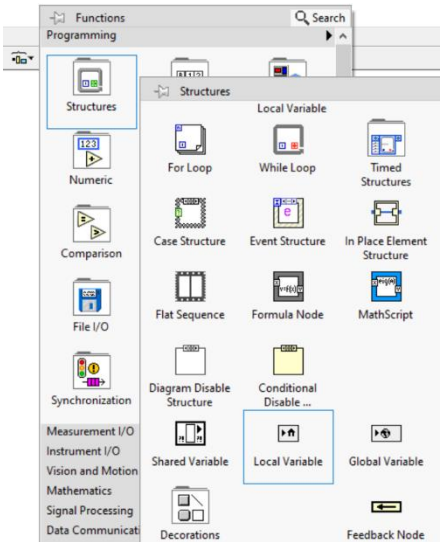
Նկ. 5.5

Նկ. 5.5-ի ձախ կողմի Flat Sequence-ի կադրից կադր տվյալները փոխանցվում են լարի միջոցով, իսկ աջ կողմի համարակալված ժապավենում դա կատարվում է Add Local Sequence-ի միջոցով, որն ընտրվում է

մկնիկի աջ կոճակի օգնությամբ: Կիրառություններին առավել մանրամասն նախատեսվում է ծանոթանալ լաբորատոր դասընթացում:

Տեղային և գլոբալ փոփոխական (Local և GlobalVariable)

Local Variable



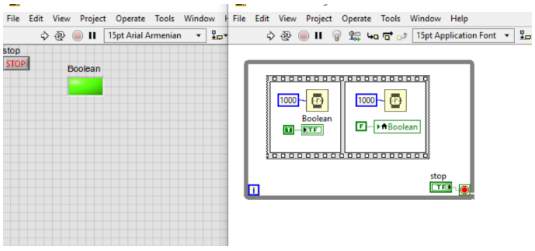
Նկ. 5.6

Local Variable-ը (տեղային փոփոխական) գտնվում է Block Diagram-ի Structures բաժնում (Նկ. 5.6): Ընտրելով այն՝ անհրաժեշտ է տեղադրել ընթացիկ VI-ում և մկնիկի աջ կոճակի օգնությամբ Select Item բաժնից ընտրել համապատասխան փոփոխականը: Մյուս տարբերակով՝ մկնիկի ձախ կոճակով տեղային փոփոխականի վրա սեղմելու արդյունքում կհայտնվի ընթացիկ ֆայլում գտնվող բոլոր փոփոխականների ցանկը, որտեղից կարելի է ընտրել համապատասխան փոփոխականը:

Կարելի է նաև ցանկացած փոփոխականի վրա սեղմել մկնիկի աջ կոճակը և ընտրելով Create/Local Variable ֆունկցիան՝ ստեղծել տվյալ փոփոխականի տեղային փոփոխականը, որը կգործի միայն ընթացիկ ֆայլում:

Դիտարկենք հետևյալ օրինակը:

Նկ. 5.7-ի ծրագրում Flat Sequence-ի առաջին կադրում սահմանված է Boolean փոփոխականը, որը անհրաժեշտ է օգտագործել նաև երկրորդ կադրում: Այս դեպքում կարելի է օգտագործել Boolean փոփոխականի Local Variable-ը: Առաջին կադրում լույսը կմիանա 1վ տևողությամբ, իսկ երկրորդում նույնքան ժամանակով կանջատվի: Արդյունքում կստացվի լույսի թարթում՝ 2վ պարբերությամբ:

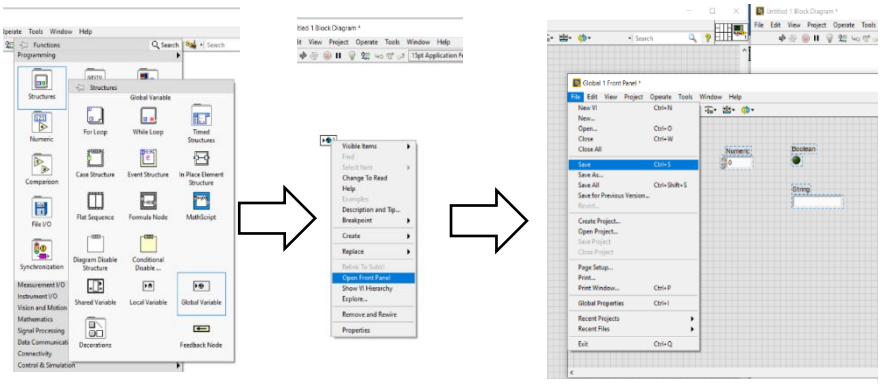


Նկ. 5.7

Global Variable

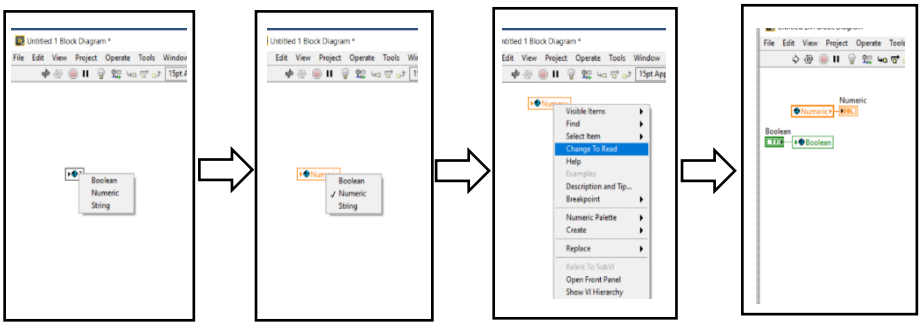
Ենթադրենք՝ ունենք երկու VI ֆայլեր (Untitled 1 և Untitled 2), և անհրաժեշտ է տվյալները տեղափոխել մեկից մյուսը: Դա կարելի է իրականացնել Global Variable-ի միջոցով:

Block Diagram-ի Structure բաժնից պետք է ընտրել Global Variable-ը և Open Front Panel-ի միջոցով բացել Global Variable-ի Front Panel-ը: Այնուհետև ներմուծվում են անհրաժեշտ փոփոխականները և պահպանվում Global 1 անվամբ ֆայլում (Նկ. 5.8), որից հետո Untitled 1 ֆայլում ընտրվում է երեք փոփոխականներից մեկը և սահմանվում՝ Change To Read կամ Write:

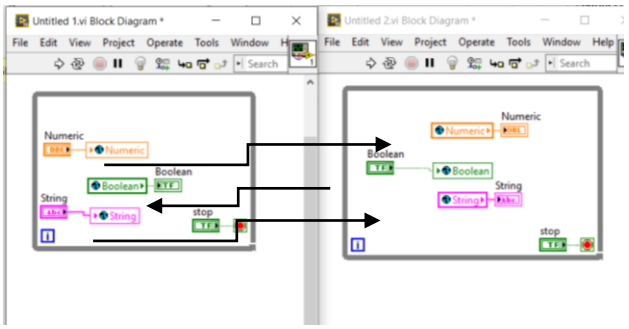


Նկ. 5.8

Այլ փոփոխականների օգտագործման համար կարելի է Global Variable-ը Copy և Paste անել (Նկ 5.9):



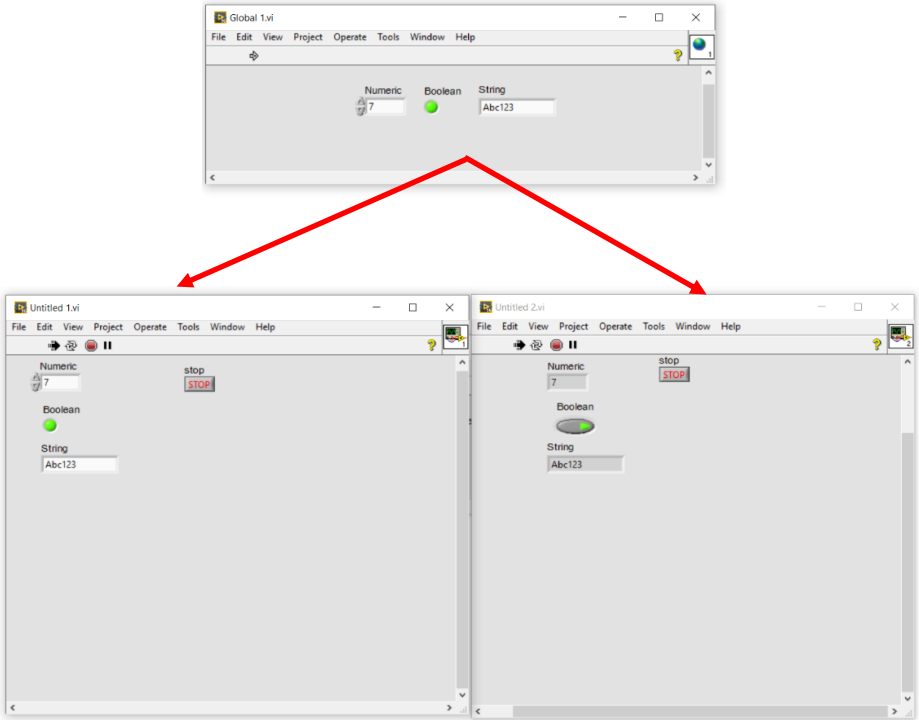
Նկ. 5.9



Նկ. 5.10

Ներմուծելով երեք տիպի փոփոխականների Global Variable-ները և տեղադրելով դրանք ցիկլերի մեջ՝ վերջնական ծրագրերի Block Diagram-ները կստանան Նկ. 5.10-ում պատկերված տեսքերը: Սլաքներով ցույց են տրված փոփոխականների տեղափոխման ուղղությունները, իսկ Նկ. 5.11-ում ցույց են տրված Untitled 1.vi, Untitled 2.vi և Global 1.vi ֆայլերի Front Panel-ների աշխատանքային տեսքերը:

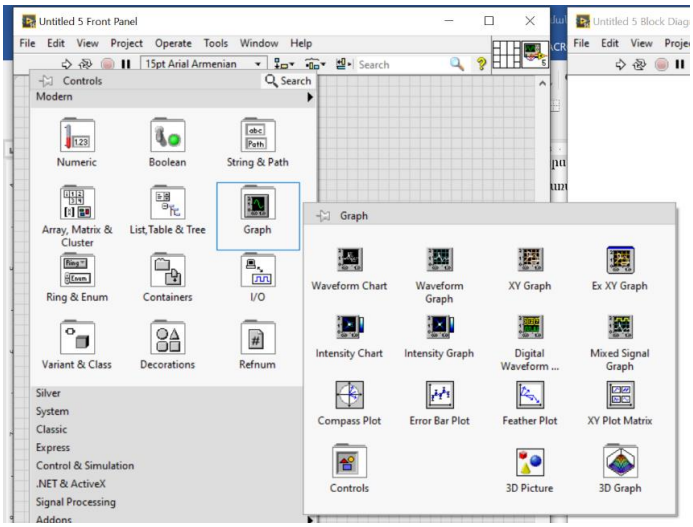
Այսպիսով, Global 1.vi-ն այս երկու ֆայլերի համար միջանկյալ գործիք (կամուրջ) է, որի միջոցով տվյալները մի ծրագրից փոխանցվում են մյուսին:



Նկ. 5.11

6. Գրաֆիկական արտապատկերման հանգույցներ (Waveform Graph, Waveform Chart, XY Graph)

LabVIEW-ում գրաֆիկներ կառուցելու համար անհրաժեշտ է Front Panel-ում Graph բաժնից ընտրել գրաֆիկների արտապատկերման համապատասխան տիպերը, որոնք ցույց են տրված Նկ. 6.1-ում:



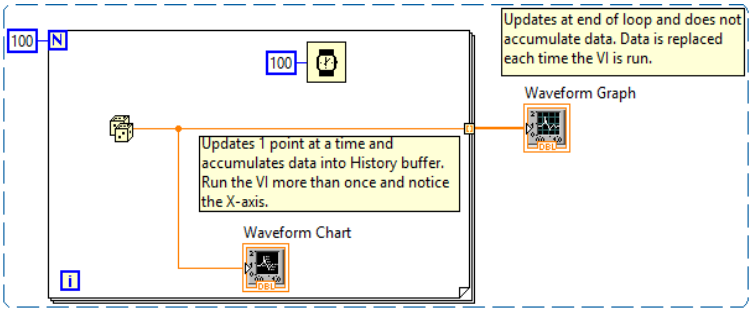
Նկ. 6.1

Տվյալների գրաֆիկական պատկերման հիմնական միջոցներն են Waveform Graph, Waveform Chart, XY Graph, Intensity Graph և Intensity Chart ֆունկցիաները:

Waveform Graph-ը հիմնականում օգտագործվում է հաստատուն արագությամբ (կետերի քանակ միավոր ժամանակում) ստացվող տվյալների գրաֆիկական պատկերման համար: Մուտքային տվյալի տիպը կարող է լինել զանգված, waveform (կլաստեր, որը թույլ է տալիս պահել պարբերական ազդանշաններ և կազմված է ազդանշանի սկզբնական ժամանակից՝ t_0, երկու հարևան կետերի միջև ժամանակային dt քայլից՝ արտահայտված վայրկյաններով, և տվյալների զանգվածից՝ Y), ինչպես նաև դինամիկ տիպի տվյալներ (ազդանշանների միաչափ ու երկչափ զանգվածներ): Բոլոր մուտքային տվյալները Waveform Graph-ի վրա արտապատկերվում են միաժամանակ: Waveform Graph-ի համար որպես մուտքային տվյալ հնարավոր է օգտագործել միայն n-չափանի զանգված-

ներ: Երբ մուտքին միացված է զանգված, ապա լռելյայն (by default) ընդունվում է, որ ժամանակի սկզբնապահը՝ $t_0 = 0$, իսկ ժամանակի քայլը՝ dt=1վ: Վերջիններս հնարավոր է փոխել՝ օգտագործելով համապատասխան property node-երը:Waveform Chart-ը բուֆերում պահպանում է որոշակի ֆիքսված քանակով տվյալներ և ցույց է տալիս դրանք: Երբ բուֆերում պահվող կետերի քանակը հավասարվում է առավելագույնին, ավելի վաղ ստացված կետերը փոխարինվում են իրենց հարևան նոր տվյալներով, իսկ ազատված մասում տեղավորում են նոր մուտքային տվյալները և կցվում նախորդներին: Որպես մուտքային տվյալներ՝ կարող են լինել ինչպես թվեր, այնպես էլ զանգվածներ: Տարբեր տվյալներով շարքեր միևնույն Waveform Chart-ի վրա կառուցելու համար անհրաժեշտ է դրանք միավորել կլաստերի կամ երկչափ զանգվածի մեջ, որոնք կպարունակեն waveform կամ դինամիկ տիպի տվյալ՝ յուրաքանչյուր առանձին տվյալների շարքի համար:

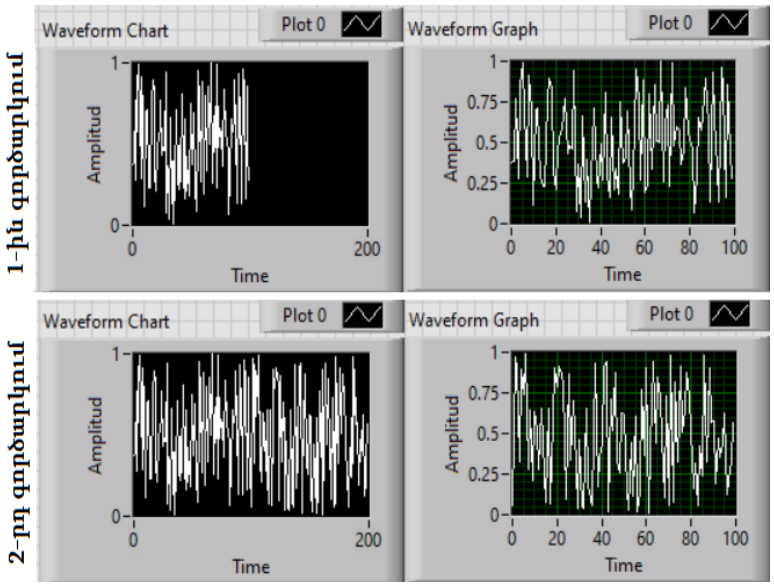
Waveform Graph-ի և Waveform Chart-ի տարբերությունը պարզաբանելու համար դիտարկենք Նկ. 6.2-ում բերված օրինակը:



Նկ. 6.2

For Loop-ում տեղադրված է պատահական թվի գեներատոր, որի պարբերությունը 100 մվ է: Իտերացիաների սահմանված թիվը 100 է: Waveform Graph-ը տեղադրված է ցիկլից դուրս, իսկ Waveform Chart-ը՝ դրա ներսում: Ի վերջո, երկուսն էլ ցույց են տալու միևնույն արդյունքը, սակայն Waveform Chart-ը գրաֆիկի վրա ցույց կտա ավելացված 100 կետերից յուրաքանչյուրը կետի հայտնվելու պահին, իսկ Waveform Graph-ը տվյալները ցույց կտա միայն ցիկլի ավարտվելուց հետո (Նկ. 6.3): Քանի որ Waveform Chart-ը ունի տվյալները պահելու համար նախատեսված բուֆեր, ամեն գործարկման ժամանակ գեներացված 100 պատա-

հական թվերը կավելանան նախորդներին, իսկ Waveform Graph-ի վրա կերևան միայն տվյալ գործարկման ժամանակ գեներացված կետերը:



Նկ. 6.3. Նախորդ նկարում ներկայացված Waveform Chart-ի և Waveform Graph-ի ցույց տված տվյալները 1-ին և 2-րդ գործարկումներից հետո

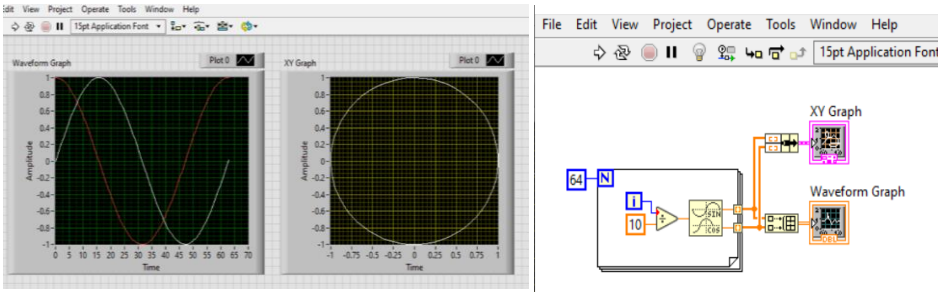
Բուֆերում պահվող կետերի քանակը կարելի է փոխել Front Panel-ից՝ մկնիկի աջ կոճակով սեղմելով Graph/Chart-ի վրա և ընտրելով Graph/Chart History Length-ը: Տվյալների պատմությունը հնարավոր է ջնջել՝ օգտագործելով համապատասխան Property node-ը կամ աջ կոճակ → Data Operations → Clear Graph/Chart:

Graph-ում և Chart-ում հնարավոր է կառուցել մեկից ավել կորեր և փոփոխել դրանց գույները, հաստությունները, ինչպես նաև տեսքերը (հոծ գիծ, կետգիծ, ընդհատ գծեր…) և այլն:

Հնարավորություն կա նաև կառուցելու XY Graph, Intensity Graph և Intensity Chart: XY Graph-ը ընդհանուր նշանակության գրաֆիկական պատկերման ունիվերսալ մեթոդ է, որը նախատեսված է դեկարտյան երկչափ կոորդինատական համակարգում բազմարժեք ֆունկցիաների (օրինակ՝ կամայական տեսքի փակ կորեր կամ փոփոխական քայլով տվյալներ) տվյալների պատկերման համար: XY Graph-ը համատեղելի է երեք տեսակի մուտքային տվյալների հետ.

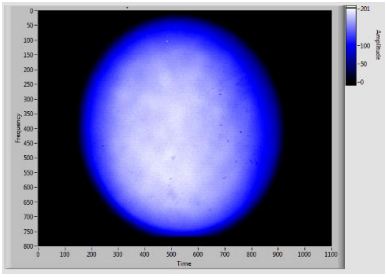
1) Կլաստեր, որը կազմված է x և y առանցքներին համապատասխանող զանգվածներից։
2) Զանգված, որի տարրերը X և Y կետերը պարունակող կլաստերներ են։
3) Զանգված, որի տարրերը կոմպլեքս թվեր են։ Այս դեպքում իրական մասը համապատասխանում է x, իսկ կեղծ մասը՝ y առանցքին։

Նկ. 6.4-ի ձախ մասում ներկայացված են սինուսի և կոսինուսի ժամանակային տեսքերը, իսկ աջ մասում՝ դրանց XY Graph-ը։



Նկ. 6.4. Waveform Graph և XY Graph

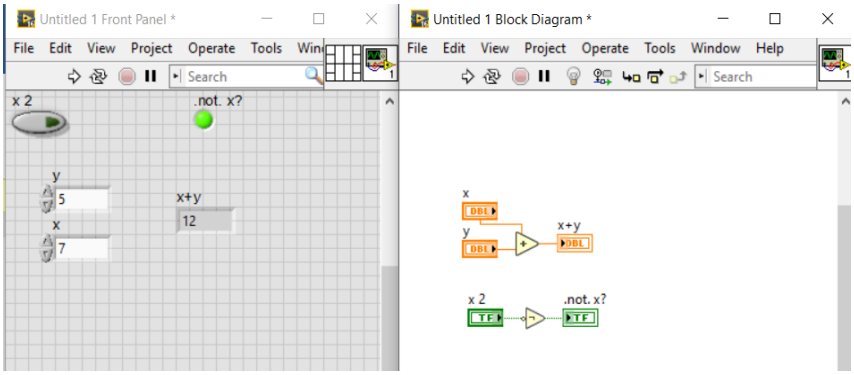
Intensity Graph-ը կամ Chart-ը օգտագործվում են երկչափ դեկարտյան կոորդինատական համակարգում՝ եռաչափ տվյալների պատկերման համար, օրինակ՝ տեղանքի ռելիեֆ, ջերմաստիճան և այլն (Նկ. 6.5)։ Այստեղ երկչափ զանգվածի տվյալ տարրի ինդեքսները համապատասխանում են Intensity Graph-ի կամ Chart-ի վրա դրա դիրքին, իսկ տարրը՝ մեծությանը, որն արտահայտվում է գույնի միջոցով։ Անհրաժեշտ է նշել, որ երկչափ զանգվածի սյուները Intensity Graph-ի կամ Intensity Chart-ի վրա կարտապատկերվեն որպես տողեր, իսկ զանգվածի տողերը Intensity Graph-ի կամ Intensity Chart-ի վրա որպես տողեր արտապատկերելու համար պետք է մկնիկի աջ կոճակի սեղմումով բացվող ցանկից ընտրել Transpose Array-ը։



Նկ . 6.5. Intensity Graph

7. Ենթածրագրեր (SubVI): Գործողություններ ֆայլերի հետ (File I/O)

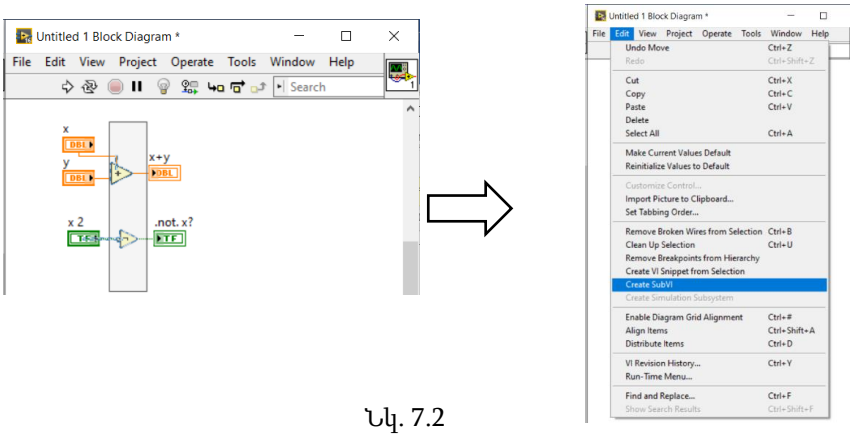
Երբեմն անհրաժեշտ է որոշակի ծրագրեր հաճախակի օգտագործել հիմնական ծրագրում: Այդ դեպքում հարմար է այն ներկայացնել որպես ենթածրագիր:



Նկ. 7.1

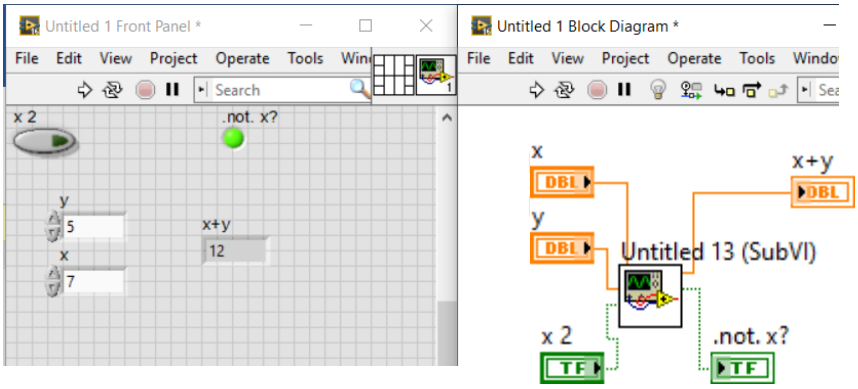
Դիցուք՝ ունենք Նկ. 7.1-ում ցույց տրված ծրագիրը:

Ձևափոխենք Նկ. 7.1-ում պատկերված ծրագիրը ենթածրագրի՝ մկնիկով նշելով ծրագրի անհրաժեշտ օպերատորը(ները) և Block Diagram-ում ընտրելով Create SubVI ֆունկցիան (Նկ. 7.2):



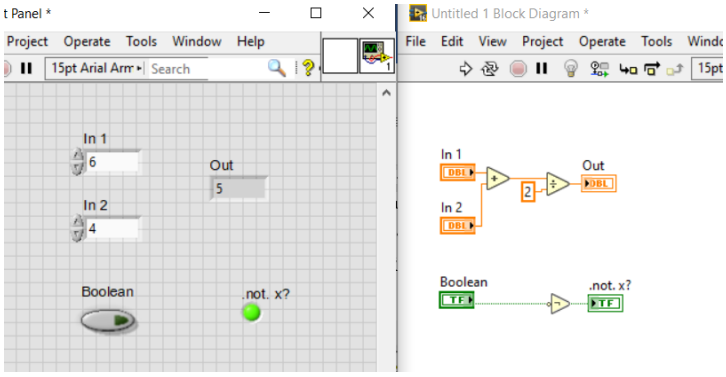
Նկ. 7.2

Արդյունքում կունենանք Նկ. 7.3-ում ցուցադրված կառուցվածքը, որտեղ կողի տվյալ մասը ենթածրագրի վերծանելուց հետո *LabVIEW*-ն գեներացնում է ենթածրագրի՝ ստանդարտ նշանով և հերթական համարով գրաֆիկական պատկեր, իսկ միացումները իրականանում են վերևից ներքև հաջորդականությամբ: Այստեղ մուտքերը դասավորված են միացման պանելի մեջտեղով ուղղահայաց ձգվող գծից ձախ, իսկ ելքերը՝ աջ: Բացառություն են կազմում Error տիպի կլաստերները, որոնք ավտոմատ միանում են ներքևից: Սա Sub VI-ի ստեղծման պարզագույն տարբերակն է: Մյուս եղանակը հանգում է առանձին մուտքերի և ելքերի, դրանց միացման սխեմայի և տեղերի, ինչպես նաև VI-ի արտաքին տեսքի ընտրությանը:



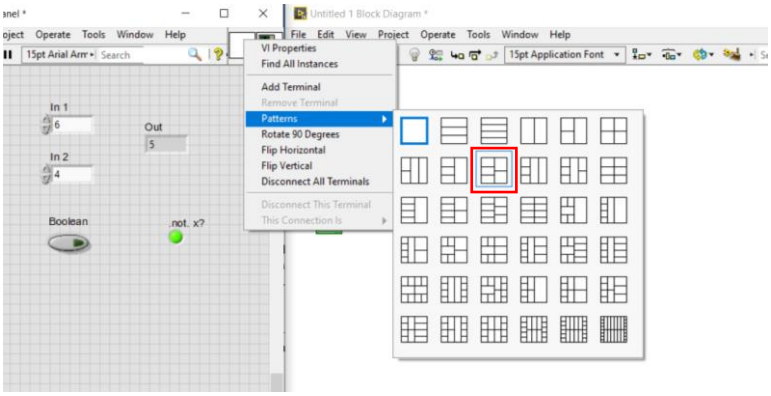
Նկ. 7.3

Դիտարկենք Նկ. 7.4-ում բերված պարզագույն ենթածրագիրը, որը հաշվում է երկու՝ In 1 և In 2 թվերի միջին թվաբանականը:



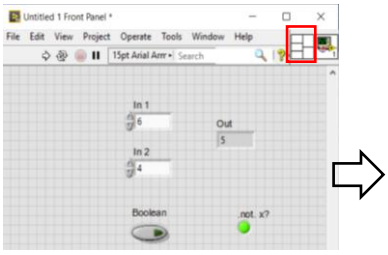
Նկ. 7.4

Front Panel-ի վերին աջ անկյունում մկնիկի աջ կոճակի օգնությամբ բացվող մենյուից ընտրում ենք Patterns բաժնի կարմիր ուղղանկյունով նշված SubVI-ի օրինակը, որի տերմինալների քանակը ցանկալի է, որ ընդհանուր դեպքում համընկնի ծրագրի փոփոխականների քանակի հետ (Տե՛ս Նկ. 7.5)։

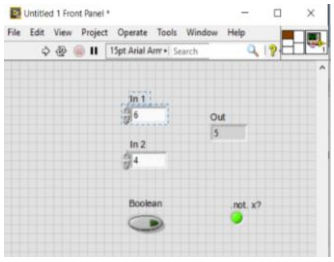


Նկ. 7.5

Միացման համապատասխան կառուցվածքը ընտրելուց հետո Front Panel-ի վերևի աջ մասում կունենանք Նկ. 7.6-ում պատկերված տեսքը։ Սեղմելով տերմինալի ցանկալի բջջի, այնուհետև Front Panel-ում նախընտրելի փոփոխականի վրա՝ կունենանք այդ երկուսի միացումը/համապատեխանեցումը (Տե՛ս Նկ. 7.7)։



Նկ. 7.6

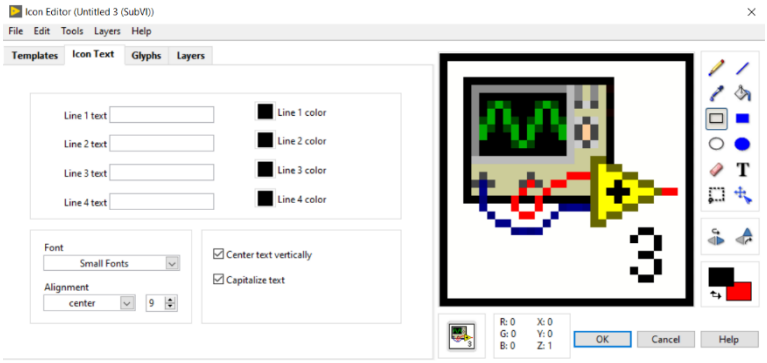


Նկ. 7.7

Ներկված վանդակները նշանակում են, որ դրանց միացված է Control կամ Indicator, իսկ որևէ ներկված հատվածի վրա մկնիկի ձախ կոճակի սեղմումով Front Panel-ի վրա կընդգծվի տվյալ տերմինալը։ Միացման պանելի մուտքային/ելքային հնարավոր տերմինալների քանակը կարելի է փոխել՝ մկնիկի աջ կոճակով սեղմելով վերջինիս վրա և Patterns

մենյուից ընտրելով ցանկալի դիզայնը: Նկատենք, որ միացման պանելի տերմինալները ընդունում են տարբեր գույներ՝ կախված տվյալի տիպից (DBL-նարնջագույն, INT-կապույտ, BOOL-կանաչ, CLUSTER-մանուշակագույն և այլն): Պանելից որևէ տերմինալ անջատելու համար սկզբում պետք է սեղմել մկնիկի ձախ կոճակը, այնուհետև աջը և ընտրել Disconnect this terminal: Տերմինալի միացման վանդակը հնարավոր է տեղափոխել՝ մկնիկի ձախ կոճակով նշելով միացման պանելի համապատասխան վանդակի վրա, այնուհետև սեղմած պահելով Ctrl կոճակը և նշելով միացման պանելի այլ վանդակ:

Ընդհանուր դեպքում ենթածրագրի ընթեռնելիությունը հեշտացնելու համար ցանկալի է, որ դրա գրաֆիկական պատկերն արտացոլի ֆունկցիոնալությունը: Գրաֆիկական պատկերը խմբագրելու համար պետք է մկնիկի ձախ կոճակը երկու անգամ սեղմել գրաֆիկական վանդակի վրա կամ կատարել մկնիկի աջ կոճակ → Edit Icon… քայլերը: Կբացվի Նկ. 7.8-ում բերված պատուհանը, որը հնարավորություն է տալիս կատարել պարզագույն գրաֆիկական գործողություններ, բացի այդ՝ ստեղծել ձևանմուշներ (Templates), ավելացնել տեքստ (Icon Text), բազում գրաֆիկական պատկերներ (Glyphs), ինչպես նաև ղեկավարել պատկերների շերտերը (Layers)՝ ավելացնելով, ջնջելով կամ թաքցնելով առանձին շերտեր:

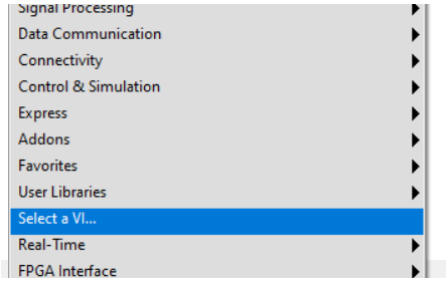


Նկ. 7.8

Վերադառնանք վերը դիտարկված միջին թվաբանականը հաշվող ենթածրագրին: Դրան համապատասխան գրաֆիկական պատկեր ստեղծելու համար նախ Layers խմբի User Layers բաժնից պետք է նշել VI Icon շերտը և ջնջել այն, այնուհետև Templates բաժնից ընտրել համապատաս-

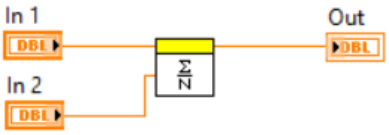
խան ձևանմուշ-կմախք, Glyphs բաժնից ֆիլտրի միջոցով գտնել ենթածրագրի ֆունկցիոնալությունը ներկայացնող պատկեր (այս պարագայում մենք օգտագործել ենք average բառը՝ պատկերը ֆիլտրելու համար) և մկնիկի ձախ կոճակի միջոցով այն տեղադրել գրաֆիկական պատկերի նախընտրելի մասում և ցանկության դեպքում գունավորել պատկերը:

Բոլոր նախընտրելի գործողությունները ավարտելուց հետո պետք է պահպանել File/Save-ի օգնությամբ: Այնուհետև մեկ այլ VI-ում այն ներմուծելու համար դրա Block Diagram-ում մկնիկի աջ կոճակով բացվող մենյուից Select a VI-ի միջոցով ընտրել SubVI ֆայլը (Նկ. 7.9):



Նկ. 7.9

Արդյունքում ենթածրագիրը կընդունի հետևյալ տեսքը.

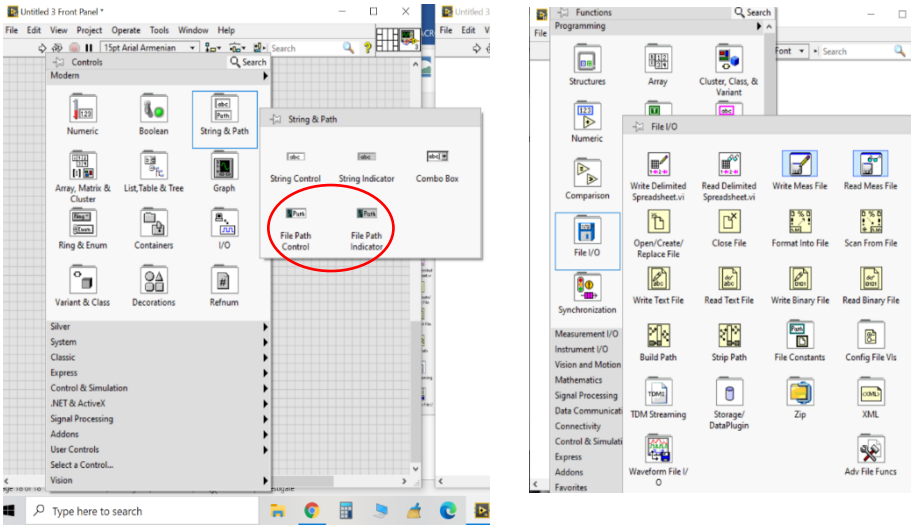


Նկ. 7.9

Տվյալների գրելը և կարդալը ֆայլերում(ից)

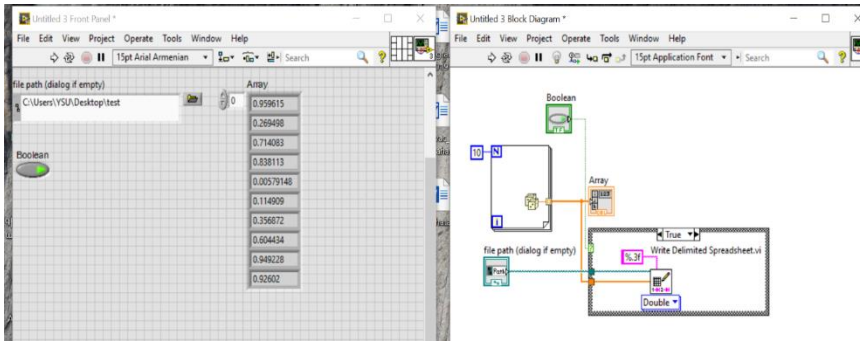
Տվյալները պահպանելու կամ կարդալու համար *LabVIEW*-ն ունի բավական հարուստ գործիքակազմ, հետևաբար այն կարելի է իրականացնել տարբեր եղանակներով: Ծանոթանանք դրանցից մեկին:

Block Diagram-ում կա File I/O, իսկ Front Panel-ում՝ String&Path բաժինը (կարմիրով նշված), որոնք ցույց են տրված Նկ. 7.10-ում:



Նկ. 7.10

Գեներացնենք 0-1 միջակայքի պատահական 10 թվեր և պահպանենք դրանք test անվամբ ֆայլում: Այս գործողությանը համապատասխանող ծրագիրը բերված է Նկ. 7.11-ում:

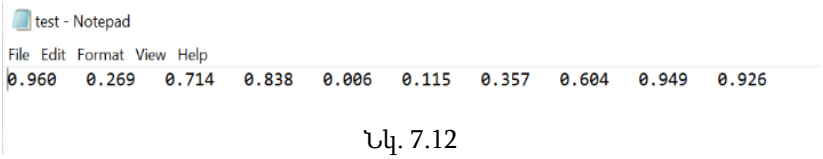


Նկ. 7.11

Պատահական թվերը գեներացվում են For Loop ցիկլի օգնությամբ, այնուհետև Case Structure-ի Boolean կոճակի միջոցով պահպանվում ֆայլում: Ֆայլում պահպանված տվյալներն ունեն Նկ. 7.12-ում պատկերված տեսքը:

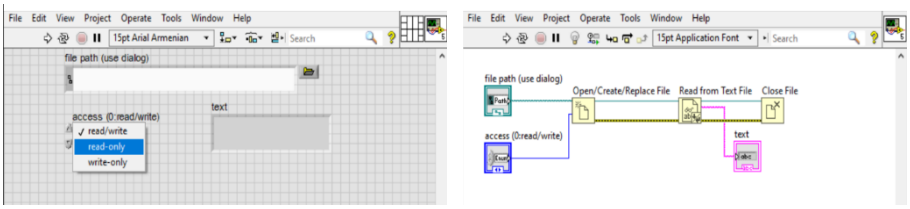
Ֆայլի անունը և հասցեն պետք է նախապես գրել Front Panel-ի File Path դաշտում: Write Delimited Spreadsheet.vi գործիքի Double (Integer,

String)-ը ցույց է տալիս տվյալների տիպը, իսկ %.3f կետից հետո՝ թվերի քանակը: Պահպանվող տվյալները կարող են լինել ինչպես 1D, այնպես էլ՝ 2D:



Նկ. 7.12

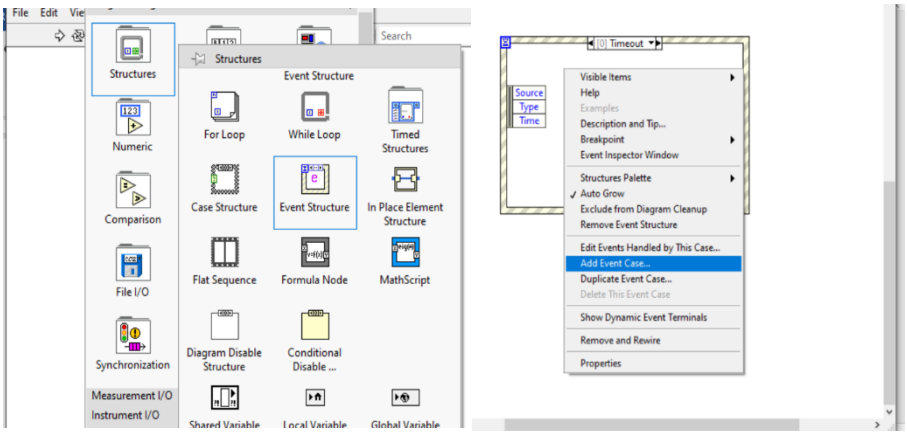
Ծրագիրն ավելի հուսալի աշխատեցնելու համար ցանկալի է օգտագործել Նկ. 7.13 գործիքակազմը: Այստեղ File path–ը նախատեսված է ֆայլի անունը գրելու, իսկ access-ը՝ գործողության տեսակը (գրել/կարդալ) ընտրելու համար: Ֆայլում(ից) գրելու կամ կարդալու համար պետք է օգտագործել համապատասխանաբար Write Text/Binary File կամ Read from Text/Binary File գործիքները:



Նկ. 7.13

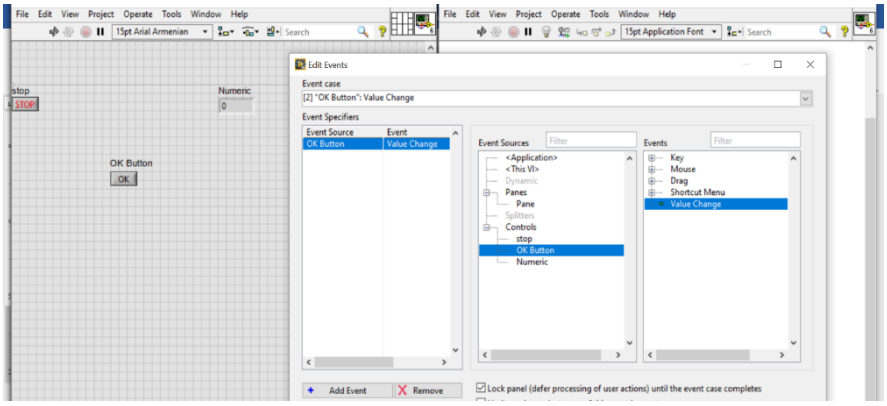
8. Իրադարձությունների կառուցվածք (Event Structure): Երկխոսության պատուհաններ (Dialog)

LabVIEW միջավայրում ծրագրի կատարման ընթացքը ղեկավարելու հնարավորությունը ներդրված է իրադարձությունների կառուցվածքում (Event Structure)։ Այս հանգույցի առկայության դեպքում ծրագրի կատարման ընթացքը կախված է որևէ իրադարձությունից (մկնիկի հետ աշխատանք՝ տեղաշարժ, ստեղնի սեղմում, ստեղնաշարի որևէ ստեղնի սեղմում, որևէ հանգույցի արժեքի փոփոխություն և այլ)։ Event Structure-ը գտնվում է Block Diagram-ի Structure բաժնում (Տե՛ս Նկ. 8.1)։ Սկզբնական վիճակում այն ունի աշխատանքային մեկ էջ, այնուհետև մկնիկի աջ կոճակի օգնությամբ կարելի է ավելացնել էջերի քանակը՝ Add Event Case, և յուրաքանչյուր էջ կապել որևէ իրադարձության (պայմանի) հետ:



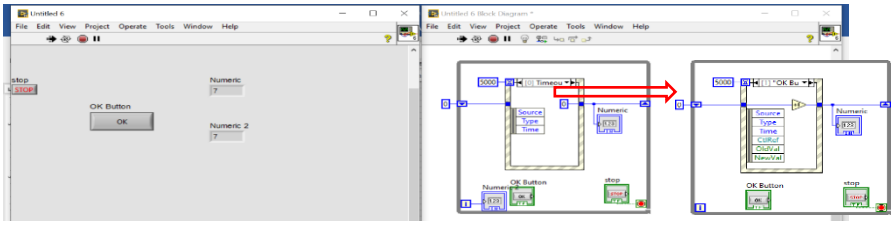
Նկ. 8.1

Օրինակ՝ ցիկլի մեջ կարելի է տեղադրել Event Structure, ապա ավելացնել մի իրադարձություն և որպես դրա աղբյուր ընտրել OK Button կոճակը (կարելի է ընտրել Event Sources պատուհանում ցանկացածը), այնուհետև Events պատուհանից ընտրել Value Change (արժեքի փոփոխություն) ֆունկցիան, ինչպես ցույց է տրված Նկ. 8.2-ում:



Նկ. 8.2

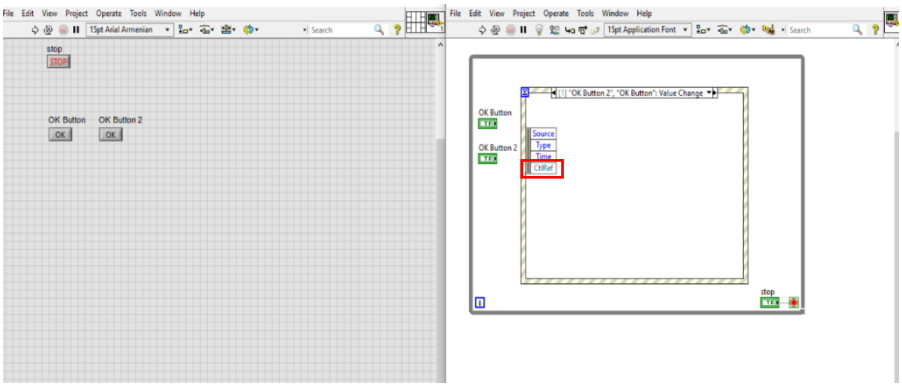
Ավելացնելով Shift Register-ը և գործարկելով ծրագիրը՝ կտեսնենք, որ ցիկլը չի աշխատում, այսինքն՝ Numeric2 ինդիկատորի ցուցմունքը չի փոխվում՝ Նկ. 8.3:



Նկ. 8.3

Սա հենց Event Structure-ի կարևորագույն հատկություններից է, որ ավելորդ գործողություն ցիկլը չի կատարում, մինչև որևէ իրադարձություն տեղի չունենա: Յուրաքանչյուր անգամ սեղմելով Ok Button-ը՝ Numeric-ի ցուցմունքը մեկով կավելանա: Event Structure-ի ձախ անկյունում գրված 5000-ը ցույց է տալիս, որ եթե 5 վ-ի ընթացքում որևէ իրադարձություն տեղի չունենա, ապա այն կանցնի Timeout էջին, որտեղ էլ Numeric-ի ցուցմունքը կզրոյացվի: Իսկ եթե Timeout էջին միացված է -1 արժեքը, ապա ծրագիրը կսպասի անվերջ երկար ժամանակ: Այսպիսով, հնարավորություն է ստեղծվում ղեկավարել ծրագրի կատարման ընթացքը օգտագործողի կողմից:

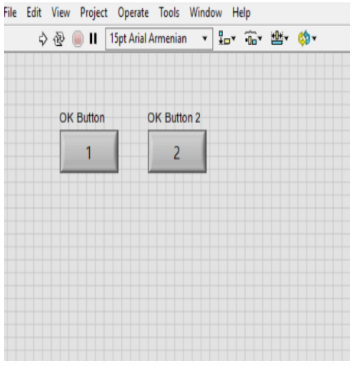
Ներկայացնենք մեկ այլ հարմարավետ ֆունկցիա՝ CtlRef: Նկ. 8.4-ի ձախ մենյուում այն ավելացնելու համար անհրաժեշտ է մկնիկի աջ կոճակով ընտրել Select Item\CtlRef:



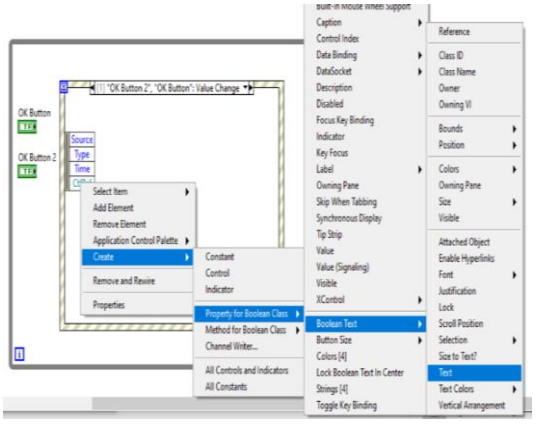
Նկ. 8.4

Front Panel-ում տեղադրենք երկու կոճակ՝ OK Button և OK Button՝ 2 անուններով, իսկ OK տեքստային գրառումները փոխարինենք համապատասխանաբար 1-ի և 2-ի (Նկ. 8.5): Block Diagram-ում CtlRef-ի վրա մկնիկի աջ կոճակի սեղմումով պետք է ընտրել Create\Prorerty for Boolean Class\Boolean Text\Text ֆունկցիան (Նկ. 8.6):

Արդյունքում կունենանք ծրագրի՝ Նկ. 8.7-ում ներկայացված տեսքը, որտեղ յուրաքանչյուր կոճակի սեղմման դեպքում Boolean Text Text ցուցիչի վրա կհայտնվի համապատասխան ստեղնի վրա գրված տեքստը:

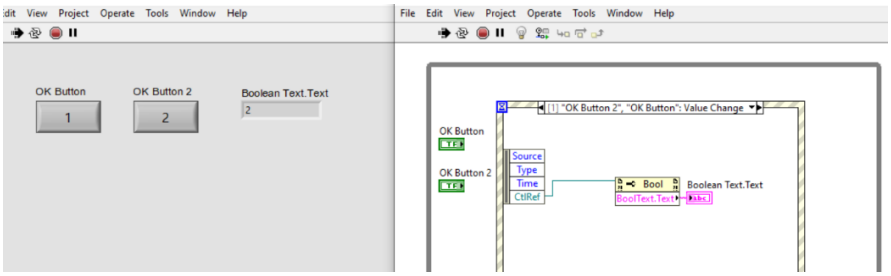


Նկ. 8.5



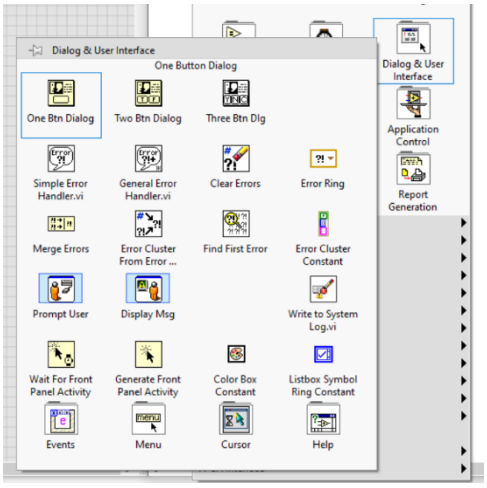
Նկ. 8.6

Event Structure-ը ունի բազմաթիվ ֆունկցիոնալ հնարավորություններ, որի կիրառության օրինակներից է հաշվիչի (Calculator) ծրագրավորումը, ինչպես նաև տարատեսակ ֆունկցիոնալություններով օգտագործողի ինտերֆեյսերի (user interface) կառուցումը:



Նկ. 8.7

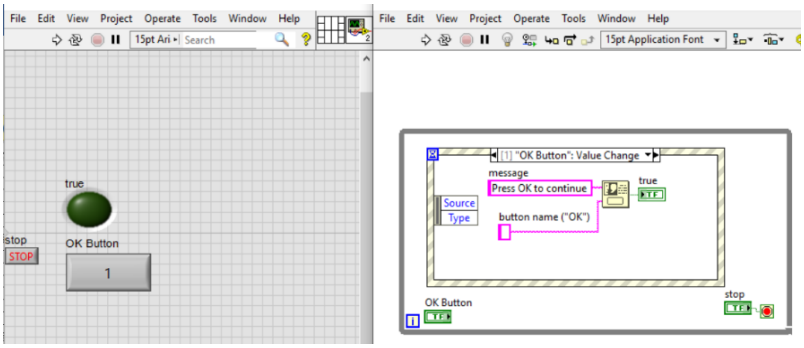
Երկխոսության պատուհաններ (Dialog)



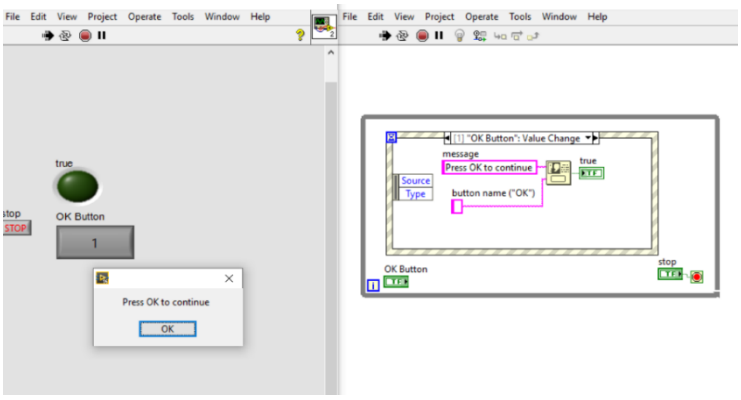
Նկ. 8.8

Օգտագործողի համար ծրագրի ինտերֆեյսը ավելի հարմար դարձնելու համար կիրառվում են երկխոսությունների պատուհաններ, որոնց միջոցով օգտագործողը որոշում է ծրագրի կատարման ընթացքը: Դիալոգները գտնվում են Block Diagram-ի Dialog&User Interface բաժնում (Նկ. 8.8): Դիալոգները Front Panel-ում առաջացող լրացուցիչ պատուհաններ են, որոնք ի հայտ են գալիս որոշակի պայմանների դեպքում: OK Button կոճակը միացնելով Event Structure-ի առաջին էջին, տեղադրելով One Button Dialog գործիքը և միացնելով համապատասխան սթրինգ հաստատունները՝ կունենանք Նկ. 8.9-ում ցույց տրված կառուցվածքը:

Գործարկելով ծրագիրը՝ Front Panel-ում կհայտնվի փոքրիկ պատուհան՝ մեկ OK կոճակով, և այն սեղմելու դեպքում կվառվի True լույսը՝ Նկ. 8.10:



Նկ. 8.9

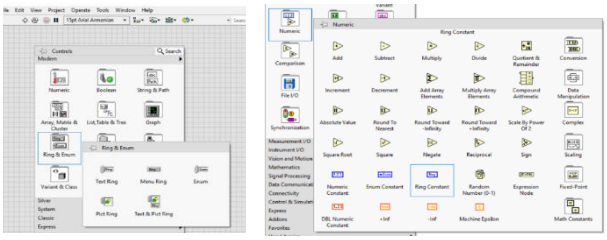


Նկ. 8.10

Մյուս գործիքներով հնարավոր է գեներացնել երկու կամ երեք ղեկավարման ստեղնով երկխոսության պատուհաններ։

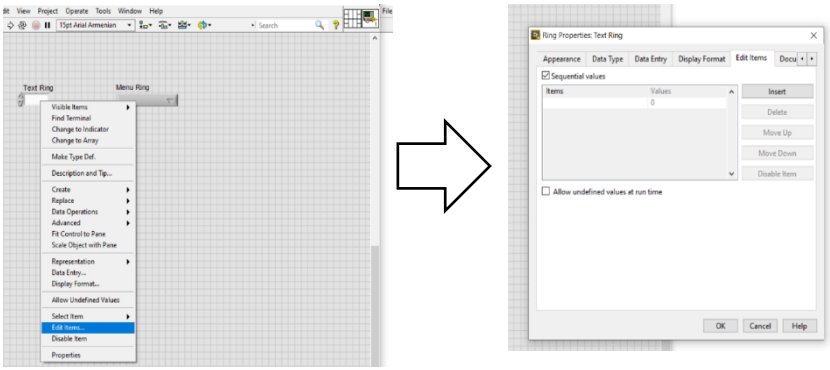
9. Text Ring, Enumerated type controls: State machine

Ring-երը և Enum-ները Front Panel-ում գտնվում են Ring&Enum բաժնում, իսկ Block Diagram-ում Numeric բաժնում՝ միայն հաստատունի տեսքով (Նկ. 9.1)։



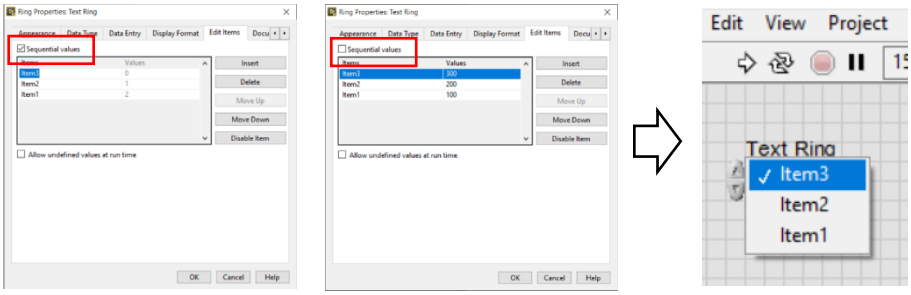
Նկ. 9.1

Ring-ում և Enum-ում տարրերի ավելացման համար Front Panel-ում պետք է տեղադրել Text Ring և Menu Ring Control-ները, այնուհետև մկնիկի աջ կոճակով ցանկից ընտրել Edit Items…-ը, ինչպես ցույց է տրված Նկ. 9.2-ում։



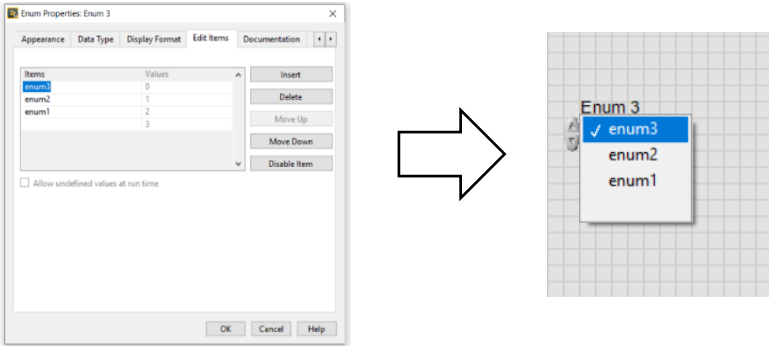
Նկ. 9.2

Insert-ով և Delete-ով հնարավոր է ավելացնել կամ ջնջել տարրերը։ Նկ. 9.3-ում պատկերված Sequential values-ի նշելու դեպքում ամեն հաջորդական տարրին վերագրվում են հաջորդական թվային արժեքներ, իսկ անջատված վիճակում տարրերին կարող են վերագրվել ներմուծված ցանկացած թվային արժեք։



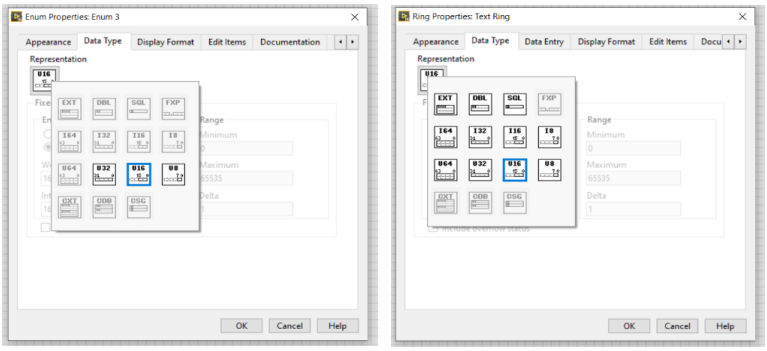
Նկ. 9.3

Ընդհանուր առմամբ, Enum-ը ու Ring-ը շատ նման են միմյանց, սակայն կան որոշակի տարբերություններ: Մասնավորապես, ի տարբերություն Ring-ի, Enum-ում Sequential values տարբերակը բացակայում է (Նկ. 9.4)։



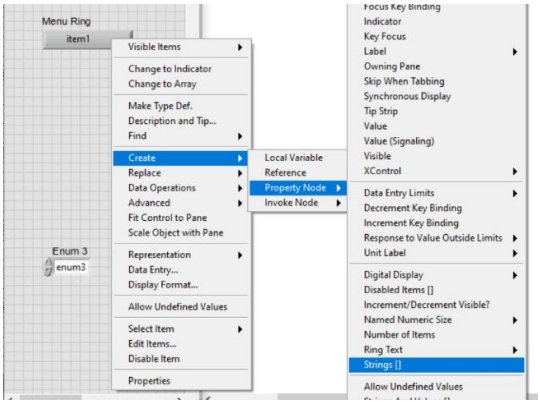
Նկ. 9.4

Enum-ի և Ring-ի թվային արժեքների ընդունման տիպերի տարբերությունները պատկերված են Նկ. 9.5-ում։



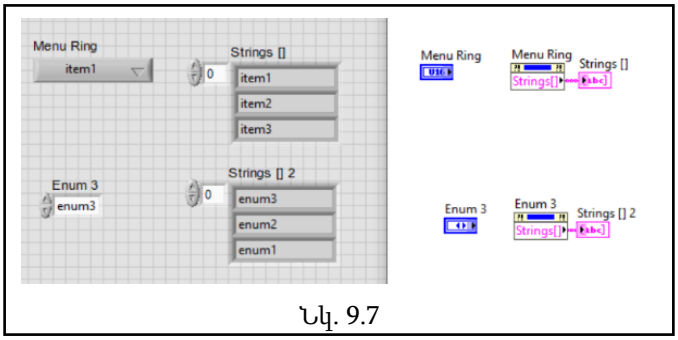
Նկ . 9.5

Ընտրենք երկուսի դեպքում էլ Strings[] ֆունկցիան Նկ. 9.6:

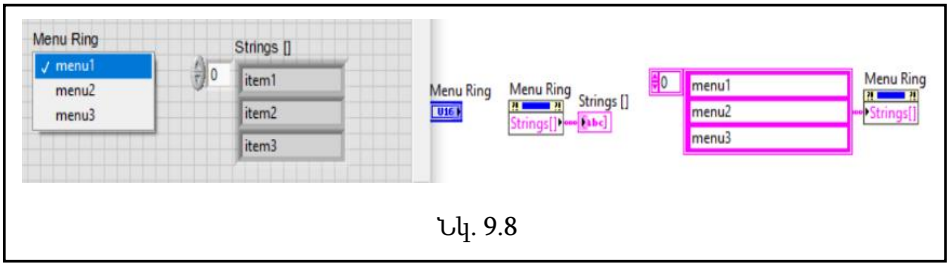


Նկ. 9.6

Արդյունքում Ring-ի և Enum-ի դեպքում էլ հնարավոր է կարդալ սթրինգները (Նկ. 9.7), իսկ գրելու դեպքում միայն Ring-ն է հնարավորություն տալիս փոփոխելու (Նկ. 9.8):

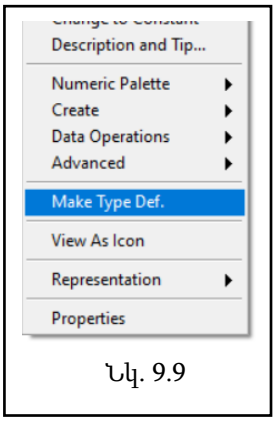


Նկ. 9.7



Նկ. 9.8

Եթե Item-ների քանակը ծրագրում ավելացնում կամ պակասեցնում ենք, և Ring-ը կամ Enum-ը մի քանի անգամ ծրագրում օգտագործվել են, ապա Make Type Def. ֆունկցիան երկուսի դեպքում էլ հնարավորություն է



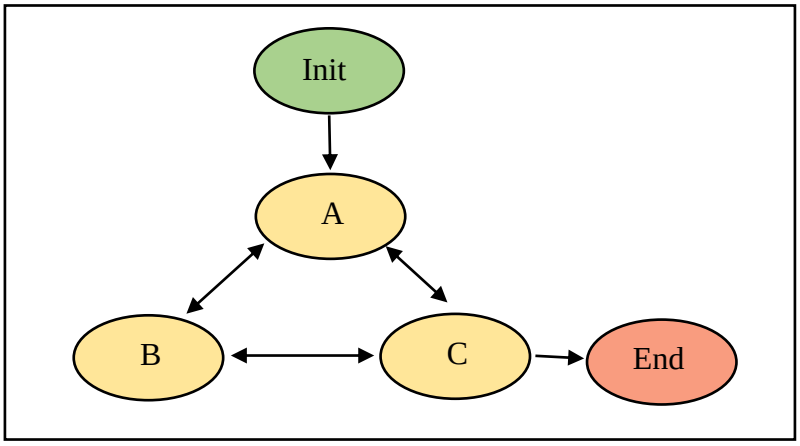
Նկ. 9.9

տալիս փոփոխությունը տարածել բոլոր տեղերում օգտագործված Ring-երի և Enum-ների վրա (Նկ. 9.9)։

State machine (Վիճակի մեքենա)

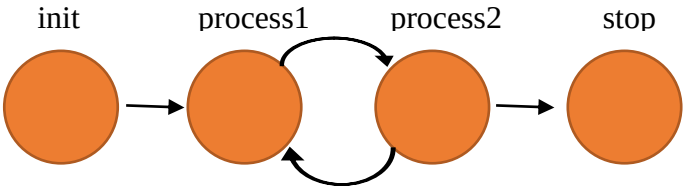
State machine-ը օգտագործվում է *LabVIEW* միջավայրում՝ բարդ ալգորիթմներում որոշումների իրականացման համար։ Ենթադրենք՝ ունենք A, B և C տարբեր վիճակներ, և անհրաժեշտ է անցում կատարել մեկից մյուսին։ Այս դեպքում հարմար է օգտագործել վիճակի մեքենան։ Վիճակի մեքենայի աշխատանքային տեսքը ներկայացված է Նկ. 9.10-ում։

Init-ը համապատասխանում է սկզբնական վիճակին, որից հետո անցում է կատարվում A վիճակի։ Այնուհետև, ըստ գրված ծրագրի հերթականության, հնարավոր են կամայական անցումներ A, B և C վիճակների միջև տարբեր ուղղություններով։ End վիճակը համապատասխանում է ծրագրի ավարտին։



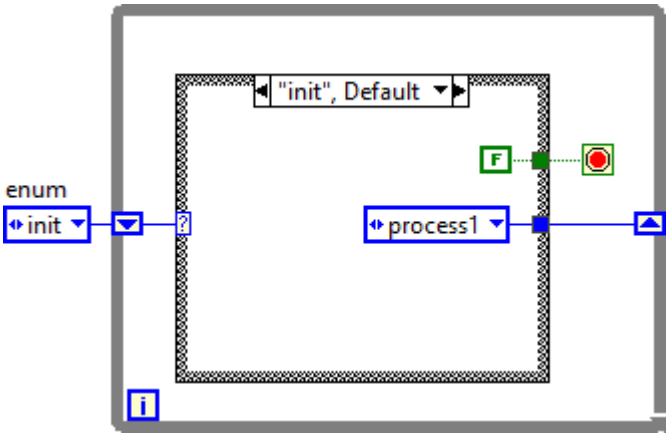
Նկ.9.10

Դիտարկենք Նկ. 9.11-ում պատկերված օրինակը։



Նկ. 9.11

Ծրագիրը սկզբնական՝ init վիճակից անցում է կատարում process1 վիճակի, որից հետո՝ process2-ի, այնուհետև՝ հակառակը, և այսպես շարունակ այնքան ժամանակ, մինչև որոշակի պայմանի բավարարման դեպքում ծրագիրը անցում կկատարի stop վիճակ։

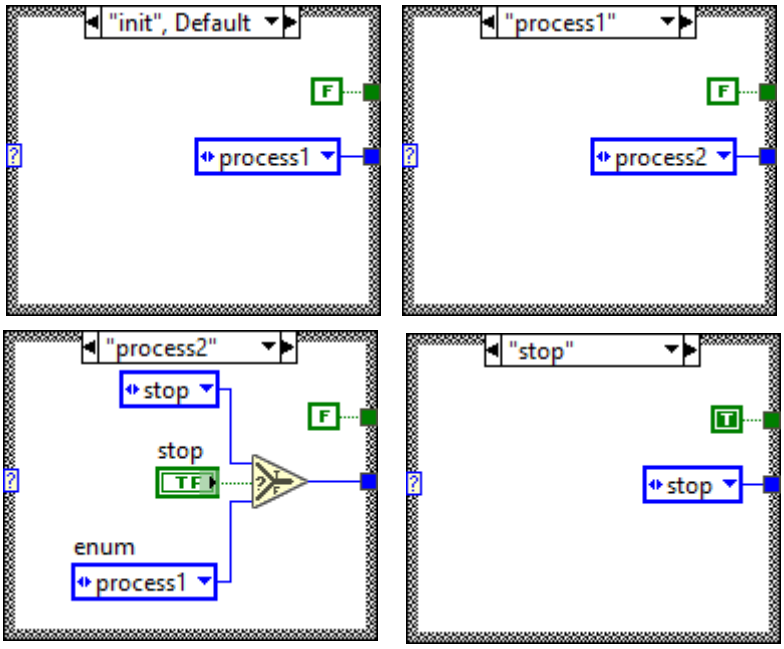


Նկ. 9.12

Ծրագիրը կունենա Նկ. 9.12-ում պատկերված տեսքը։ Ինչպես տեսնում ենք, մի վիճակից մյուսին անցնելու համար օգտագործվել է Enum փոփոխականը, սակայն կարելի է օգտագործել նաև Ring։

Գործարկելուց հետո այն անցում է կատարում init վիճակի, որտեղից էլ հրահանգում է գնալ դեպի process1, հետո process2, այնուհետև նորից process1, այնքան ժամանակ, մինչև որ stop պայմանը կլինի True։ Այդ դեպքում անցում կկատարի stop վիճակի և կավարտվի։

Նկ. 9.13-ում ցույց է տրված Case Structure-ի մնացած վիճակների տեսքերը:



Նկ . 9.13

10. UDP և TCP

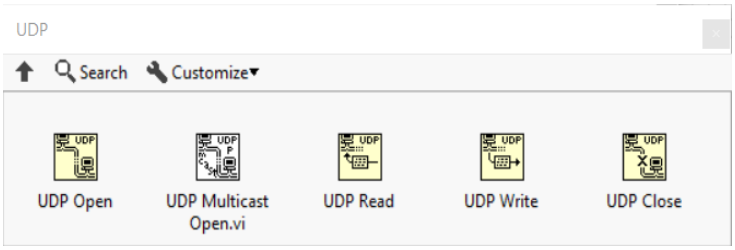
Հաճախ գիտափորձերի ավտոմատացման գործընթացում կարիք է առաջանում համակարգների մշտադիտարկման և հեռավար կառավարման՝ միևնույն ցանցի ներսում գտնվող այլ համակարգչի միջոցով: Ավելին՝ փորձարարական տվյալների հավաքագրման և վերլուծության համար հարմար է օգտագործել այլ համակարգիչ, որը անմիջապես կապված չէ փորձարարական սարքերի հետ: Քանի որ բազմաթիվ սարքեր համատեղելի են *LabVIEW*-ի հետ, և գոյություն ունեն *LabVIEW* միջավայրում դրանց ղեկավարման համար անհրաժեշտ ծրագրային փաթեթները, ապա տվյալների փոխանցումը ցանցի ներսում տարբեր համակարգիչների միջև նույնպես հարմար է իրականացնել *LabVIEW*-ի միջոցով: User Datagram Protocol-ը (UDP) և Transmission Control Protocol-ը (TCP) ցանցի ներսում կոմունիկացիայի հիմնական միջոցներն են:

UDP

UDP-ն ապահովում է միևնույն ցանցի ներսում համակարգիչների միջև ինֆորմացիայի պարզ և հիմնական մակարդակի (low-level) փոխանակում: Համակարգիչների տարբեր պրոցեսների միջև հաղորդակցությունը իրականանում է նպատակակետ հանդիսացող համակարգչի պորտին դատագրամներ ուղարկելով: Վերջիններս հանդիսանում են ցանցային հաղորդակցման միավոր տվյալները, որոնք, որպես կանոն, պարունակում են ուղարկվող ինֆորմացիան, ինչպես նաև ուղարկողի և ստացողի հասցեները: Պորտը տվյալ ցանցի ներսում համակարգչի այն վիրտուալ կետն է, որից ուղարկվում և որտեղ ստացվում է դատագրամը (ինֆորմացիան): Եթե տվյալ պորտը բաց չէ (կապը հաստատված չէ), ապա UDP-ն հեռացնում է դատագրամը: UDP-ն օգտագործվում է այնպիսի խնդիրներում, որտեղ ինֆորմացիայի ստացման հուսալիությունը և տվյալների որոշ մասի կորուստը կրիտիկական չեն: UDP-ն ուղարկում է տվյալներ՝ առանց նպատակակետի կողմից դրա ստացման հաստատումը ստանալու անհրաժեշտության: Այլ կերպ ասած՝ UDP-ն կարող է հաղորդել ինֆորմացիա՝ անկախ այն բանից՝ կա՞ն արդյոք այդ ինֆորմացիային սպասող պրոցեսներ, թե՞ ոչ, ինչը UDP-ն հարմար է դարձնում

նաև ինֆորմացիայի սփռման համար։ Եթե ուղարկվում են որոշակի հերթականությամբ տարբեր ինֆորմացիաներ, ապա UDP-ն չի երաշխավորում նաև դրանց՝ միևնույն հերթականությամբ ստացումը։

LabVIEW միջավայրում UDP-ի VI-ները կարելի է գտնել Block Diagram-ում՝ սեղմելով մկնիկի աջ կոճակը, այնուհետև՝ Data Communication → Protocols → UDP (Տե՛ս Նկ. 10.1):



Նկ.10.1

UDP Open – հաստատում է լոկալ UDP կապ՝ մեկ ստացողի դատագրամներ ուղարկելու համար՝ օգտագործելով մուտքային պորտի համարը կամ սերվիսի անունը։ Ստացող պրոցեսը պիտի կապ հաստատի միևնույն պորտի կամ սերվիսի հետ, ինչ ուղարկողը։

UDP Multicast – օգտագործվում է մեկ ուղարկողից միաժամանակ երկու և ավելի ստացող պրոցեսների դատագրամներ հաղորդելու/սփռելու համար։ Ուղարկողի համար անհրաժեշտ է սահմանել սփռման IP հասցե։ Սփռման IP հասցեները ընկած են 224.0.0.0-ից 239.255.255.255 միջակայքում։

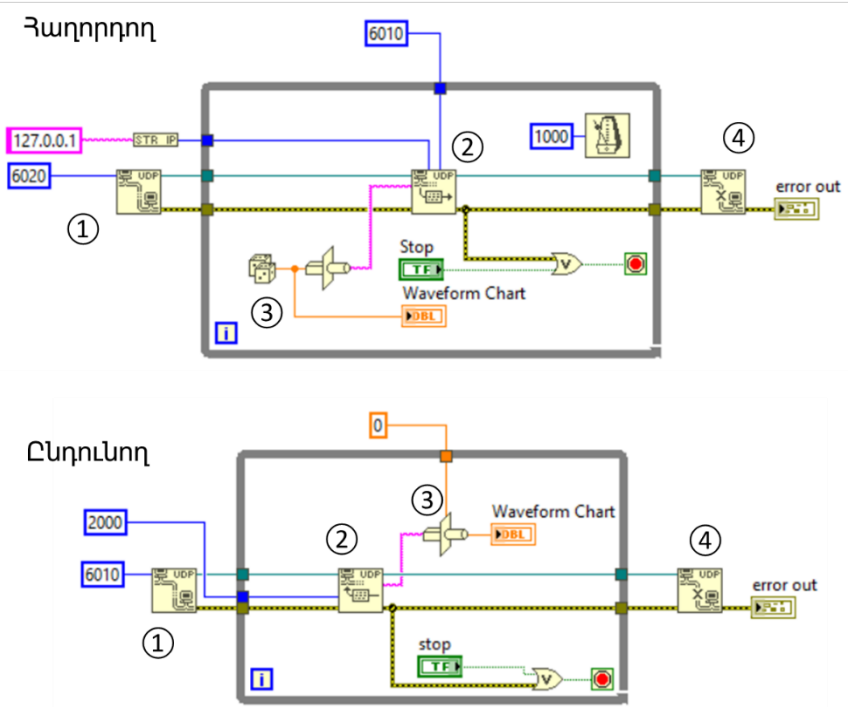
UDP Read – կարդում է UDP-ով ստացված դատագրամները։

UDP Write – UDP-ով ուղարկում է դատագրամներ։

UDP Close – փակում է կապը։

Դիտարկենք Նկ. 10.2-ում բերված ինֆորմացիայի փոխանցման պարզ համակարգի օրինակը՝ հիմնված UDP-ի վրա։

Վերևի մասում պատկերված է հաղորդող, իսկ ներքևում՝ ընդունող համակարգը։ Այստեղ դատագրամը 0-ից 1 միջակայքում գեներացված պատահական թիվ է (DBL)։



Նկ.10.2

Հաղորդող համակարգը կառուցված է հետևյալ տրամաբանությամբ.

① UDP Open VI-ը դատագրամ ուղարկելու համար հաստատում է լոկալ UDP հաղորդակցության ուղի՝ սահմանված պորտին կամ սերվիսին: Այստեղ մուտքային «6020» պորտը անհրաժեշտ է UDP կապ ստեղծելու համար, որն այն պորտը չէ, որին ուղարկվում է ինֆորմացիան:

② UDP Write VI-ը ուղարկում է դատագրամը նշված «6010» պորտին: Այստեղ որպես IP հասցե գրված է լոկալ ցանցի «127.0.0.1» հասցեն: Օգտագործվել է «String to IP» ֆունկցիան՝ տեքստային IP հասցեն թվայինի փոխակերպելու համար:

③ Ուղարկվող դատագրամը, որն իրենից ներկայացնում է պատահական թիվ՝ փոխակերպված տեքստային հաղորդագրության «Type Cast» ֆունկցիայի միջոցով: Վերջինս փոխակերպում է մի տիպի տվյալը սահմանված այլ տիպով (հիմնականում մուտքային տվյալները փոխակերպում է տեքստի):

④ UDP Close VI-ը փակում է կապը:

Դատագրամի ուղարկման պարբերությունը 1000 մվ է:

Ընդունող համակարգը կառուցված է հետևյալ տրամաբանությամբ.

① UDP Open VI-ը հաստատում է լոկալ UDP հաղորդակցության ուղի «6010» պորտի միջոցով: Նկատենք, որ սա նույն պորտն է, ինչ ***Հաղորդող համակարգի*** ② մասում UDP Write VI-ի մուտքային պորտը:

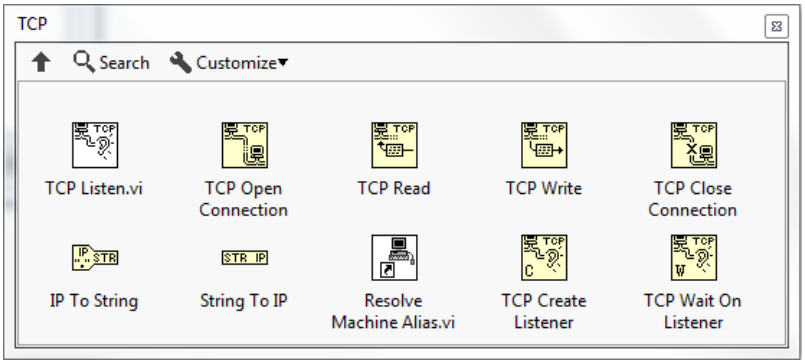
② UDP Read VI-ը կարդում է դատագրամը՝ նշված «6010» պորտից տեքստի տեսքով:

③ Ստացված տեքստային դատագրամը այնուհետև «Type Cast» ֆունկցիայի միջոցով փոխակերպվում է օրիգինալ՝ DBL տիպի թվային տվյալի, որը ցուցադրվում է Waveform Chart-ի վրա: UDP Read VI-ի սպասելու ժամանակը օրինակում 2000 մվ է, որի ընթացքում դատագրամ չստանալու դեպքում VI-ը կվերադարձնի «56» կոդով սխալ, և պրոցեսը կանգ կառնի: Նշենք, որ ընդունող համակարգի համար որպես While Loop-ի աշխատանքի պարբերության կարգավորիչ հանդիսանում է հենց Հաղորդող համակարգից տվյալների ստացման պարբերականությունը (եթե վերջինս փոքր է UDP Read VI-ի սպասելու ժամանակից):

④ UDP Close VI-ը փակում է հաղորդակցությունը:

TCP

TCP-ն հաղորդակցման կապի վրա հիմնված պրոտոկոլ է, որը ապահովում է տվյալների հուսալի փոխանցում ցանցերի ներսում՝ ապահովելով տվյալների հաղորդման հաջորդականությունը, կորուստների և կրկնությունների բացակայությունը: Ի տարբերություն UDP-ի, որտեղ հաղորդող համակարգի աշխատանքի համար ընդունողի առկայությունը և կապը դրա հետ պարտադիր չեն, TCP-ի հաղորդող համակարգը մինչև տվյալների փոխանցումը պետք է կապ հաստատի միևնույն պորտին միացած ընդունողի հետ: Կապի ապահովումից հետո միայն հաղորդող և ընդունող համակարգերը կարող են միմյանց տվյալներ փոխանցել: Այստեղ հարկ է նշել, որ «հաղորդող» և «ընդունող» անվանումները ընտրված են պայմանականորեն, քանի որ երկուսն էլ կարող են թե՛ ստանալ, թե՛ հաղորդել տվյալներ: *LabVIEW* միջավայրում TCP-ի VI-ները կարելի է գտնել Block Diagram-ում՝ սեղմելով մկնիկի աջ կոճակը, այնուհետև՝ Data Communication → Protocols → TCP (Տե՛ս Նկ. 10.3):



Նկ. 10.3

TCP Listen VI-ը ստեղծում է TCP կապ նշված պորտի հետ և սպասում տվյալ պորտին միացող համակարգների:

TCP Open Connection - կապ է հաստատում նշված IP-ով ցանցի ներսում նշված պորտի հետ:

TCP Close Connection VI - փակում է հաղորդակցությունը համակարգների միջև:

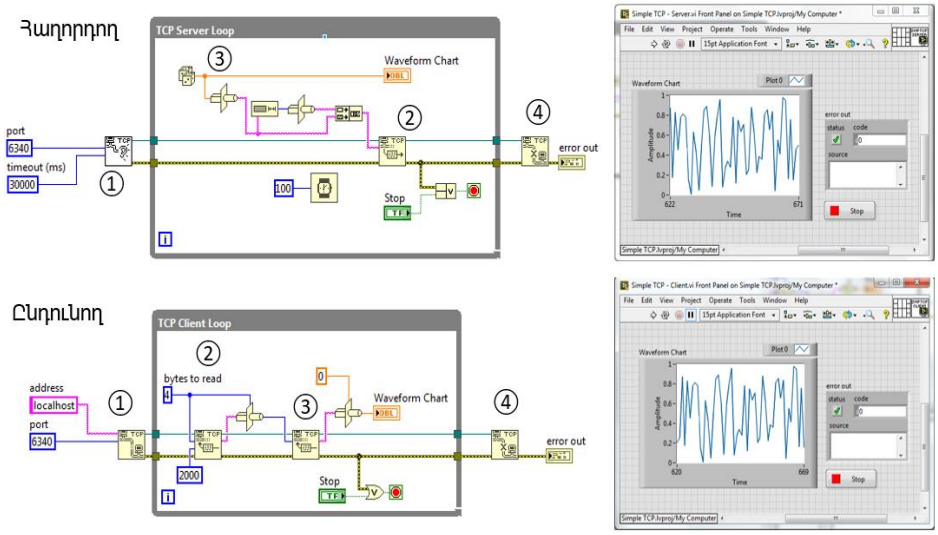
TCP Write - ուղարկում է տեքստային տիպի դատագրամներ:

TCP Read - կարդում է ուղարկված դատագրամների նշված բիթերին համապատասխանող մասը:

TCP Create Listener - օգտագործվում է մյուս կողմի համակարգից TCP Open Connection-ի հետ կապ հաստատելու համար: Այստեղ կապը պետք է լինի միևնույն պորտի հետ, ինչ TCP Open Connection-ի պարագայում:

TCP Wait on Listener - կարելի է օգտագործել While ցիկլի հետ միասին, եթե կա անհրաժեշտություն սպասելու TCP Create Listener-ին միանալ փորձող համակարգներին: Երբ սպասելու ժամանակը (timeout) նշված է «–1», VI-ը կսպասի այնքան ժամանակ, մինչև որևէ համակարգ կապ հաստատի դրա հետ, որի հետ գործողությունից հետո համակարգը կրկին կսպասի հաջորդ կապի փորձին:

Այժմ դիտարկենք տվյալների փոխանցման՝ Նկ. 10.4–ում բերված պարզագույն համակարգի օրինակը՝ հիմնված TCP-ի վրա: Վերևի մասում պատկերված են հաղորդող, իսկ ներքևում՝ ընդունող համակարգի Block Diagram-ը (ձախից) և Front Panel-ը (աջից): Այստեղ դատագրամը 0-1 միջակայքում գեներացված պատահական թիվ է (DBL):



Նկ. 10.4

Հաղորդող համակարգի կառուցվածքը

① TCP Listen VI-ը հաստատում է լոկալ TCP կապ նշված «6340» պորտի հետ ու սպասում միևնույն պորտի հետ կապվել փորձող այլ համակարգի: Հենց որ ընդունող կողմից հաստատվում է կապ միևնույն «6340» պորտի հետ, VI-ը դուրս է գալիս սպասման վիճակից և սկսում ուղարկել դատագրամներ: Նկատենք, որ այստեղ սպասելու ժամանակը 30000 մվ է, որի ավարտին «6340» պորտին հաստատված կապեր չլինելու դեպքում VI-ը կվերադարձնի «56» կոդով սխալ (համապատասխանում է սպասելու ժամանակի ավարտին-timeout), և ծրագիրը կանգ կառնի:

② TCP Write VI-ը ուղարկում է դատագրամը նշված «6340» պորտին միացած համակարգին: Օգտագործվել է «String to IP» ֆունկցիան՝ տեքստային IP հասցեն թվայինի փոխակերպելու համար:

③ Ուղարկվող դատագրամը իրենից ներկայացնում է պատահական թիվ՝ փոխակերպված տեքստային հաղորդագրության «Type Cast» ֆունկցիայի միջոցով, և տվյալ տեքստային հաղորդագրության երկարությունը՝ կցված սկզբից: Ուղարկման պարբերությունը 100 մվ է:

④ TCP Close VI-ը փակում է հաղորդակցությունը:

Ընդունող համակարգի կառուցվածքն է.

① TCP Open Connection VI-ի միջոցով հաստատում է հաղորդակցության ուղի «6340» պորտի հետ, որին միացած է նաև ***Հաղորդող համակարգի*** TCP Listen VI-ը, որը սպասում է հաստատված կապերի:

② ***Հաղորդող համակարգի*** հետ կապ հաստատվելուց և դատագրամը ստանալուց հետո առաջին TCP Read VI-ը կարդում է դատագրամի սկզբի 4 բայթը, որը իրենից ներկայացնում է ինֆորմացիա դատագրամում ներառված բուն հաղորդագրության երկարության մասին: Այնուհետև «Type Cast» ֆունկցիայի միջոցով տեքստային տվյալը փոխակերպվում է I32 տիպի տվյալի:

③ I32 տիպի տվյալի փոխակերպված թիվը տրվում է երկրորդ TCP Read VI-ի «bytes to read» (կարդացվող բայթեր) մուտքին, որից հետո ստացված և DBL տիպի տվյալի փոխակերպված (վերոնշյալ «Type Cast» ֆունկցիայի միջոցով) հաղորդագրությունը իրենից ներկայացնում է հաղորդող համակարգի կողմից ուղարկված պատահական թիվը, որը և գրաֆիկորեն ցուցադրվում է Waveform Chart-ի վրա: Նշենք, որ այստեղ առաջին TCP Read VI-ի սպասելու ժամանակը 2000 մվ է, որը ընդհանուր դեպքում պետք է մեծ լինի հաղորդման պարբերությունից:

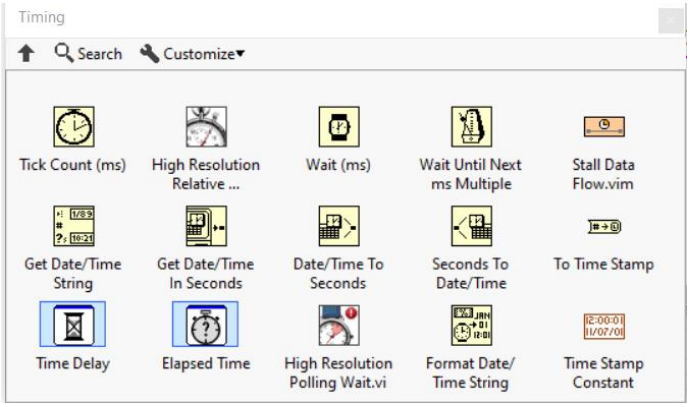
④ TCP Close VI-ը փակում է կապը:

Ամփոփելով, հարկ է նշել, որ, ընդհանուր առմամբ, UDP-ն ավելի արագ, սակայն ոչ հուսալի հաղորդակցության մեթոդ է: Միևնույն ժամանակ, TCP-ն ապահովում է ուղարկված տվյալների հաջորդական և հուսալի ստացում: Ի տարբերություն UDP-ի, որի աշխատանքի համար հակառակ (ստացող) կողմի հետ կապ և տվյալների ստացման հաստատում անհրաժեշտ չէ, TCP-ն աշխատում է միայն հակառակ կողմից կապի հաստատման դեպքում և չի կարող օգտագործվել տվյալների սփռման համար:

11. Համակարգչի ժամացույցի, մկնիկի և ստեղնաշարի տվյալների ներմուծում (Data types and Timing)

Ժամացույց

Ժամանակի VI-ները և ֆունկցիաները ընդհանուր դեպքում օգտագործվում են պրոցեսների իրականացման արագությունը ղեկավարելու, ինչպես նաև ժամանակի և ամսաթվի մասին ինֆորմացիա դուրս բերելու համար: Ժամանակի VI-ները գտնվում են Block Diagram-ի Timing բաժնում: Նկ. 11.1-ում ներկայացված են Ժամանակի VI-ները և ֆունկցիաները, ինչպես նաև դրանցից ամենահիմնականների բացատրությունները:



Նկ. 11.1. Ժամանակի VI-ները

High Resolution Relative Seconds VI-ը ելքում տալիս է տվյալ պահի հարաբերական ժամանակը՝ արտահայտված վայրկյաններով: Օգտագործելով երկու հաջորդական ժամանակային արժեքների տարբերությունը՝ մեծ ճշտությամբ կարելի է որոշել պրոցեսների տևողությունը: Նույն ֆունկցիան է կատարում նաև Tick Count (ms) VI-ը, սակայն ավելի փոքր ճշտությամբ:

Wait (ms) և Wait Until Next ms Multiple ֆունկցիաները կարող են առաջին հայացքից նույնական թվալ, քանի որ երկուսն էլ նախատեսված են հաջորդական պրոցեսների (օրինակ՝ ցիկլի իտերացիաների) միջև սպասելու ժամանակը ղեկավարելու համար: Դրանք երկուսն էլ օգտագործում են օպերացիոն համակարգի միլիվայրկյանային ժամացույցը

որպես հիմք՝ հաշվարկելու համար սպասելու ժամանակը: Սակայն այս ֆունկցիաների միջև առկա է կոնցեպտուալ տարբերություն: Wait Until Next ms Multiple ֆունկցիան համաժամանակացնում է սպասելու ժամանակը օպերացիոն համակարգի ներքին ժամանակի հետ, ինչի արդյունքում հաջորդական պրոցեսների միջև սպասելու ժամանակը առաջին ինտերացիայի ժամանակ կարող է տարբերվել Wait Until Next ms Multiple-ի մուտքային ժամանակից, մինչև որ այն համափուլ կաշխատի օպերացիոն համակարգի ժամանակի հետ: Wait (ms) ֆունկցիան ապահովում է սպասելու ժամանակը ճիշտ այնքան, ինչքան որ տրված է դրա մուտքին, և օպերացիոն համակարգի ժամանակի հետ համաժամանակացված չէ:

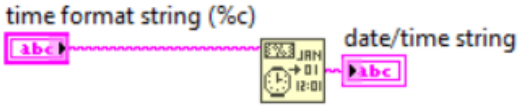
Նկ. 11.1-ում ցույց տրված մի քանի ֆունցիաների նկարագրությունները:

Get Date/Time String ֆունկցիան առանձին ելքերով (տեքստային տիպի տվյալ) վերադարձնում է տվյալ պահի ժամանակը և ամսաթիվը՝ կարճ, երկար կամ հապավումներով ֆորմատով, կամ փոխակերպում է ժամանակային կնիքը (timestamp) ժամի և ամսաթվի:

Get Date/Time in Seconds ֆունկցիան վերադարձնում է տվյալ պահի ժամանակային կնիքը (ժամը միլիվայրկյանների ճշտությամբ և ամսաթիվը):

Format Date/Time String ֆունկցիան փոխակերպում է ժամանակը և ամսաթիվը՝ ըստ նշված տեքստային ֆորմատի:

Ֆորմատի մի քանի կարևոր կոդեր և օրինակներ ցույց են տրված ստորև:

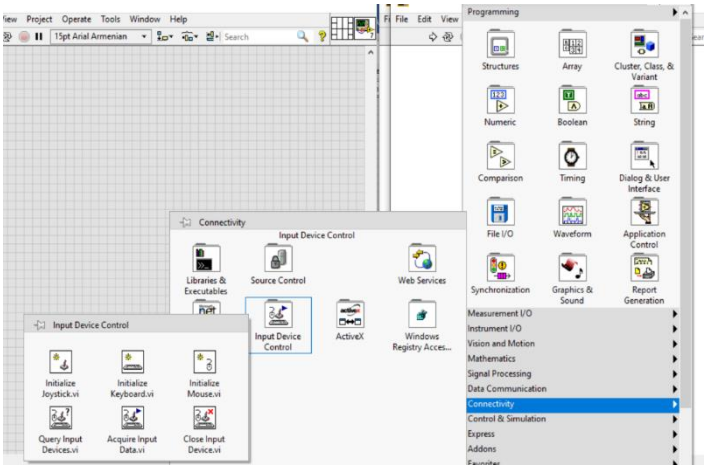


Time format string (%c)	Date/time string
%d-%a-%Y, %H:%M:%S%5u	04-Sun-2022, 14:19:07.79636
%Z	Caucasus Standard Time
Day of Year is %j	Day of Year is 338

Կոդ	Նշանակություն		Կոդ	Նշանակություն
%a	շաբաթվա օրը՝ կրճատ (Sun)		%m	ամիսը թվերով (0-12)
%A	շաբաթվա օրը՝ երկար (Sunday)		%M	րոպեն (0-59)
%d	ամսվա օրը (01–31)		%S	վայրկյանը (0-59)
%H	ժամը՝ 24-ժամյա ֆորմատով		%«X»u	մնացորդային վ-ները, X=1,2,..
%I	ժամը՝ 12-ժամյա ֆորմատով		%Y	տարին՝ ներառյալ հարյուրամյակը
%j	տարվա օրը (001–366)		%Z	ժամային գոտին

Ստեղնաշար

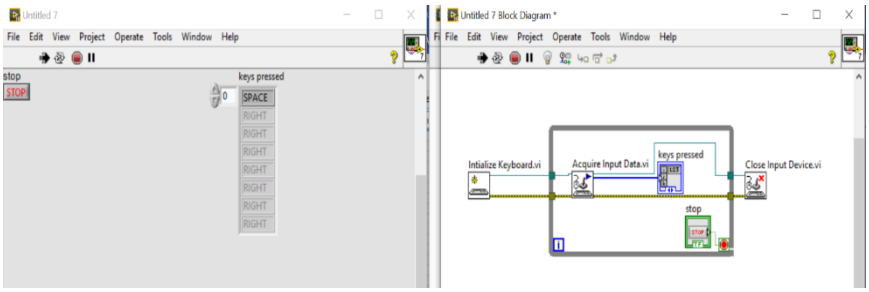
Համակարգչի ստեղնաշարի և մկնիկի հետ կապ հաստատելու համար Block Diagram-ի ֆունկցիաներից պետք է ընտրել Connectivity բաժնի Input Device Control ենթաբաժինը (Տե՛ս Նկ. 11.2):



Նկ. 11.2

Ստեղնաշարի տվյալների ներմուծման/օգտագործման ծրագրի պարզագույն օրինակը ներկայացված է Նկ. 11.3-ում: Նախ պետք է հայտարարագրել ստեղնաշարը ցիկլից դուրս (Initialize Keyboard.vi), որը

ստեղծում է համապատասխան կապ ստեղնաշարի և *LabVIEW*-ի միջև: Տվյալների ներմուծումը իրականացվում է Acquire Data.vi-ի միջոցով՝ ցիկլի ներսում, քանի որ ստեղների մասին տվյալները անհրաժեշտ է ցուցադրել դինամիկ կերպով, իրական ժամանակում: Ստեղնաշարի և *LabVIEW*-ի միջև հղումը փակելու համար անհրաժեշտ է օգտագործել Close Input Device.vi-ը, որը տեղադրվում է ցիկլից դուրս: Այստեղ key pressed զանգվածից երևում է, որ սեղմված է համակարգչի ստեղնաշարի բացատ (Space) ստեղնը: Եթե զուգահեռ սեղմենք մեկից ավելի ստեղներ,

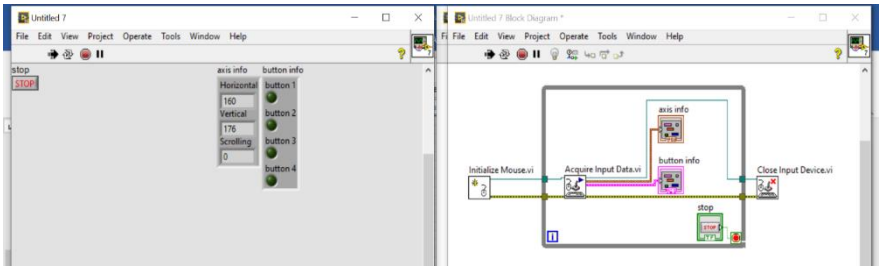


Նկ. 11.3

ապա զանգվածի էլեմենտները համապատասխանաբար կավելանան:

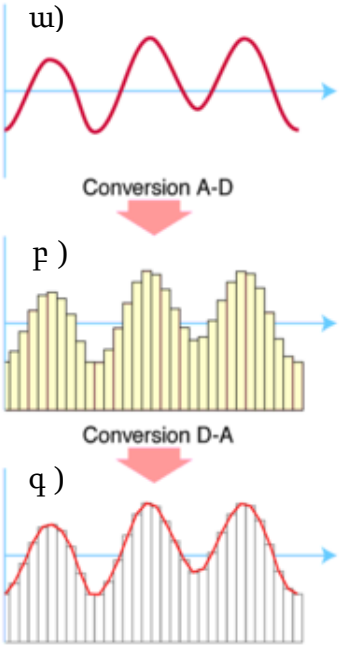
Մկնիկ

Փոխարինելով ստեղնաշարի ծրագրում Initialize Keyboard.vi-ը Initialize Mouse.vi-ով և միացնելով համապատասխան ինդիկատորները՝ ծրագիրը կունենա Նկ. 11.4-ում ներկայացված տեսքը: Ինչպես երևում է, Front Panel-ում ներմուծվել են մկնիկի ընթացիկ կոորդինատների (պիքսելներ) և կոճակների տվյալները:



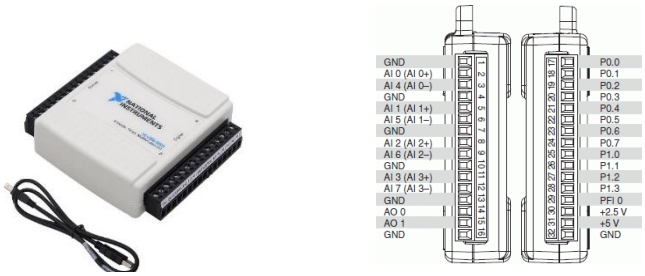
Նկ. 11.4

12. Թվային-անալոգային փոխակերպիչներ (Data Acquisition DAQ)



Նկ. 12.1

Ժամանակակից տեխնոլոգիաներում լայն կիրառություններ ունեն անալոգ–թիվ և թիվ-անալոգ փոխակերպիչները: Անալոգաթվային փոխակերպման սկզբունքը սխեմատիկորեն ներկայացված է Նկ.12.1-ում: Իրական ազդանշանների ներմուծում/արտածումը *LabVIEW* միջավայր կարելի է իրականացնել Data Acquisition (DAQ) սարքերի միջոցով, որոնց կարևորագույն հանգույցներից են անալոգ-թիվ և թիվ–անալոգ փոխակերպիչները: Սրանց կարևորագույն հատկություններն են զգայունությունը, արագագործությունը (Sample rate), բիթայնությունը, դինամիկ լայն տիրույթը, մուտք-ելքերի քանակը և այլն: Տվյալների ներմուծումն ու արտածումը որևէ համակարգ իրականացվում է հենց DAQ-երի օգնությամբ: Ինչպես բազմաթիվ արտադրողներ, այնպես էլ National Instrument-ը ունի տարբեր տեսակի անալոգաթվային ձևափոխիչներ: Ներկայացնենք դրանցից պարզագույններից մեկը՝ NI USB 6009 Նկ. 12.2: Մուտքային լուծունակությունը 14 բիթ է, իսկ առավելագույն արագությունը՝ 48 kS/s:

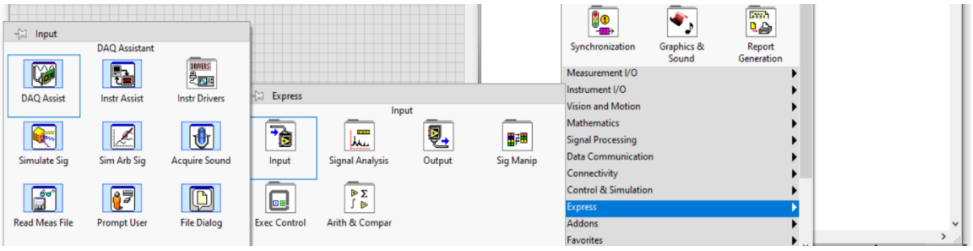


Նկ. 12.2

Նկ. 12.2-ի աջ կողմում պատկերված է մուտքերի և ելքերի նկարագրությունը: 1-16 պորտերը նախատեսված են անալոգային ազդանշանների համար, իսկ 17-32 պորտերը՝ թվայինի:

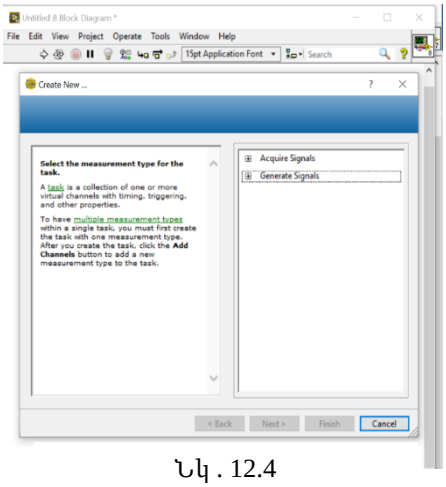
Ինչպես նկատելի է, անալոգային մուտքերն ու ելքերը ֆիքսված են, իսկ թվայինի դեպքում կարելի է ծրագրային փոփոխել: Թվային մուտքերի և ելքերի առավելագույն լարման արժեքներն են 5վ (False=0 True=5վ), իսկ անալոգայինինը՝ մուտք՝ +-10վ, ելք՝ 0-5վ: Չևափոխիչների պարամետրերին առավել մանրամասն ծանոթանալու համար կարելի է այցելել www.ni.com կայքը:

NI USB 6009-ի համապատասխան կարգավորումները (Driver) իրականացնելուց հետո այն միացվում է համակարգչին և Block Diagram-ի ֆունկցիոնալ վահանակից ընտրվում Express\Input\ DAQ Assistant.vi կամ Measurement I/O\NI DAQmx\DAQ Assistant (Տե՛ս Նկ. 12.3):

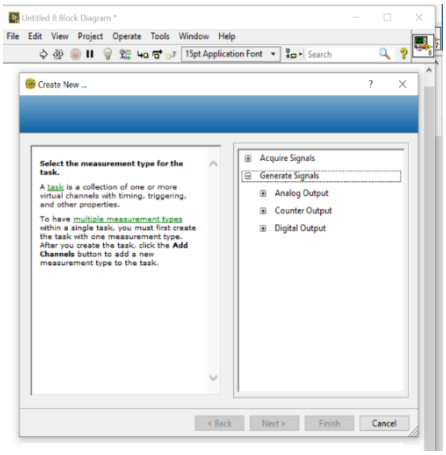


Նկ. 12.3

Տեղադրելով DAQ Assistant-ը՝ կբացվի Նկ. 12.4-ում պատկերված պատուհանը, որտեղ կա ընտրության երկու հնարավորություն՝ Acquire Signals (ստանալ ազդանշան) և Generate Signals (գեներացնել ազդանշան): Generate Signals-ը ընտրելու դեպքում կբացվի անալոգային և թվային ելքերի ընտրության հնարավորություն (Նկ. 12.5):

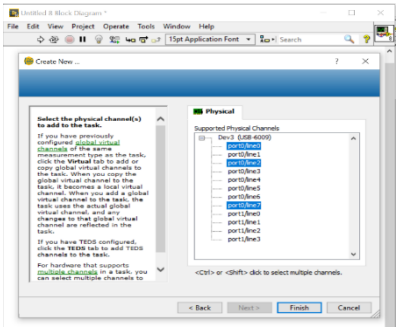


Նկ . 12.4

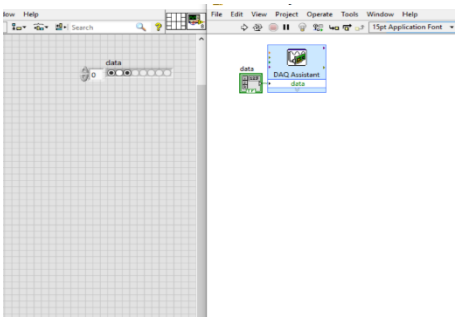


Նկ . 12.5

Բնականաբար, Acquire Signals-ի դեպքում մուտքերի համար կունենանք նույն ֆունկցիոնալ հնարավորությունները: Ընտրելով Digital Output և Line Output՝ կցուցադրվեն բոլոր թվային ելքերը, և այստեղ կարող ենք ընտրել ցանկացած թվային ելք՝ <Ctrl>-ի և <Shift>-ի օգնությամբ (Նկ. 12.6): Սեղմելով Finish, հետո Ok և տեղադրելով Create Control՝ կունենանք ղեկավարվող զանգված (Նկ. 12.7): Այստեղ անհրաժեշտ է, որպեսզի զանգվածի էլեմենտների քանակը հավասար լինի հայտարարագրված պորտերի քանակին. բերված օրինակում այն 3 հատ է (Տե՛ս Նկ. 12.6):



Նկ . 12.6



Նկ . 12.7

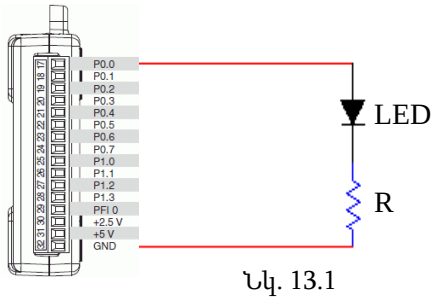
Նույն սկզբունքով կարելի է աշխատել անալոգային ելքերի հետ: Հետագայում լաբորատոր աշխատանքներում ցույց կտրվեն անալոգային և թվային մուտքերի հետ աշխատելու հմտությունները:

13. Լաբորատոր աշխատանք 1

Լուսադիոդների (LED) ղեկավարում

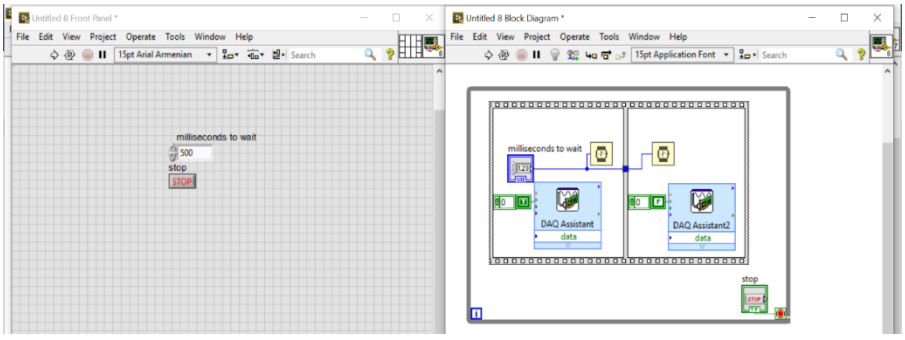
Լաբորատոր աշխատանքում ցույց կտրվի NI USB 6009 անալոգա-թվային փոխակերպիչի միջոցով լուսադիոդների ղեկավարումը: Լուսա-դիոդի ղեկավարումը կարելի է իրականացնել և՛ անալոգային, և՛ թվային ելքերի միջոցով:

<u>Թվային ղեկավարում</u>



Նկ. 13.1

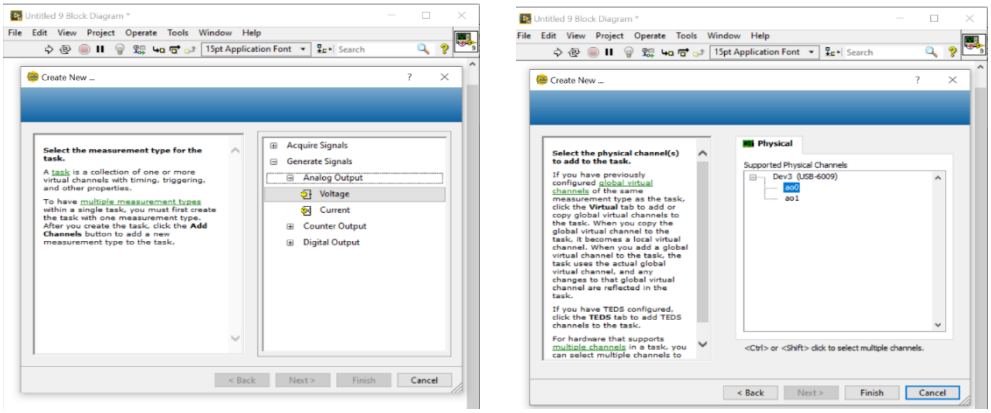
Քանի որ փոքր հզորության լու-սադիոդների աշխատանքային լարու-մը սովորաբար ընկած է U=2-3.5վ (I=10-40 մԱ) տիրույթում, իսկ թվային ելքերի լարման մաքսիմալ արժեքը 5վ է, ապա անհրաժեշտ է լուսադիոդին հաջորդաբար միացնել ~100 Օմ դի-մադրություն (Նկ. 13.1): Լուսադիոդը DAQ-ի (NI USB 6009) P0.0 պորտին միացնելուց հետո անհրաժեշտ է հայտարարագրել որպես ելք և ծրագրային ընտրել նույն պորտը (Տե՛ս Նկ. 12.4-12.7): Front Panel-ում բուլյան զանգվածի էլեմենտի True արժեքի դեպքում լուսադիոդը կմիանա, իսկ False-ի դեպքում կանջատվի: Այժմ կառուցենք մի ծրագիր, որի միջոցով լուսադիոդը կթարթի 500 մվ տևո-ղությամբ (Տե՛ս Նկ. 13.2): Այստեղ ժամանակի հապաղումն իրականաց-նելու համար ընտրված է Timing բաժնի Wait(ms) գործիքը:



Նկ. 13.2

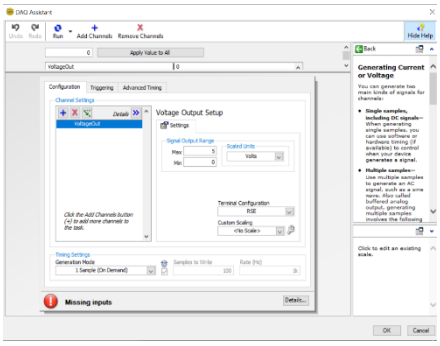
Անալոգային ղեկավարում

Անալոգային ելքով ղեկավարման համար կարելի է ընտրել DAQ Assistant\Generate Signals\Analog Output\Voltage և ao1 պորտը (Նկ. 13.3):

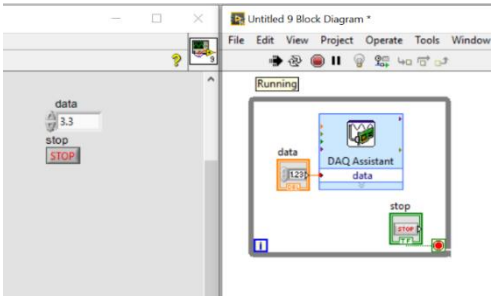


Նկ . 13.3

Սեղմելով Finish կոճակը՝ կբացվի Նկ. 13.4-ում պատկերված պատուհանը: Ծրագրային ղեկավարմամբ անալոգային ելքի լարումը կարող ենք սահուն փոփոխել 0-5 Վ միջակայքում: Այստեղ Max և Min արժեքների միջոցով կարող ենք իրականացնել լարման սահմանափակումներ, հետևաբար շղթայից կարելի է հանել լուսադիոդին հաջորդաբար միացված դիմադրությունը: OK կոճակը սեղմելուց հետո data մուտքին միացնում ենք Numeric Control-ը: Տեղադրելով համակարգը ցիկլի մեջ՝ կստանանք Նկ. 13.5-ում պատկերված տեսքը: Փոփոխելով data ինդիկատորի լարման արժեքը՝ կտեսնենք լուսադիոդի ինտենսիվության փոփոխություն:



Նկ. 13.4



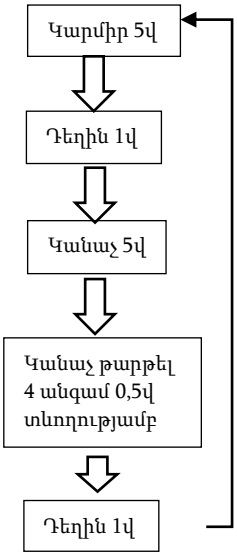
Նկ. 13.5

14. Լաբորատոր աշխատանք 2

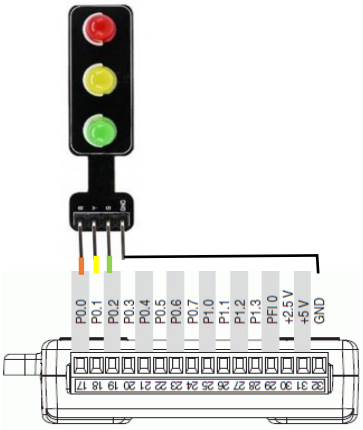
Լուսադիոդներով լուսացույցի ծրագրավորում

Լուսացույցի աշխատանքի սկզբունքին ծանոթ են բոլորը: Ծրագրավորենք ըստ Նկ. 14.1-ում ներկայացված բլոկ սխեմայի աշխատող լուսացույցը:

Դիցուք՝ ունենք Traffic Light LED Module - 5V - 56x21x11mm տիպի լուսացույցը: Նկ. 14.2-ում բերված է միացման սխեման:

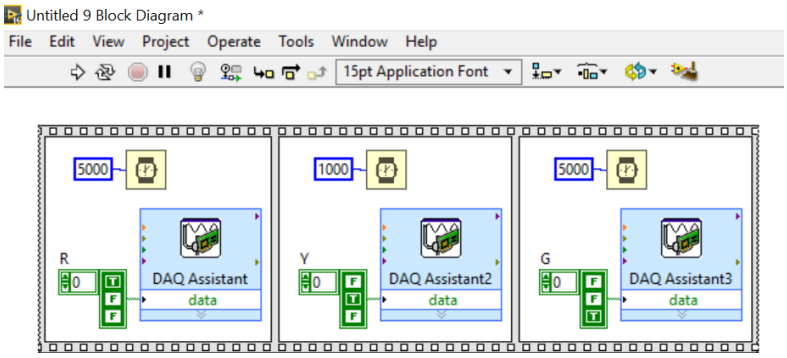


Նկ. 14.1



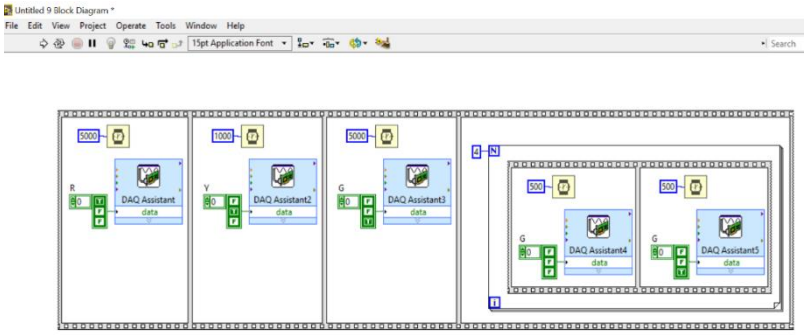
Նկ. 14.2

DAQ Assistant-ում հայտարարագրենք համապատասխան թվային ելքերը և միացնենք հաստատուն բույլան զանգվածին: Զանգվածում ակտի-



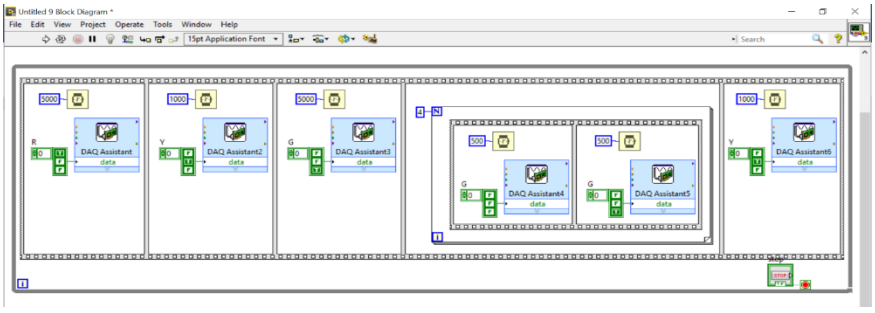
Նկ. 14.3

վացնելով 3 տարրեր՝ կունենանք համապատասխանաբար կարմիր, դեղին և կանաչ գույնի լույսերը: Flat Sequence-ի մեջ տեղադրելով ժամանակային հապաղումները՝ կստանանք կարմիր, դեղին և կանաչ գույների հերթականությունը (Նկ. 14.3): Այնուհետև ավելացնենք թարթման բլոկը՝ կրկին օգտագործելով Flat Sequence-ը, որի առաջին էջում անջատված վիճակն է, իսկ երկրորդում՝ միացված: Չորս անգամ գործողությունը կրկնելու համար այն տեղադրենք For Loop ցիկլի մեջ (Նկ. 14.4):



Նկ. 14.4

Վերջում ավելացնենք կրկին դեղին գույնը և տեղադրենք այն While Loop ցիկլի մեջ: Վերջնական ծրագիրը կունենա Նկ. 14.5-ում պատկերված տեսքը:



Նկ. 14.5

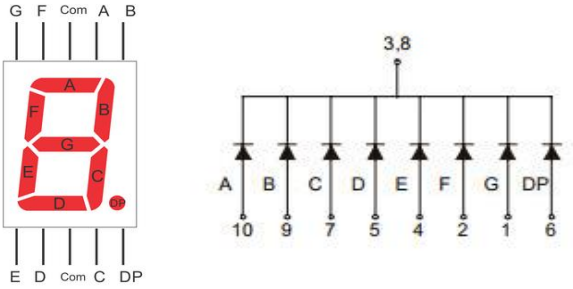
15. Լաբորատոր աշխատանք 3

7-սեգմենտանի 1 և 4-նիշանի LED դիսփլեյների ղեկավարում

1-նիշանի 7-սեգմենտանի LED դիսփլեյի ղեկավարումը

(1 digit 7 segment led display)

Նկ. 15.1-ի ձախ մասում բերված է դիսփլեյի ֆիզիկական, իսկ աջ մասում՝ սխեմատիկ տեսքը: A-G՝ յոթ սեղմակները նախատեսված են LED-երի, DP սեղմակը՝ կետի, և 3,8 (Com) ընդհանուր սեղմակը՝ հողանցման համար:

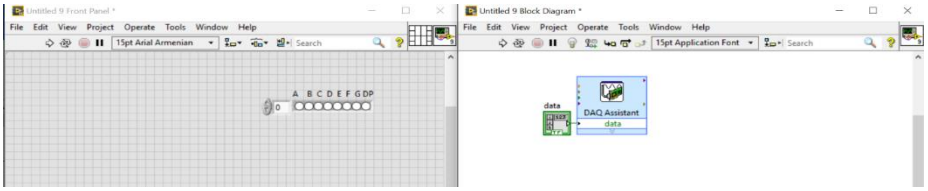


Նկ. 15.1

Այստեղ A, B, C, D, E, F, G, DP կետերը պետք է միացնել NI USB 6009 անալոգաթվային փոխակերպիչի P00, P01,…, P07 պորտերին, իսկ 3 կամ 8-ը՝ GND-ին: Ծրագրով ընտրելով համապատասխան պորտերը և ակտիվացնելով զանգվածում նույն քանակով տարրեր (Տե՛ս Նկ. 12.4-12.7)՝ կստանանք Նկ. 15.2-ում պատկերված տեսքը: Յուրաքանչյուր նիշ պատկերելու համար անհրաժեշտ է զանգվածում ակտիվացնել A-G համապատասխան տարրերը, օրինակ՝ 1(BC), 2(ABGED) և այլն:

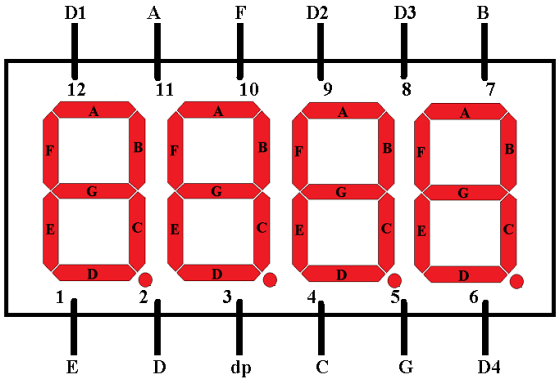
4-նիշանի 7-սեգմենտանի LED դիսփլեյի ղեկավարումը

(4 digit 7 segment led display)



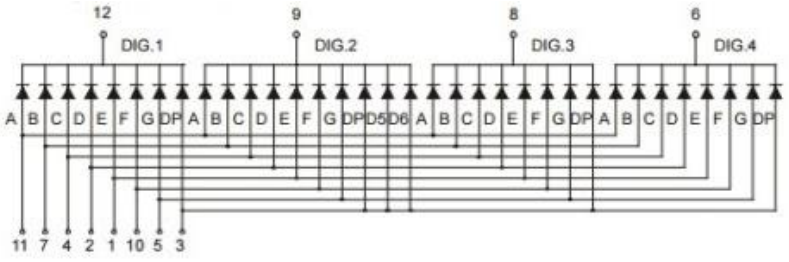
Նկ. 15.2

Նկ. 15.3-ում պատկերված է 4-նիշանի 7-սեգմենտանի LED դիսփլեյի ֆիզիկական, իսկ Նկ. 15.4-ում՝ սխեմատիկ տեսքը: Ինչպես երևում է,



Նկ. 15.3

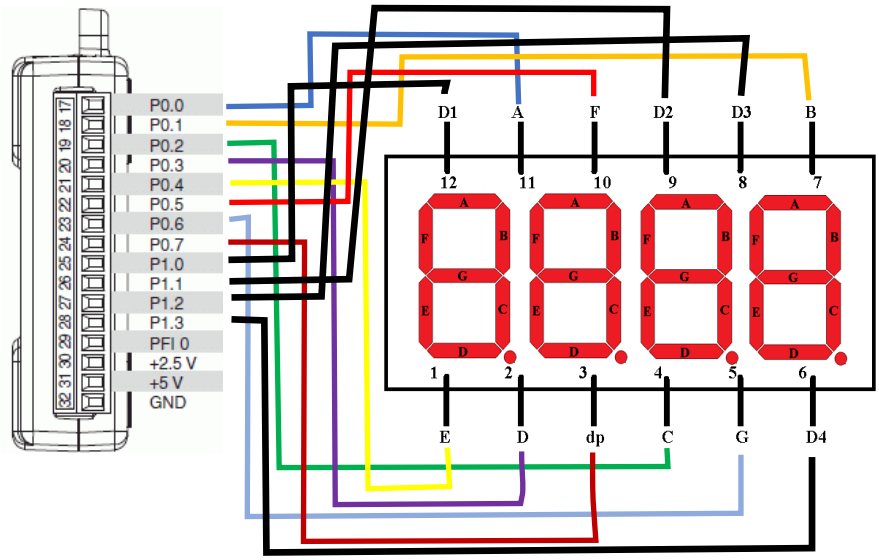
այստեղ բոլոր նիշերի A, B, C, D, E, F, G, DP կետերը միացված են իրար, բացառությամբ ընդհանուր կետերի՝ D1, D2, D3, D4 (Նկ. 15.4): Քանի որ ցանկացած նիշ միացնելու համար անհրաժեշտ է ղեկավարել ընդհանուր կետերը, իսկ դա ծրագրով հնարավոր չէ իրականացնել, ապա ընդհանուր կետերը նույնպես պետք է միացնել թվային պորտերին:



Նկ. 15.4

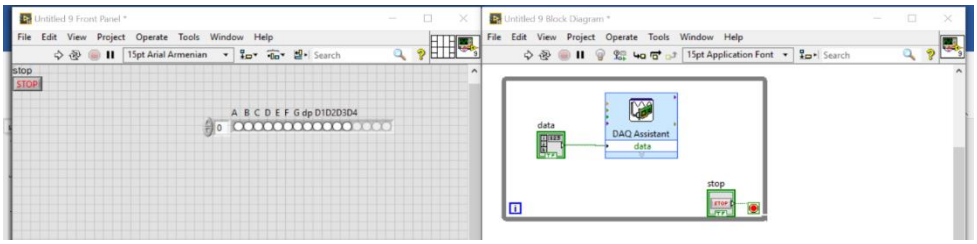
Կունենանք 0Վ (False), կմիանա նիշը, իսկ 5Վ-ը (True) կանջատվի, քանի որ պոտենցիալների տարբերությունը կլինի 0: Սխեմայից նաև երևում է, որ հնարավոր չէ վահանակի վրա պատկերել միաժամանակ երկու կամ ավելի տարբեր թվեր: Խնդիրը լուծելու համար անհրաժեշտ է, որ մեկ թվից մյուսին անցումը տեղի ունենա շատ արագ, այնպես, որ առաջանա տեսողական խաբկանք (աչքը չհասցնի թարթումը նկատել):

Կատարենք համապատասխան միացումները անալոգաթվային փոխակերպիչին և ծրագրավորենք այն:



Նկ . 15.5

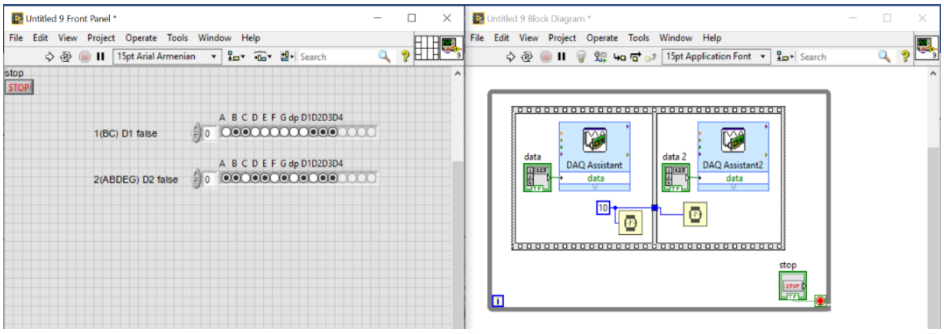
Նկ. 15.5-ում պատկերված են LED դիսպլեյի միացումները NI USB 6009 սարքին: Սև գույնով միացված են ընդհանուր կետերը, իսկ գունավորներով՝ A-G-ն և dp-ն: DAQ Assistant-ով ընտրենք 12 թվային ելքերով պորտեր և ավելացնելով զանգվածի համապատասխան քանակով էլեմենտները՝ կստանանք Նկ. 15.6–ում պատկերված տեսքը:



Նկ. 15.6

Ծրագրավորենք այնպես, որ երկու նիշերի վրա լինեն երկու տարբեր թվեր՝ 1 և 2 (Նկ. 15.7): Front Panel-ում երկու թվերի համար ունենք տարբեր զանգվածներ, որտեղ, ըստ տառերի դասավորվածության, կունենանք

համապատասխան թվերը, իսկ ընդհանուր կետերի համար՝ False միացված, True անջատված:



Նկ. 15.7

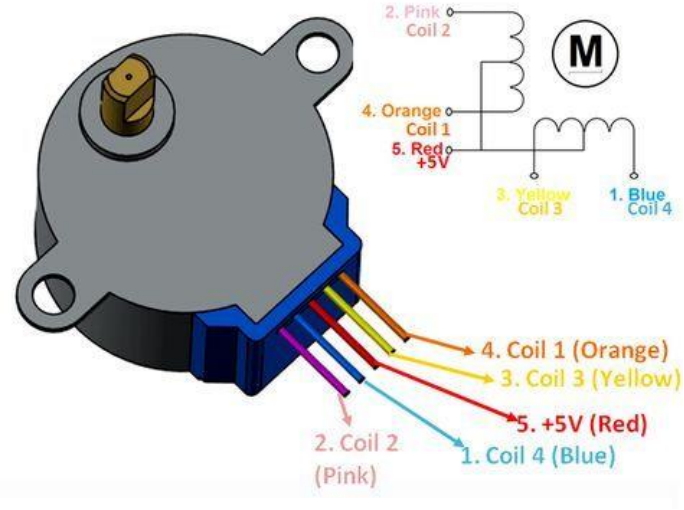
Առաջադրանք 1 - Ծրագրավորել այնպես, որ վահանակը ծառայի որպես ժամացույց՝ տվյալները վերցնելով համակարգչի ժամացույցից:

Առաջադրանք 2 – Ծրագրավորել վայրկյանաչափ:

16. Լաբորատոր աշխատանք 4

Քայլային շարժիչի ղեկավարում

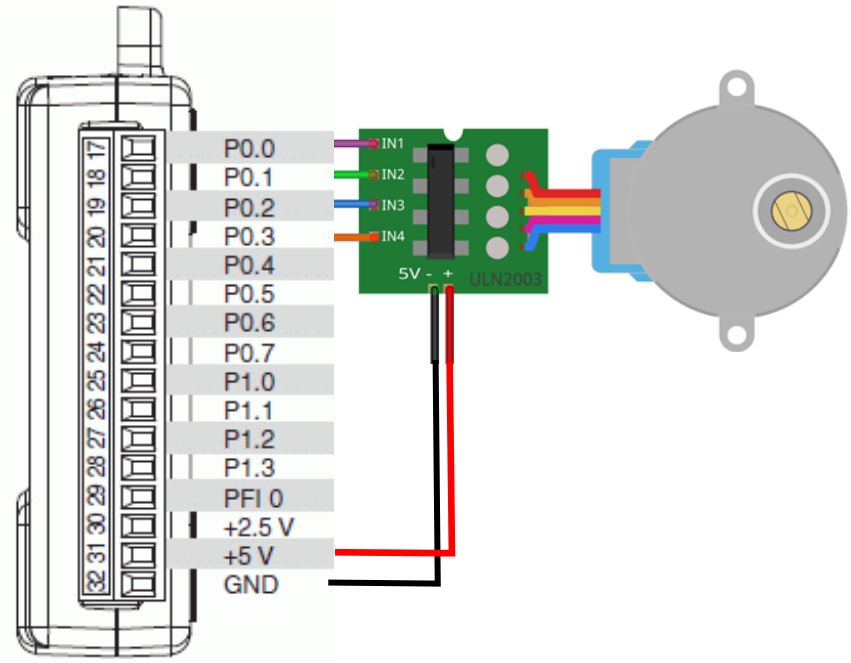
Ներկայումս քայլային շարժիչները լայն կիրառություն են գտել տարբեր ոլորտներում, ինչպիսիք են՝ մեքենաշինությունը, ավիաշինությունը, բժշկությունը և այլն: Քայլային շարժիչները սովորական շարժիչներից տարբերվում են նրանով, որ ոչ միայն հնարավոր է ղեկավարել պտտման արագությունը, այլ նաև պտտման անկյունը, այսինքն՝ ռոտորը որոշակի ճշտությամբ (քայլով) կարելի է ֆիքսել կամայական անկյան վրա: Նշված շարժիչների կարևորագույն պարամետրերից մեկը քայլի չափն է կամ նվազագույն անկյունը: NI USB 6009-ի թվային ելքերի միջոցով ղեկավարենք հետևյալ stepper motor 28byj-48 քայլային շարժիչը:



Նկ. 16.1

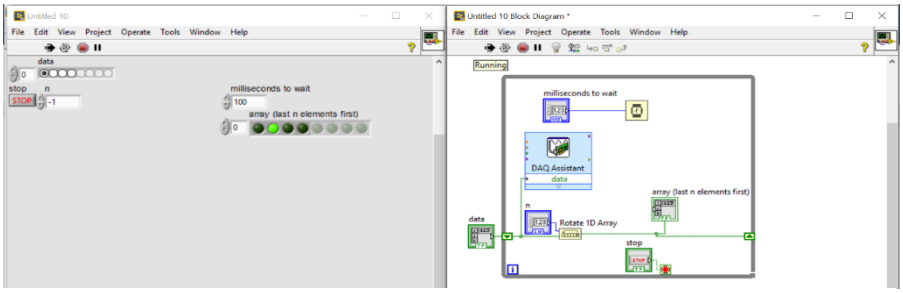
Նկ. 16.1-ում պատկերված է շարժիչի ֆիզիկական և սխեմատիկ տեսքը: Այն բաղկացած է չորս փաթույթներից և նշված թվերի հերթականությամբ ապահովում է պտույտը: Շարժիչը ղեկավարելու համար անհրաժեշտ է ուժային բլոկի չորս լարերը միացնել փոխակերպիչի թվային ելքերին, իսկ +5վ-ը՝ սնուցման աղբյուրին (Նկ. 16.2):

DAQ Assistant-ի միջոցով ընտրում ենք չորս թվային պորտերը՝ որպես ելքեր, այնուհետև, ավելացնելով նույն քանակով էլեմենտներով զանգվածը, կստանանք Նկ. 16.3-ում պատկերված տեսքը:



Նկ. 16.2

Ցիկլից դուրս data զանգվածի ցանկացած էլեմենտի վերագրենք True արժեք և Shift Register-ի միջոցով ներմուծենք ցիկլ: Այստեղ էլեմենտի փոխատեղում ստանալու համար օգտագործվում է Array բաժնից Rotate 1D Array գործիքը, իսկ պտույտ ստանալու համար Shift Register-ը: Արդյունքում ընտրելով n-ի 1 կամ -1 արժեքները՝ կունենանք LED-ի True արժեքի տեղափոխում դեպի ձախ կամ աջ, ինչը կապահովի շարժիչի աջ ու ձախ պտույտը, իսկ 0-ն կլինի դադարի ռեժիմը: Արագությունը ղեկավարվում է հապաղման ժամանակի օգնությամբ:

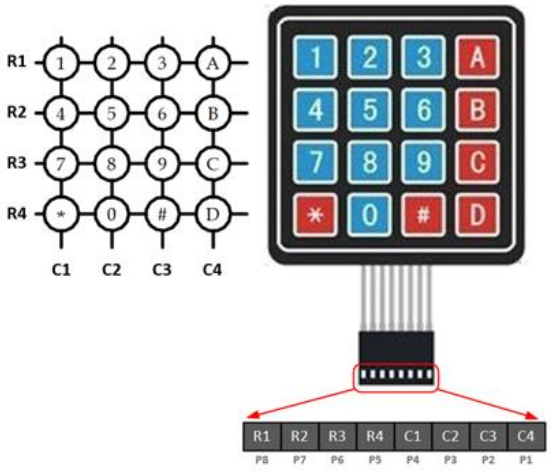


Նկ. 16.3

17. Լաբորատոր աշխատանք 5

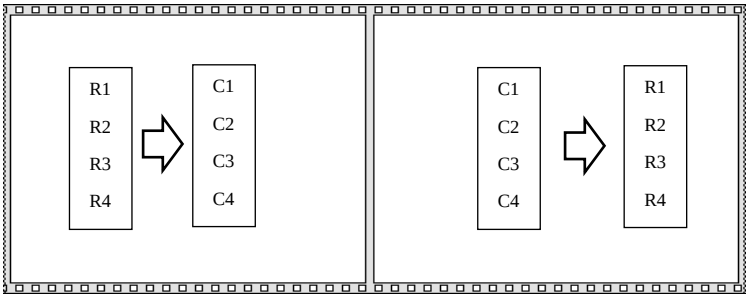
Touchpad զանգվածի ծրագրավորում

Նկ. 17.1-ի աջ մասում պատկերված է սենսորային վահանակի (Touchpad) տեսքը, իսկ ձախ մասում՝ դրա սխեմատիկ տեսքը, որտեղ յուրաքանչյուր կոճակ սեղմելով՝ միացնում է համապատասխան սյունը



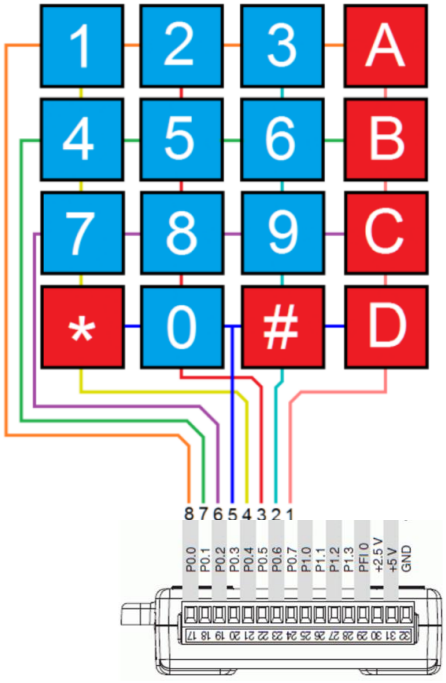
Նկ. 17.1

տողին: Օրինակ՝ եթե սեղմենք 6-ը, իրար կմիացնի R2 տողը և C3 սյունը: Այսպիսով, ալգորիթմը ունի Նկ. 17.2-ում պատկերված տեսքը: Սկզբում Ri-ն ուղարկում է True արժեքներ, իսկ Ci-ն՝ ստանում այն, հետո Ci-ն է ուղարկում, Ri-ն՝ ընդունում:



Նկ. 17.2

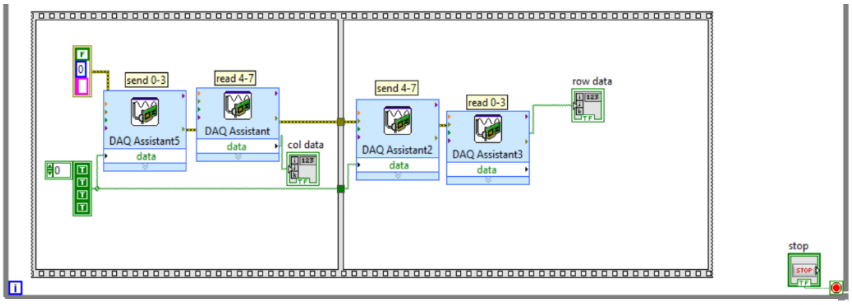
Նկ. 17.3-ում պատկերված է Touchpad-ի միացման տեսքը անալոգ-թվային փոխակերպիչին, որտեղ Touchpad-ի ութ լարերը միացված են փոխակերպիչի թվային պորտերին: Ri-ն կլինի P0.0-P0.3, իսկ Ci-ն՝ P0.4-P0.7:



Նկ. 17.3

DAQ Assistant-ով Ri-ի համար անհրաժեշտ է ընտրել P0.0-P0.3 չորս թվային պորտերը՝ որպես ելքեր (նշելով բոլոր ելքերը invert line), իսկ Ci-ի համար՝ P0.4-P0.7 պորտերը՝ որպես մուտքեր (նշելով բոլոր մուտքերը invert line): Այնուհետև ընտրելով Ci-ն որպես ելք, իսկ Ri-ն որպես մուտք և զանգվածում ավելացնելով համապատասխան քանակով տարրեր՝ ծրագիրը կունենա Նկ. 17.4-ում ցույց տրված տեսքը:

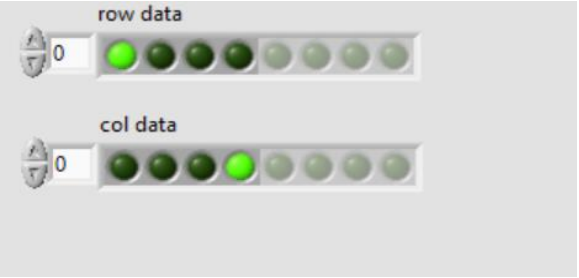
Արդյունքում ունենում ենք երկու զանգվածներ՝ row data և col data, որոնք ցույց են տալիս



Նկ. 17.4

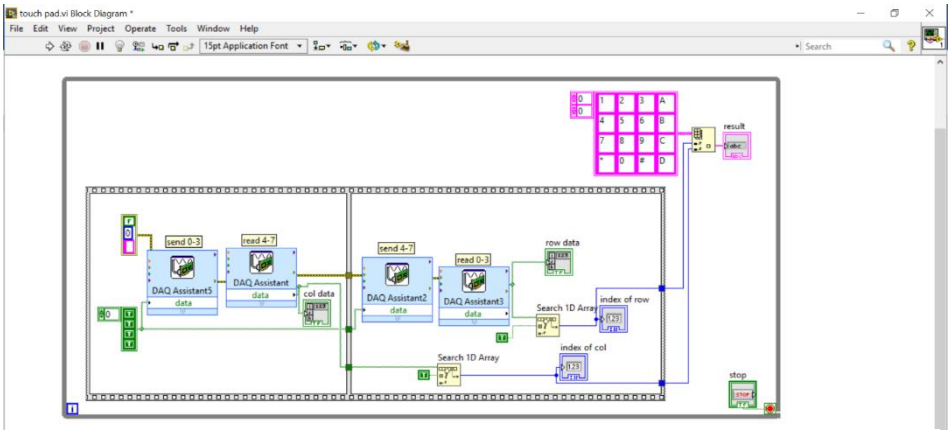
սեղմված կոճակի կոորդինատները: Օրինակ՝ եթե Touchpad-ի վրա սեղմենք A կոճակը, կունենանք Նկ. 17.5-ի տեսքը՝ I տող IV սյուն: Զանգ-

վածների բաժնից ընտրելով Search 1D Array գործիքը՝ կգտնենք ընթացիկ True արժեքի ինդեքսը (կոորդինատը):



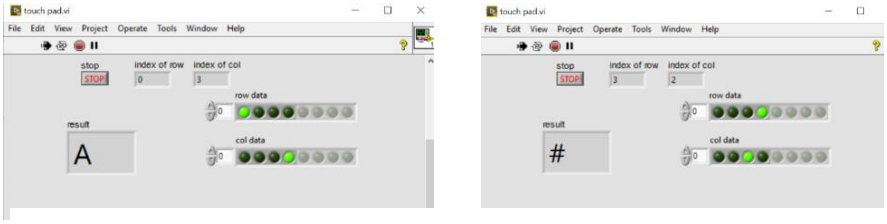
Նկ. 17.5

Այնուհետև, ավելացնելով տեքստային տարրերով երկչափ զանգված, վերջնական ծրագիրը, կունենա Նկ. 17.6-ում պատկերված տեսքը:



Նկ. 17.6

Նկ. 17.7-ում պատկերված են համապատասխան կոճակների սեղմման դեպքում ստացված արժեքները:

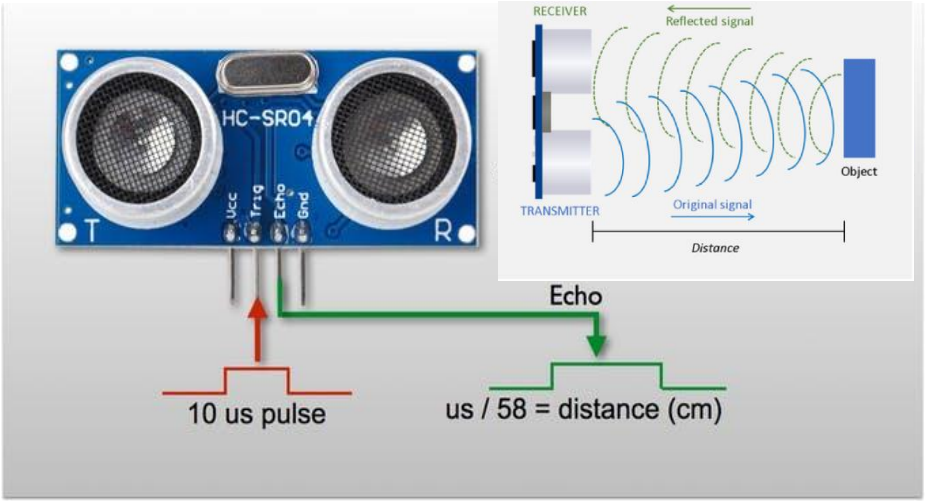


Նկ. 17.7

18. Լաբորատոր աշխատանք 6

Ուլտրաձայնային տվիչի ծրագրավորում (HC-SR04)

Ավտոմատացված համակարգերը առանց տվիչների նախագծելը բավական բարդ է, որոշ դեպքերում՝ նույնիսկ անհնար, քանի որ նրանք համակարգի տեխնիկական զգայարաններն են: Հեռավորության չափման համար ներկայումս կան տարբեր եղանակներով աշխատող տվիչներ, որոնցից հաճախակի հանդիպում են ուլտրաձայնային տվիչները: Ուլտրաձայնային տվիչները նախատեսված են փոքր հեռավորություններ չափելու համար: Համակարգի աշխատանքը հիմնված է էխոլոկացիայի սկզբունքի հիման վրա. այն ուղարկում է ուլտրաձայնային ազդանշան և ընդունում է արձագանքը: Չափելով ուղարկած և ստացած ազդանշանների միջև ընկած ժամանակահատվածը՝ որոշում է տվիչի և ազդանշանի տարածմանը խոչընդոտող առարկայի միջև եղած հեռավորությունը:



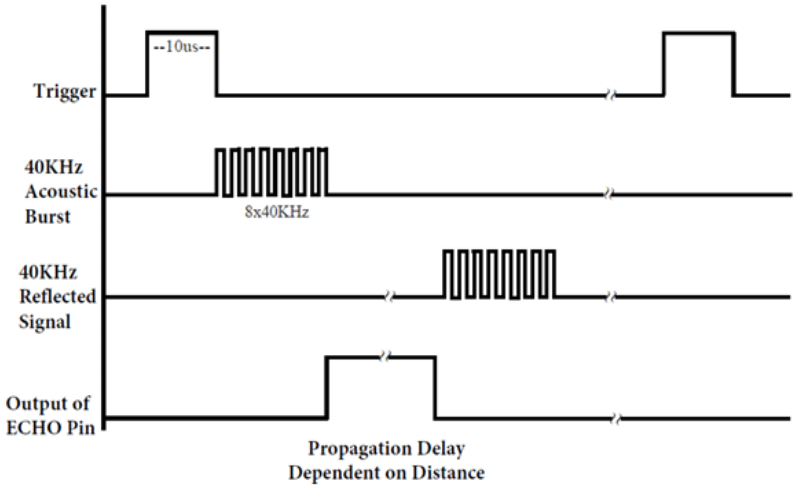
Նկ. 18.1

Այս լաբորատոր աշխատանքում *LabVIEW* միջավայրում ծրագրավորենք HC-SR04 մոդելի ուլտրաձայնային տվիչը (Տե՛ս Նկ. 18.1):

Համակարգն ունի միացման 4 կոնտակտներ (PIN), որոնցից Vss-ն և GND-ն սնուցման համար են, իսկ Trig-ը և Echo-ն՝ ազդանշանի ուղարկման և ընդունման համար: Trig PIN-ին տալով 10մկվ տևողությամբ (+5վ) ազդանշան՝ համակարգը գեներացնում է 40ԿՀց հաճա-

խությամբ 8 ազդանշան, իսկ անդրադարձած ազդանշանը ստանալուց հետո Echo PIN-ի վրա ունենում ենք անցած ճանապարհի տևողությունը (Նկ.18.2):

Իմանալով Echo PIN-ի վրա ազդանշանի տևողությունը՝ կարող ենք



Նկ.18.2. Echo և Trig PIN-երին տրվող ազդանշանները

հաշվել թիրախի հեռավորությունը՝ ըստ հետևյալ բանաձևի.

$$S_{\text{թիրախ}} = V * t:$$

Քանի որ ազդանշանը անցնում է թիրախի հեռավորությունից երկու անգամ ավելի երկար ճանապարհ, ապա թիրախի հեռավորությունը որոշելու համար $S_{\text{թիրախ}}$-ը պետք է բաժանել 2-ի՝

$$S = \frac{V_{\text{ձ}} * t}{2},$$

որտեղ $V_{\text{ձ}}$ -ն ձայնի արագությունն է օդում՝

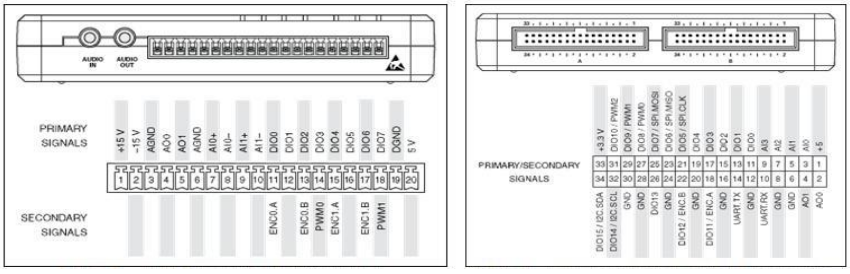
$$V_{\text{ձ}} \approx 340\ {}^{\text{մ}}/_{\text{վ}} = 0.034\ {}^{\text{սմ}}/_{\text{մկվ}}$$

$$S = \frac{V_{\text{ձ}} * t(\text{մկվ})}{2} \approx 0.017 * t(\text{մկվ})$$

LabVIEW միջավայրում ուլտրաձայնային տվիչը ծրագրավորենք NI myRIO սարքի միջոցով: Սարքը բաղկացած է երկու ծրագրավորման հարթակներից՝ Real Time և FPGA (Field Programming Gate Array): Ծրագրավորենք FPGA միջավայրում, քանի որ գրանցվող ժամանակահատվածը ընկած է միկրովայրկյանային տիրույթում: Սարքը հնարավորու-

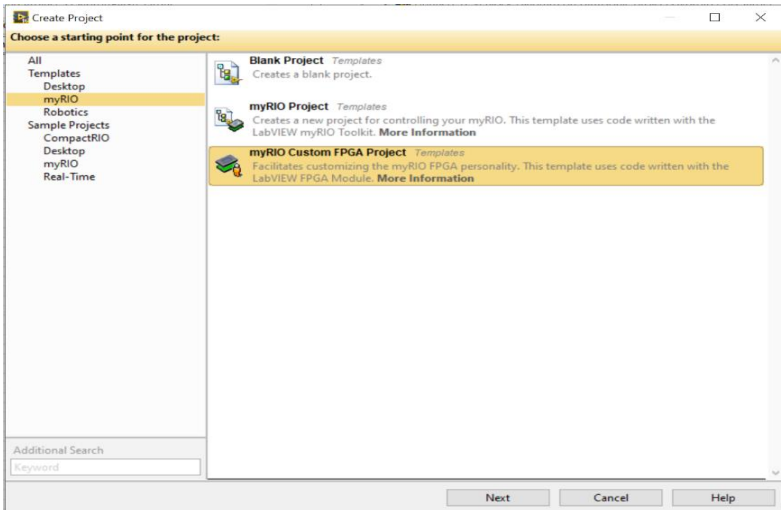
թյուն է տալիս ներբեռնել անհրաժեշտ ծրագիրը և օգտագործել այն առանց համակարգչի: Նկ. 18.3-ում պատկերված է myRIO սարքի ֆոտո-պատկերը, իսկ Նկ. 18.4-ում՝ անալոգաթվային մուտքերի և ելքերի միաց-ման հանգույցների սխեմատիկ տեսքերը, որոնք բաժանված են երեք խմբաքանակների՝ A, B և C:

Նկ. 18.3. myRIO սարքի արտաքին տեսքը



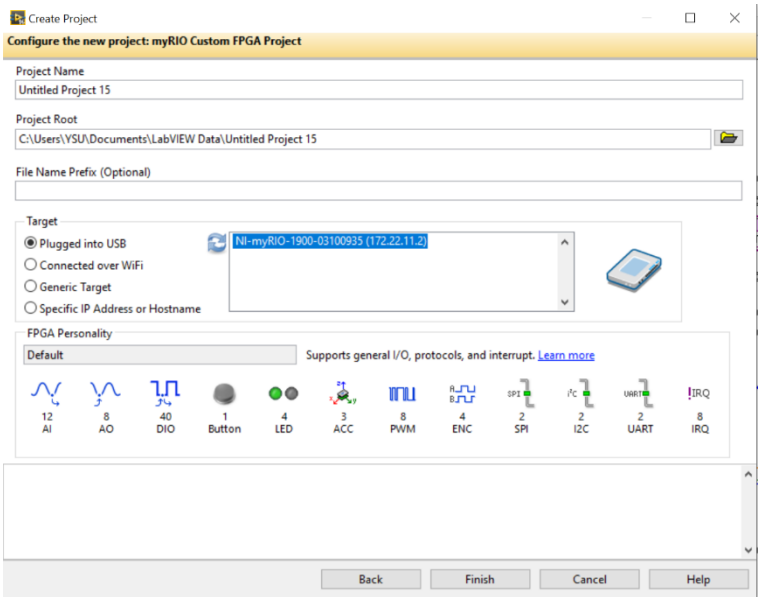
Նկ. 18.4. Աջ կողմում A և B խմբավորված հանգույցներն են, իսկ ձախ կողմում՝ C-ն:

Տեղադրելով սարքի անհրաժեշտ կարգավորումները համակարգչում՝ *LabVIEW* միջավայրում ընտրում ենք File բաժինի Create Project…-ը, կստանանք Նկ. 18.5-ում պատկերված պատուհանը: Այստեղից ընտրենք myRIO/myRIO Custom FPGA Project-ը և Next-ը:



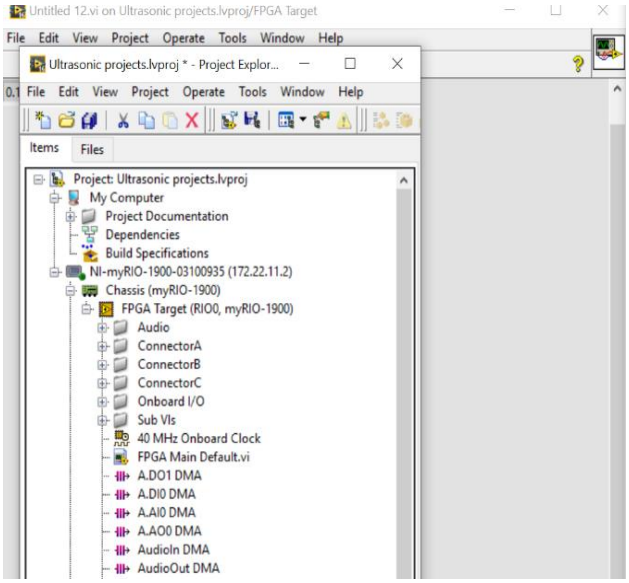
Նկ. 18.5

Արդյունքում կստանանք Նկ. 18.6 պատուհանը, որտեղ առաջին տողում նախագծի անունն է, որը կարելի է փոփոխել նախընտրելի անվամբ: Երկրորդ տողում նախագծի տեղն է, իսկ Target բաժնում՝ միացման հնարավորությունները:



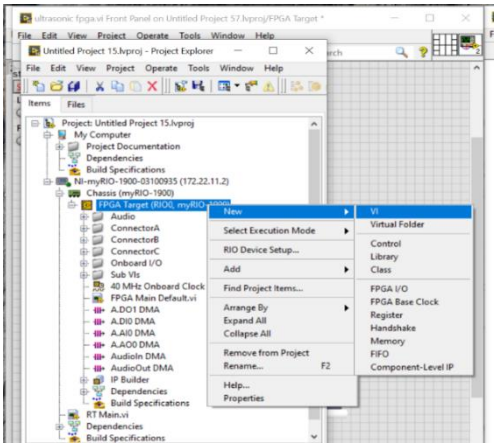
Նկ. 18.6

Սեղմելով Finish կոճակը՝ կստանանք Նկ. 18.7-ում պատկերված պատուհանը, որտեղ երևում են սարքի բոլոր թվային և անալոգային մուտքերն ու ելքերը: Քանի որ անհրաժեշտ է ծրագրավորել FPGA միջավայրում, ապա պետք է VI ստեղծել հենց այդ մասում՝ Նկ. 18.8: Ստեղծելուց հետո ծրագրավորենք այդ VI ֆայլում:



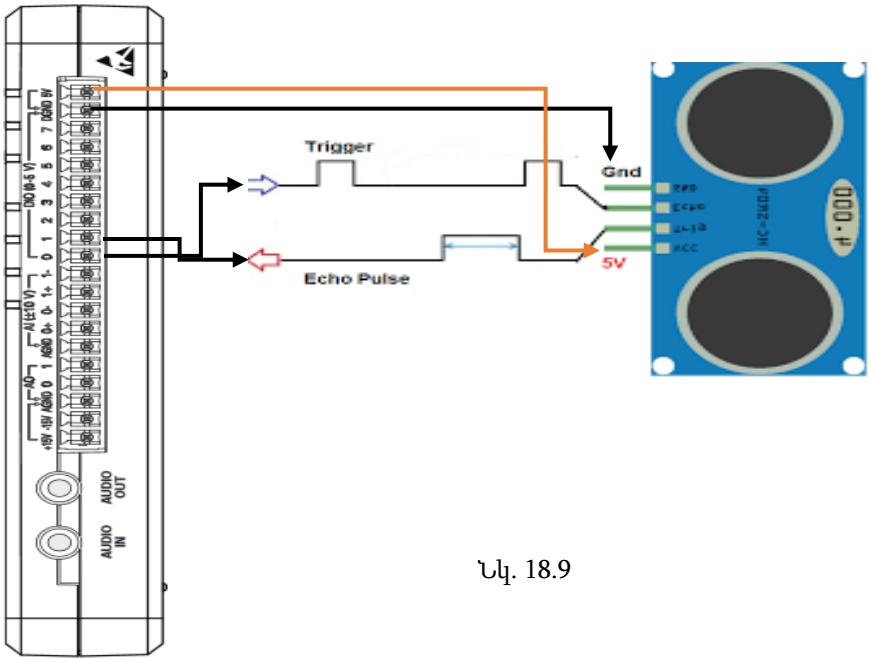
Նկ. 18.7

Հարմարության համար ընտրենք Connector C խմբավորված ելքերն ու մուտքերը:



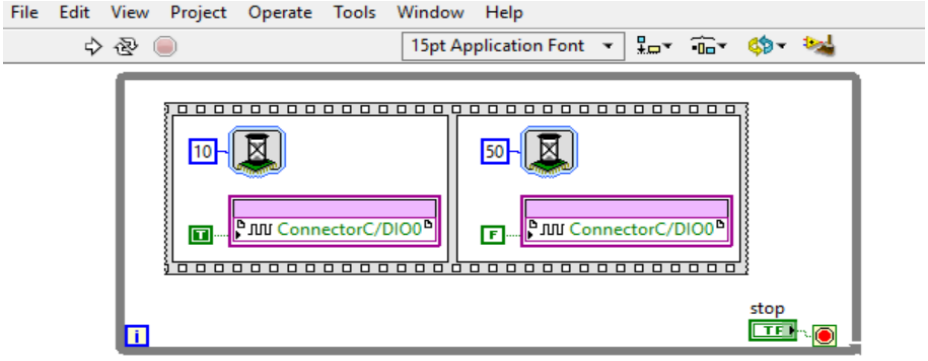
Նկ. 18.8

Նկ. 18.9-ում պատկերված է տվիչի միացման տեսքը, որտեղ 5V-ն և GND-ն միացվում են սարքի ստատիկ աղբյուրին, իսկ Trigger և Echo ազդանշանները՝ համապատասխանաբար՝ DIO0-ին և DIO1-ին:



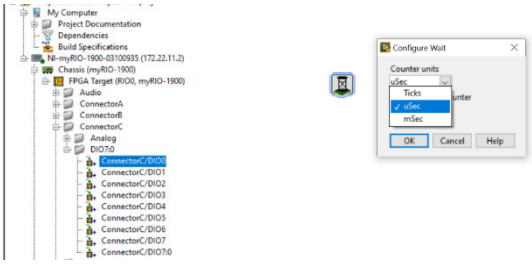
Նկ. 18.9

Սկզբում Trigger-ի համար գեներացնենք ազդանշան DIO0-ն ելքին: Ծրագիրը կընդունի Նկ. 18.10-ում պատկերված տեսքը:



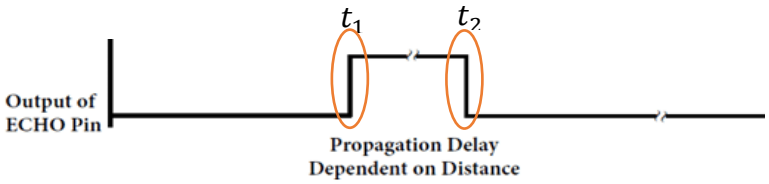
Նկ. 18.10

Connector C-ի միջից ընտրենք DIO0 պորտը և մկնիկի օգնությամբ այն տեղափոխենք համապատասխան VI-ի մեջ, իսկ ժամանակը վերցնենք Timing բաժնից և ընտրենք մկվ (Նկ. 18.11):



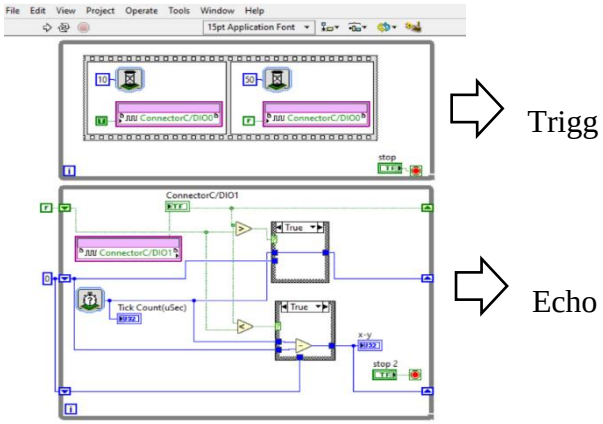
Նկ. 18.11

Այսպիսով, ստացվում է Նկ. 18.2-ի Trigger ազդանշանը, իսկ Echo Pin-ի ազդանշանից t ժամանակը ստանալու համար անհրաժեշտ է գրանցել իմպուլսի տևողությունը՝ $t = t_2 - t_1$ (Նկ. 18.12):



Նկ. 18.12

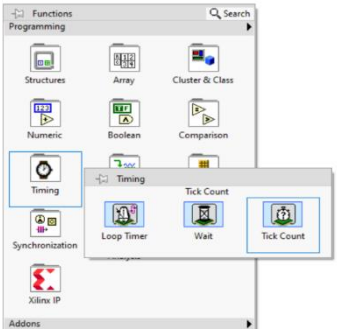
Այսպիսով, վերջնական ծրագիրը կունենա Նկ. 18.13-ում պատկերված տեսքը: Այն բաղկացած է երկու ցիկլերից, որոնցից մեկը Trigger



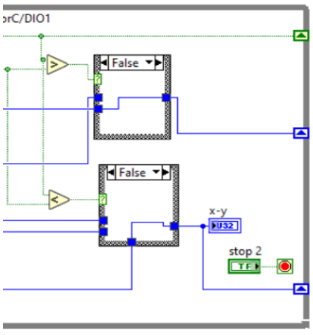
Նկ. 18.13

ազդանշանի գեներացման համար է, իսկ մյուսը՝ Echo ազդանշանի գրանցման և մշակման:

Տրամաբանական   մեծ և փոքր գործիքների շնորհիվ գրանցում ենք վերելքի և վայրէջքի ժամանակները, իսկ ակնթարթային ժամանակը գրանցելու համար վերցնենք Timing բաժնի Tick Count գործիքը՝ ընտրելով մկկ (Նկ. 18.14): Նկ. 18.15-ում պատկերված է Case Structure-ների False էջերի միացումները:

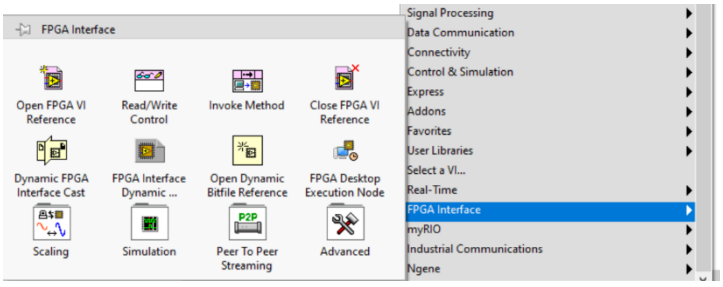


Նկ. 18.14



Նկ. 18.15

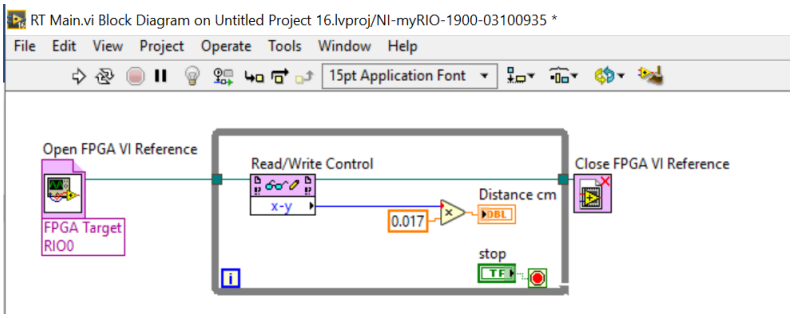
FPGA հարթակից տվյալների դինամիկ ստացման համար Real Time բաժնում ստեղծենք VI: Այնուհետև վերջինիս Block Diagram-ում ընտրենք FPGA Interface-ը և վերցնենք Open FPGA VI Reference, Read/Write Control և Close FPGA VI Reference գործիքները (Նկ. 18.16):



Նկ . 18.16

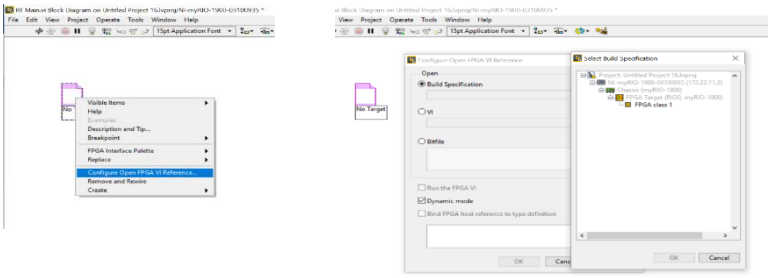
Ծրագիրը կունենա Նկ. 18.17-ում ցուցադրված տեսքը:

Որտեղ FPGA Target RIO0-ն ընտրենք հետևյալ կերպ. իր վրա սեղմելով մկնիկի աջ կոճակը՝ ընտրենք Configure Open FPGA VI Reference… և ընտրենք համապատասխան FPGA VI ֆայլը՝ OK-ի միջոցով



Նկ. 18.17

(Նկ. 18.18)։ Read/Write Control-ից ընտրենք $x - y$ փոփոխականը, այնուհետև այն բազմապատկելով 0.017 հաստատունով՝ կստանանք չափվող հեռավորությունը սմ-ով։



Նկ. 18.18

19. Լաբորատոր աշխատանք 7

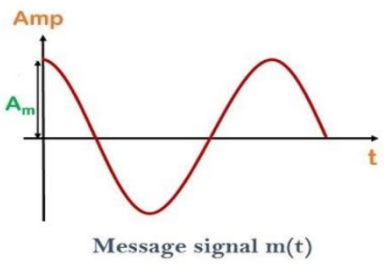
Ամպլիտուդային մոդուլում

Ժամանակակից կապի համակարգերում տվյալների փոխանցման համար կապուղու հատկություններին առավել հարմարավետ ազդանշաններ ձևավորելու համար կատարվում է մոդուլում: Մոդուլումը բարձրհաճախային (կրող) հարմոնիկ ազդանշանի որևէ պարամետրի՝ ամպլիտուդի, հաճախության կամ փուլի փոփոխումն է ինֆորմացիոն (ցածրհաճախային) ազդանշանի օրենքով: Եթե ինֆորմացիոն ազդանշանի օրենքով փոփոխվում է կրողի ամպլիտուդը, այն անվանում են ամպլիտուդամոդուլված ազդանշան, իսկ մոդուլման տեսակը՝ ամպլիտուդային, եթե փոփոխվում է հաճախությունը, հաճախային, եթե փուլը, փուլային: Նշված մոդուլման եղանակներից ամենապարզը ամպլիտուդային մոդուլումն է:

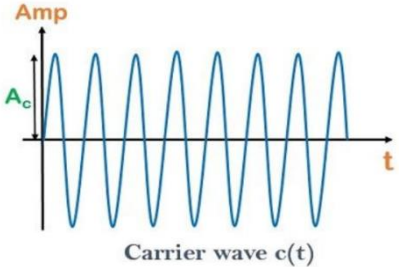
Այս լաբորատոր աշխատանքում կդիտարկենք ամպլիտուդային մոդուլման իրականացումը *LabVIEW* միջավայրում:

Դիցուք՝ ունենք $m(t)$ մոդուլող (ինֆորմացիոն) և $c(t)$ կրող ազդանշաններ (Տե՛ս Նկ. 19.1 և Նկ. 19.2):

$$m(t) = A_m cos(\omega_m t), \qquad c(t) = A_c cos(\omega_c t)$$



Նկ.19.1



Նկ. 19.2

Ամպլիտուդամոդուլված ազդանշանը կլինի՝

$$y(t) = A_0 cos(\omega_c t):$$

Քանի որ y(t) հավասարման ամպլիտուդը պետք է փոփոխվի ինֆորմացիոն ազդանշանի օրենքով, ապա կունենանք՝

$$A_0 = A_c + m(t):$$

Տեղադրելով A_0-ի արժեքը y(t)-ի արտահայտության մեջ՝ կստացվի՝

$$y(t) = \big(A_c + A_m cos(\omega_c t)\big) cos(\,\omega_c t):$$

Այնուհետև բաժանելով A_cAc-ի վրա՝ կստացվի՝

$$y(t) = A_c \left(1 + \frac{A_m}{A_c} cos(\omega_m t)\right) cos(\,\omega_c t)$$

$$\frac{A_m}{A_c} = \mu,$$

որտեղ μ-ն կոչվում է մոդուլման ցուցիչ կամ մոդուլման խորություն:

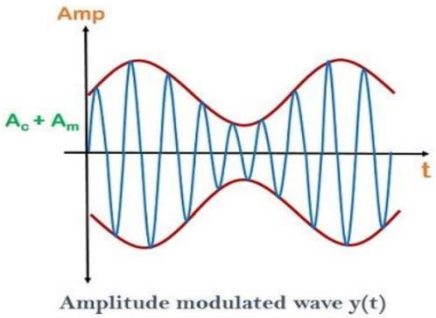
Որոշենք ամպլիտուդի մաքսիմալ և մինիմալ արժեքները, երբ

$$cos(\omega_m t) = 1 \qquad A_{max} = A_c + A_m\,,$$
$$cos(\omega_m t) = -1 \quad A_{max} = A_c + A_m:$$

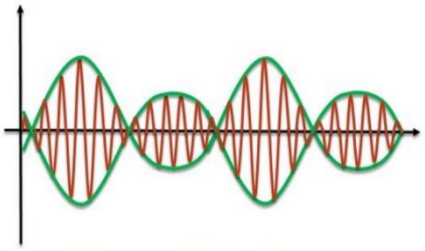
Որոշակի ձևափոխություններից հետո կստանանք՝

$$\mu = \frac{A_{max} - A_{min}}{A_{max} + A_{min}}:$$

Երբ մոդուլման ցուցիչը փոքր է 1-ից, ապա կստանանք գծային մոդուլում (Նկ. 19.3):



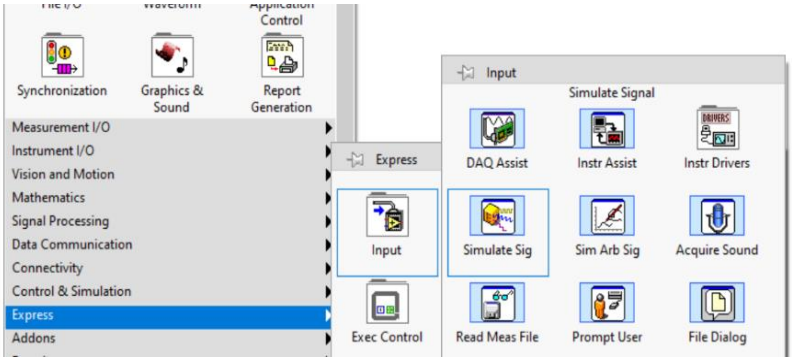
Նկ.19.3



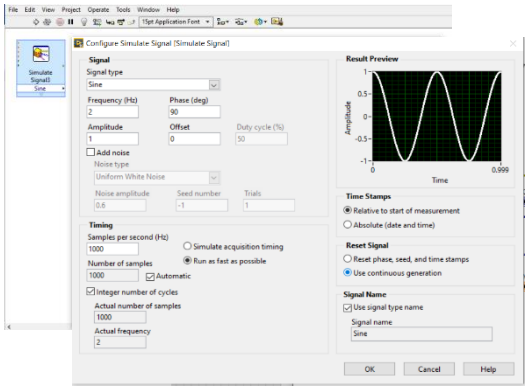
Նկ.19.4

Իսկ եթե μ -ն մեծ է 1-ից, ապա կստացվի վերամոդուլում (Նկ. 19.4):

Ամպլիտուդային մոդուլումը կարելի է իրականացնել *LabVIEW* միջավայրում՝ համակարգչային նմանակման միջոցով: Block Diagram-ում ընտրենք Express\Input և Simulate Signal գործիքը (Նկ. 19.5): Տեղադրելով փոփոխականները ω_c-ի և ω_m-ի համար՝ կունենանք cos(ωt) ֆունկցիան (Նկ. 19.6):

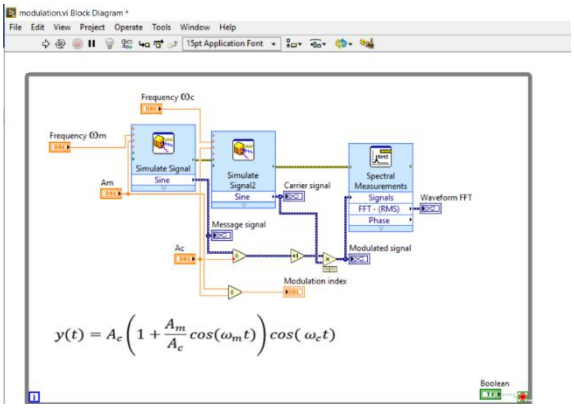


Նկ. 19.5



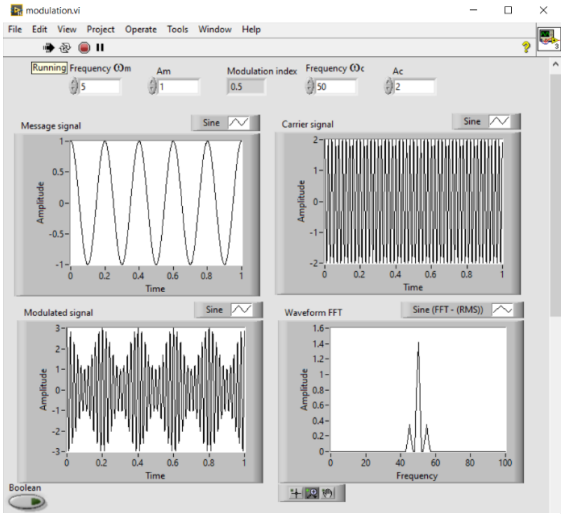
Նկ. 19.6

Ծրագիրը վերջնականապես կընդունի Նկ.19.7-ում ցույց տրված տեսքը: Ներկայացված է նաև y(t) ֆունկցիայի մաթեմատիկական տեսքը:



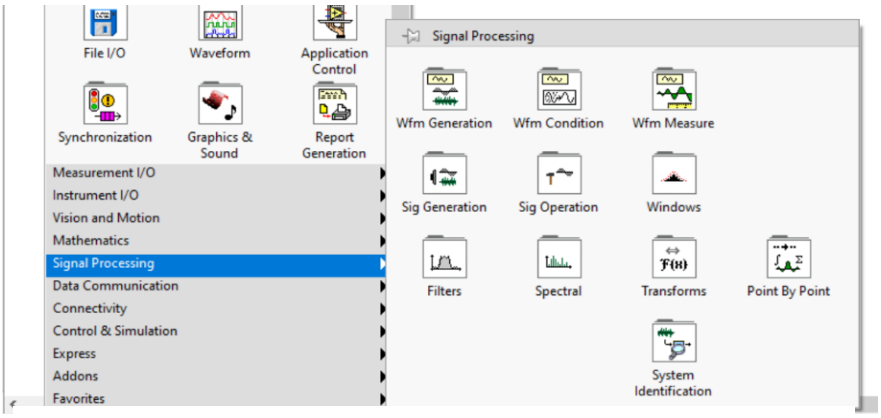
Նկ. 19.7

Նկ. 19.8-ում պատկերված է Front Panel-ի տեսքը, որտեղ երևում են բարձր ու ցածր հաճախություններով ազդանշանների գրաֆիկական տեսքերը, իսկ ներքևում երևում են մոդուլված ազդանշանը և դրա սպեկտրը: Այստեղ կարելի է փոփոխել ազդանշանների հաճախություններն ու ամպլիտուդները: Ցուցադրված է նաև մոդուլման խորության ընթացիկ արժեքը:



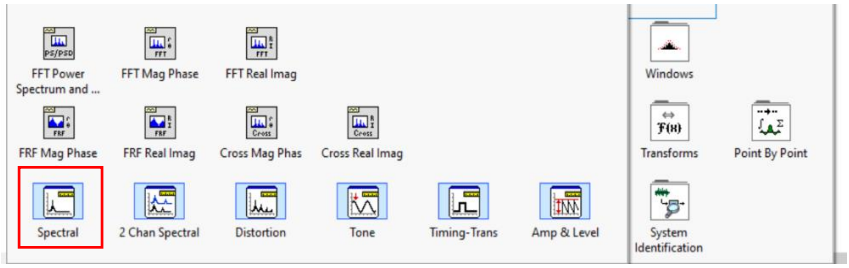
Նկ. 19.8

Ստացված ազդանշանի սպեկտրալ տեսքը ներկայացնելու համար կարելի է օգտագործել Signal Processing բաժնի Wfm Measure խմբի Spectral գործիքը (Տե՛ս Նկ. 19.9 և Նկ. 19.10):

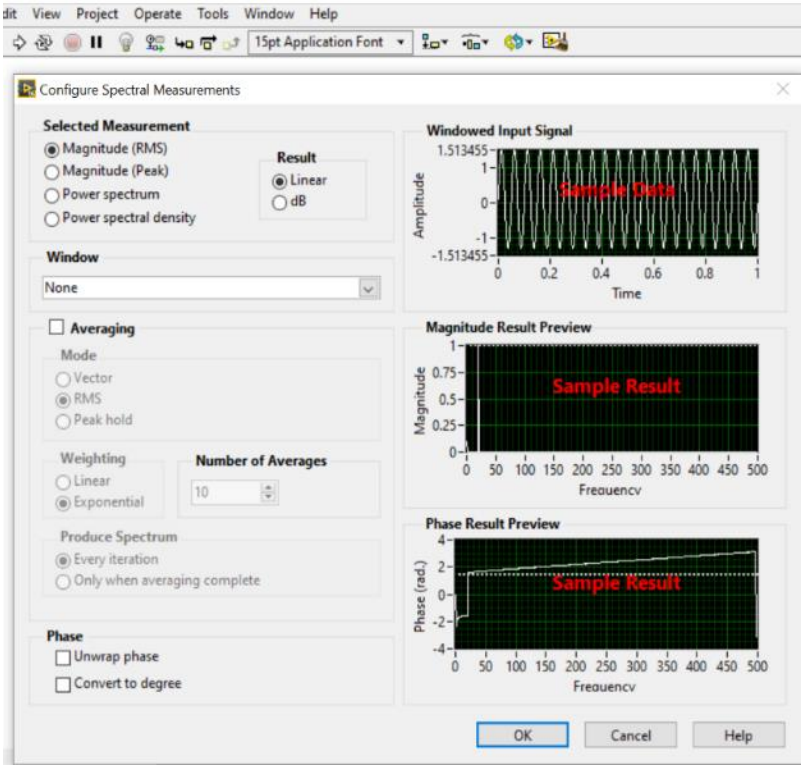


Նկ. 19.9

Տեղադրելով համապատասխան սպեկտրալ կարգավորումները՝ կունենանք Waveform FFT գրաֆիկական տեսքը (Նկ. 19.8)։



Նկ. 19.10



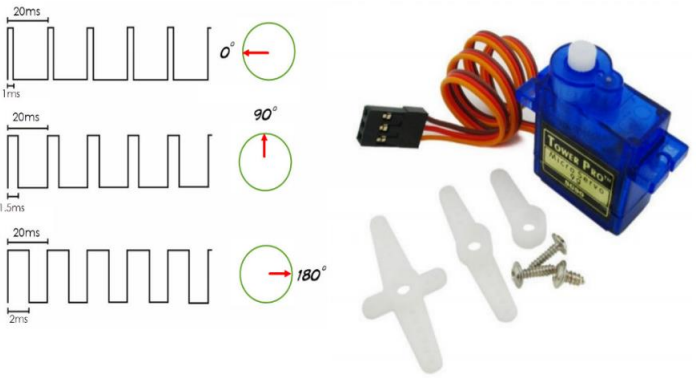
Նկ. 19.11

20. Լաբորատոր աշխատանք 8

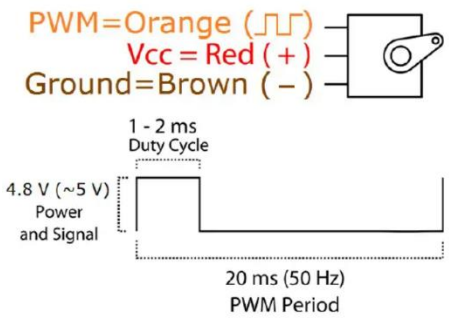
Սերվո շարժիչի ղեկավարումը

Ինչպես քայլային շարժիչները, այնպես էլ սերվո շարժիչները լայնորեն օգտագործվում են թե՛ ռոբոտաշինության մեջ և թե՛ ավտոմատացված համակարգերում: Սերվո շարժիչները, ի տարբերություն քայլային շարժիչների, ունեն հետադարձ կապ, որով հնարավոր է ավելի մեծացնել քայլի կամ անկյան ճշտությունը: Այստեղ նույնպես հնարավոր է ղեկավարել ոչ միայն անկյունը, այլ նաև պտտման արագությունը: Պտտման անկյունը կազմում է 180^0 կամ $\pm 90°$:

Նկ. 20.1-ում պատկերված է SG-90 սերվո շարժիչի արտաքին տեսքը և տրվող ազդանշանի տեսքը, որտեղ իմպուլսի լայնության փոփոխությամբ ընտրվում է համապատասխան անկյան դիրքը:



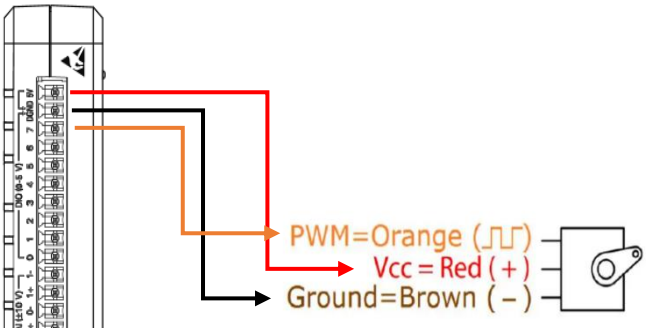
Նկ. 20.1



Նկ. 20.2

Սերվո շարժիչի մուտքերն են՝ սնուցման աղբյուրը (V_{cc} և GND) և PWM(Pulse Width Modulation) իմպուլսալայնքային մոդուլված ազդանշանը, որտեղ իմպուլսի տևողությունը 20ms է, իսկ դրական մասի տևողությունը՝ 1-2ms (Նկ. 20.2): Պտտման անկյան շատ կետեր ունենալու համար շարժիչի ղեկավարումը իրականացնելու ենք FPGA հարթակում՝ ժամանակի համար ընտրելով մկվ:

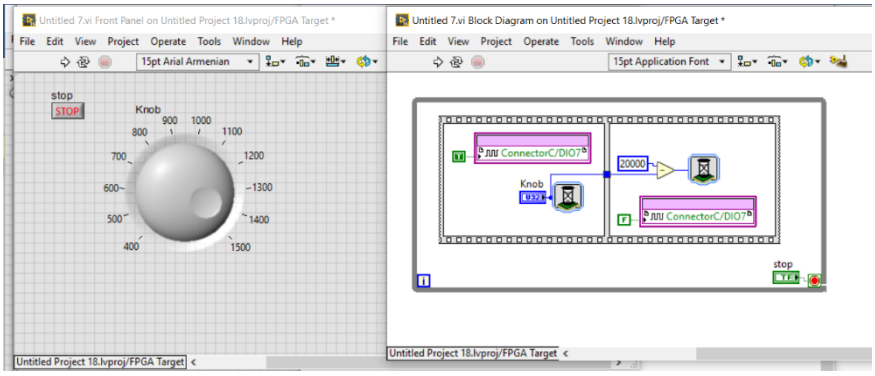
Շարժիչը myRIO-ին միացման սխեմատիկ տեսքը պատկերված է Նկ. 20.3-ում: Օգտվելով Լաբորատոր աշխատանք 6-ից, որտեղ նկարագրված են myRIO-ի FPGA հարթակի ծրագրավորման հմտությունները, սերվո շարժիչի ղեկավարման ծրագիրը կբերվի Նկ. 20.4-ում պատկերված տեսքին:



Նկ. 20.3

Իմպուլսի տևողությունը հաստատուն պահելու համար 20000մկվ-ից հանենք դրական մասի ընթացիկ լայնությունը: Իսկ Front Panel-ում տեղադրված բռնակի միջոցով անկյունը փոփոխվում է 400-1500-ի, որը համապատասխանում է 0-180⁰-ին:

Անհրաժեշտության դեպքում Real Time հարթակի միջոցով կարելի է ղեկավարել ծրագիրը (Մանրամասն նկարագրությունը տե՛ս Լաբորատոր աշխատանք 6-ում):



Նկ. 20.4

РЕЗЮМЕ

Автоматизация научного эксперимента в среде *LabVIEW*

Учебное пособие “Автоматизация научного эксперимента в среде *LabVIEW*” посвящено описанию серии лабораторных работ, предназначенных для проведения лабораторных занятий курсов “Автоматизация научного эксперимента” и “Практика автоматизации научного эксперимента” в соответствии с учебной программой радиофизического факультета. Представлены инструментарий программной среды *LabVIEW* и особенности работы пакетов, используемых в лабораторных работах. Приведено описание необходимого экспериментального оборудования для проведения лабораторных работ, а также детально представлен ход каждого эксперимента.

В руководстве описаны процессы импорта данных в компьютер и их обработки, характеристики вывода данных с компьютера. Представлен процесс автоматизации управления аппаратурой и проведения экспериментов с применением программной среды *LabVIEW*.

SUMMARY

Automation of a Scientific Experiment in the *LabVIEW* environment

The textbook "Automation of a Scientific Experiment in the *LabVIEW* environment" is devoted to a series of descriptions of laboratory work, which, in accordance with the curriculum of the Faculty of Radiophysics, are intended as a part of the laboratory classes of the courses "Automation of a scientific experiment" and "Practice of automation of a scientific experiment". The tools of the *LabVIEW* software environment and the operating features of the packages used in laboratory work are presented. Descriptions of laboratory work are provided with the equipment for conducting them and the step-by-step description of the experiment itself.

The manual presents the data import and processing in the computer. The characteristics of data output from a computer, and the automation of the process of controlling equipment and conducting experiments using the *LabVIEW* software environment are introduced.

Գրականություն

1. https://www.ni.com/ [Official Website]
2. Блюм, Питер. *LabVIEW*: стиль программирования. Litres, 2022.
3. Ashley, Kenneth L. Analog electronics with *LabVIEW*. Prentice Hall Professional, 2002.
4. Bitter, Rick, Taqi Mohiuddin, and Matt Nawrocki. *LabVIEW*™ Advanced Programming Techniques. CRC press, 2017.
5. Essick, John. Hands-on introduction to *LabVIEW* for scientists and engineers. Oxford University Press, 2013.
6. Kehtarnavaz, Nasser, and Namjin Kim. Digital signal processing system-level design using *LabVIEW*. Elsevier, 2011.
7. Travis, Jeffrey. *LabVIEW* for everyone. Pearson Education India, 2009.

ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Աբրահամյան Տիգրան Արսենի, Պարսամյան Հենրիկ Աշոտի

ԳԻՏԱՓՈՐՁԻ ԱՎՏՈՄԱՏԱՑՈՒՄ *LabVIEW* ՄԻՋԱՎԱՅՐՈՒՄ

(ուսումնամեթոդական ձեռնարկ)

Հրատ. պատ. խմբագիր՝ Լ. Հովհաննիսյան
Համակարգչային ձևավորումը՝ Կ. Չալաբյանի
Կազմի ձևավորումը՝ Ա. Պատվականյանի
Հրատ. սրբագրումը՝ Ա. Գույումջյանի

Ստորագրված է տպագրության՝ 01.10.2023:
Չափսը՝ 70x100 $^{1}/_{16}$: Տպ. մամուլը՝ 5.875:

ԵՊՀ հրատարակչություն
ք. Երևան, 0025, Ալեք Մանուկյան 1
www.publishing.ysu.am