Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Сучасні цифрові пристрої систем автоматизації

Комп’ютерний практикум

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою
«Комп'ютерно-інтегровані системи та технології в приладобудуванні»
спеціальностей
151 «Автоматизація та комп’ютерні технології» та
174 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка»

Укладач: О.М. Павловський

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2024

УДК 004.04:06 (П12)
П12

Укладач: *Павловський Олексій Михайлович*, канд. техн. наук, наук, доцент, доцент кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем КПІ ім. Ігоря Сікорського

Рецензент: *Головач С.В.* - канд. техн. наук, головний конструктор напрямку АТ «Елміз»

Відповідальний редактор: *Півторак Д.О.* - канд. техн. наук, доцент, доцент кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем КПІ ім. Ігоря Сікорського

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 5 від 29.02.2024 р.)
за поданням Вченої ради Приладобудівного факультету
(протокол № 2/24 від 26.02.2024 р.)

П12

Сучасні цифрові пристрої систем автоматизації. Комп'ютерний практикум [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра/ за освіт. програмою «Комп'ютерно-інтегровані системи та технології в приладобудуванні» спец. 151 «Автоматизація та комп'ютерні технології» та 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»/ КПІ ім. Ігоря Сікорського ; уклад.: О.М. Павловський. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 80 с.

Навчальний посібник містить теоретичні відомості та практичні рекомендації до виконання комп'ютерних практикумів з дисципліни «Сучасні цифрові пристрої систем автоматизації». Під час виконання комп'ютерних практикумів студенти здобудуть навички роботи із середовищем графічного програмування LabVIEW від National Instruments, познайомляться із принципами створення віртуальних приладів, основними конструкціями та палітрами інструментів. Реалізують інтерфейси для взаємодії із сучасними мікроконтролерами для опитування чутливих елементів та керування виконавчими елементами кіберфізичних систем.

Навчальний посібник призначений для здобувачів ступеня бакалавра за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології» та 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка», буде також корисним студентам споріднених технічних спеціальностей.

УДК 004.04:06 (П12)

Реєстр. № НП 23/24-328 Обсяг 3,8 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ВСТУП

У сучасному світі використання електронних пристроїв перейшло на абсолютно новий рівень, і тому саме поняття «цифрові пристрої» має більш широке значення ніж в межах класичної цифрової електроніки та схемотехніки. Відтак, сучасні пристрої та прилади окрім відповідної елементної бази мають ряд додаткових властивостей та функціональних можливостей, з врахуванням тенденції до комплексування та об'єднання сучасних приладів в єдині екосистеми, то і підхід до розробки таких пристрої суттєво відрізняється. В найпростішому випадку вже не достатньо розробити і виготовити автономний пристрій, сучасні тенденції вимагають наявності інтерфейсів для передачі, прийому, збереження та обробки даних на віддалених системах (в найпростішому випадку – від мікроконтролера до ПК або на сервер). Тож реалізація інтерфейсу зв'язку є обов'язковою вимогою. В даному навчальному посібнику і буде запропоновано освоїти створення таких інтерфейсів за допомогою платформи LabVIEW від National Instrumets (NI). Для цього студентам пропонується опанувати основні прийоми по створенню віртуальних приладів починаючи від базових інструментів і конструкцій, закінчуючи розробкою прототипів пристроїв з передачею даних від чутливих елементів та керуванням виконавчих пристроїв. Враховуючи простоту графічної мови програмування, яка використовується в LabVIEW, наявність тулбоксів та можливість її маштабованості на різні платформи, такий підхід є найбільш гнучким при розробці сучасних цифрових пристроїв.

Матеріали в посібнику розташовані по мірі ускладнення і сформовані в окремі тематичні комп'ютерні практикуми. Для фокусування уваги, по тексту будуть зустрічатись примітки які акцентують на важливих моментах або додатково розкривають наведені викладки.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 1
СТВОРЕННЯ ПРОСТИХ ВІРТУАЛЬНИХ ПРИЛАДІВ. ПАЛІТРА EXPRESS

Мета роботи: ознайомлення з основним інструментарієм середовища NI LabVIEW створення віртуальних приладів та палітрою Express.

1.1. Теоретичні відомості

Програми створені у середовищі NI LabVIEW називаються віртуальними приладами (ВП або VI – Virtual Instrument) бо їх вигляд і функціонал імітують фізичні прилади. Середовище NI LabVIEW містить великий набір інструментальних засобів для збору, аналізу, обробки і збереження даних, а також інструменти, які допомагають відладити створюваний код.

Написання програми у NI LabVIEW починаються зі створення інтерфейсу користувача(тобто лицьової панелі), яка містить елементи керування і елементи для відображення інформації(індикаторні або індикативні елементи). Прикладами елементів керування є ручки, кнопки, кругові шкали і інші елементи вводу, індикативними – світлодіоди, показоміри, діаграми та осцилограми та ін.

1.1.1. Створення ВП із шаблону

Для прикладу створимо найпростіший віртуальний прилад, що генерує прості періодичні сигнали і відображує їх на лицьовій панелі. Після запуску відкривається стартове вікно середовища, яке дозволяє створити новий віртуальний прилад або проект, також в ньому будуть відображені останні файли з якими проводили роботу користувачі. Загальний вигляді стартового вікна показаний на рис.1.1. Зазначимо, що в залежності від встановленої версії стартовий інтерфейс може незначно змінюватись, проте основний функціонал залишається не змінним.

Для прискорення створення ВП використаємо шаблонну заготовку Generate and Display, для цього скористаємося меню *File -> New...* і у

відкрившомуся вікні оберемо зазначений віртуальний прилад в категорії *VI->From Template -> Simulated.*

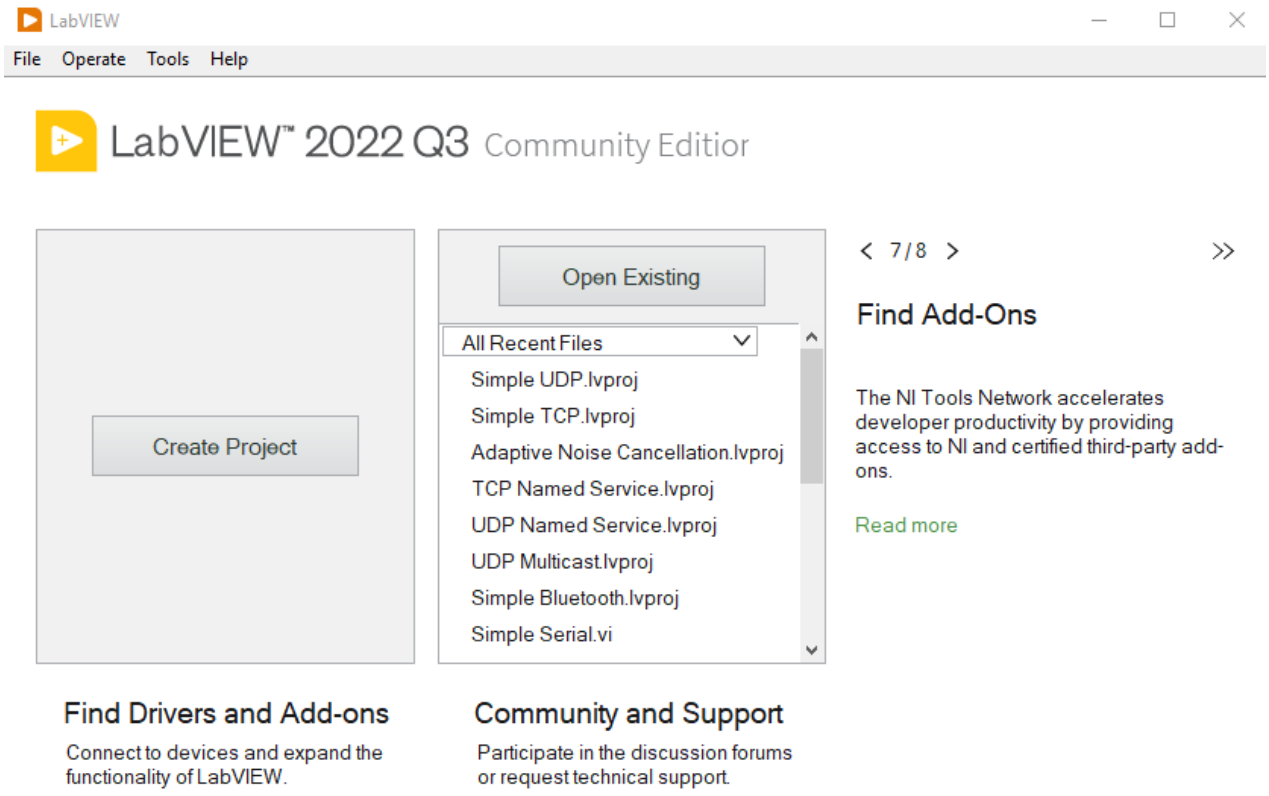
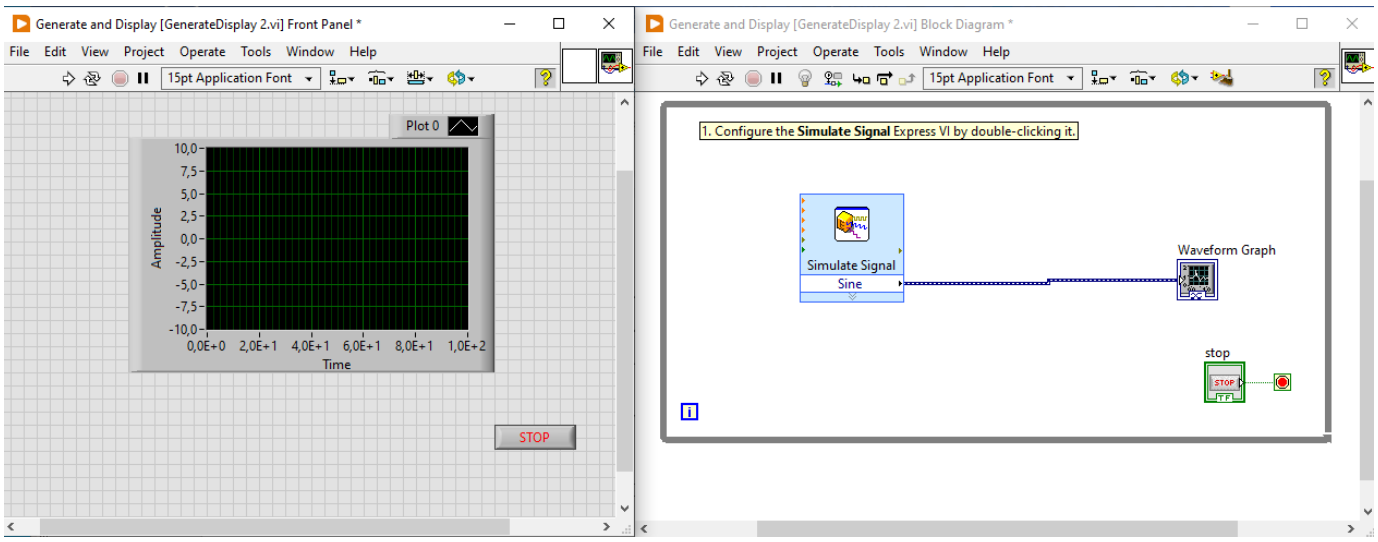


Рис. 1.1. Стартове вікно середовища NI LabVIEW

Віртуальний прилад умовно розділений на дві частини у вигляді двох взаємопов'язаних панелей, лицьової панелі - для реалізації користувальницького інтерфейсу, та панелі блок-діаграм для створення функціоналу, або простіше «для написання коду» використовуючи мову графічного програмування «G». Загальний вигляд двох панелей ВП-шаблону показаний на рис. 1.2.

Примітка: Якщо лицьової панелі або блок-діаграми не видно, можна зробити її видимою/активною, вибравши меню Window »Show Front Panel / Window »Show Block Diagram. Для швидкого перемикання між вікном лицьової панелі і блок-діаграмою пропонується використовувати комбінацію клавіш <Ctrl-E>.

Перед початком роботи з віртуальним приладом необхідно його зберегти із зручною назвою використовуючи меню *File ->Save As…*



а) б)

Рис. 1.2. Загальний вигляд ВП-шаблону Generate and Display :
а) Лицьова панель; б)Блок-діаграма

Примітка: Середовище NI LabVIEW має пагану зворотню сумісність версій, тому якщо ваш віртуальний прилад буде відкриватись в середовищі попередніх версій слід перед збереження скористатись меню *File -> Save for Previous Version* з обранням необхідної або старішої версії середовища та, при необхідності, тулкіта.

Якщо запустити віртуальний прилад з функціональної панелі кнопкою запуску , то на лицьовій панелі у вікні графіка буде відображена синусоїда. Модифікуємо цей ВП таким чином, щоб можна було змінювати параметри цього сигналу безпосередньо з лицьової панелі. Для простоти будемо змінювати лише амплітуду гармонічного сигналу.

Як було зазначено раніше, елементи керування лицьової панелі моделюють пристрої введення на фізичному приладі і забезпечують даними блок-діаграму ВП. Більшість фізичних приладів мають ручки керування, повертаючи які можна змінювати вхідні значення в широкому діапазоні із необхідною точністю.

Щоб додати ручку керування, необхідно з лицьової панелі викликати меню інструментів, клікнувши правою кнопкою миші по пустому простору панелі, і в теці Numeric обрати інструменти Knob або Dial як показано на рис. 1.3. Розмістимо обрану ручку на лицьовій панелі у зручному місці.

Примітка: Для того щоб палітри інструментів лицьової панелі або панелі блок-діаграм не зникали після вибору необхідного блоку, можна її зафіксувати на робочому просторі, для цього треба натиснути на зображення шпильки у верхній частині панелі.

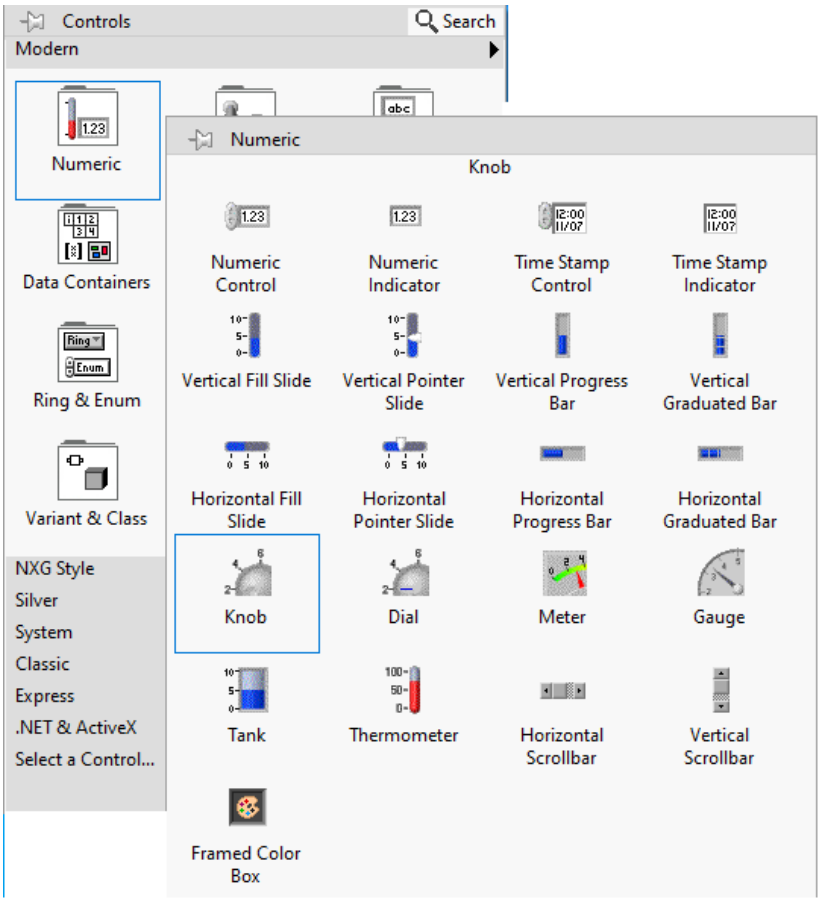


Рис. 1.3. Палітра інструментів Controls Лицьової панелі

Налаштуємо зовнішній вигляд доданого елементу керування, для цього на лицьовій панелі клацніть правою кнопкою миші по ручці і виберіть опцію Properties (Властивості) з контекстного меню. З'явиться діалогове вікно Knob Properties (Властивості ручки керування). У розділі Caption на вкладці Appearance, видаліть з текстового поля текст Knob і введіть Амплітуда (див. рис. 1.4).

Далі, у вкладці Scale в розділі Scale Style, виділіть пункт Show color ramp для реалізації градієнтної заливки шкали обертової ручки. Клацніть по кнопці OK, щоб зберегти поточну конфігурацію і закрити діалогове вікно Knob Properties. Ручка на лицьовій панелі оновиться відповідно до цих змін.

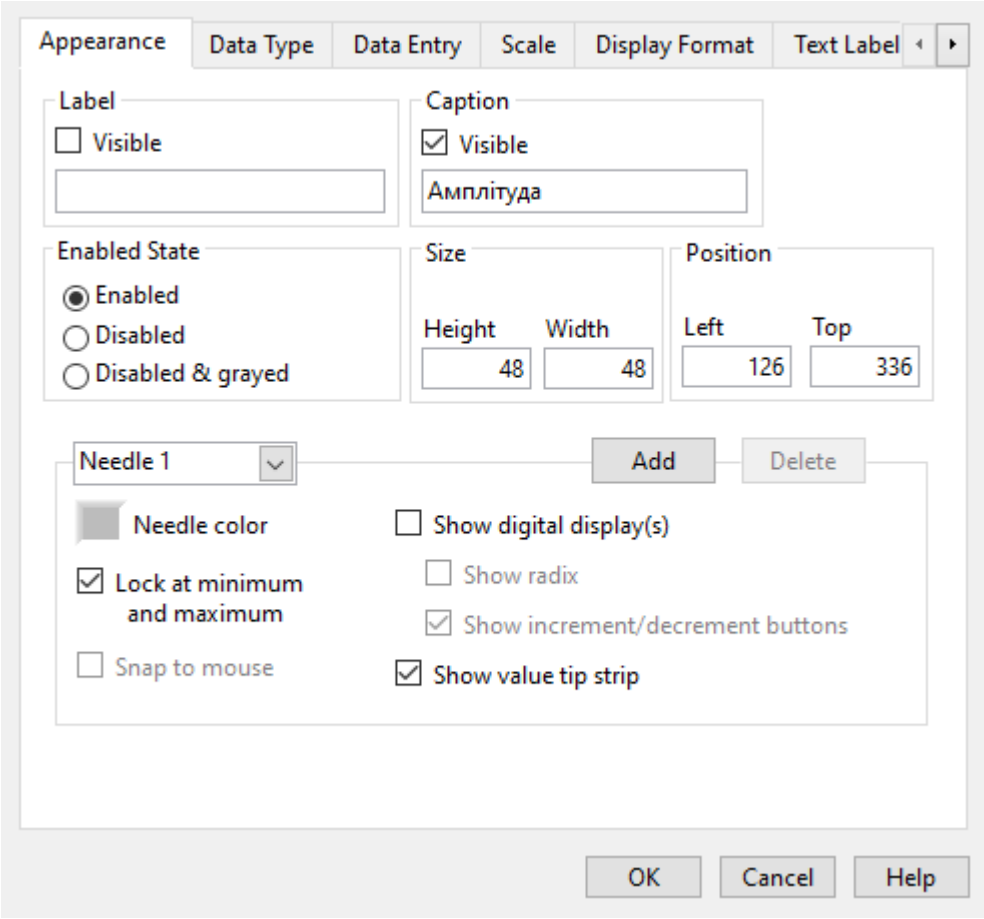


Рис. 1.4. Діалогове вікно налаштування властивостей обертової ручки Knob Properties

Після проведених операцій лицьова панель віртуального приладу буде мати наступний вигляд (рис. 1.5):

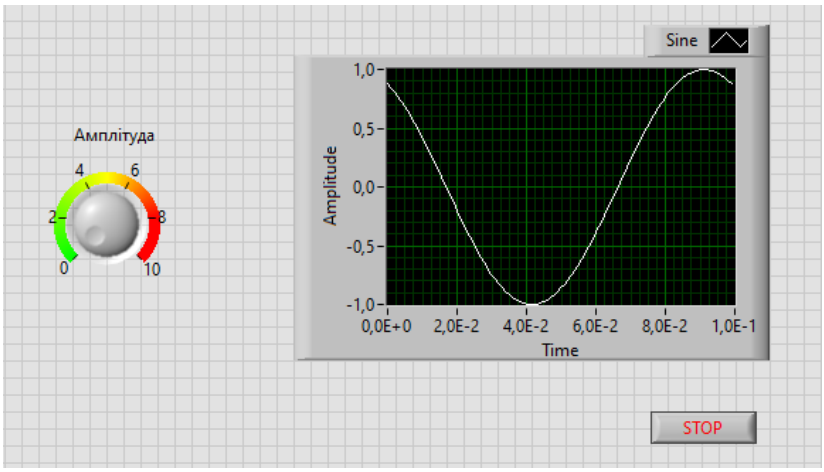


Рис. 1.5. Вигляд лицьової панелі після додавання ручки керування

За необхідністю, можна змінити назву ручки у властивості Caption.

Попереднє налаштування інтерфейсу програми завершено і можна переходити до роботи з панеллю блок-діаграм.

Кінцевий варіант блок-діаграми має виглядати як показано на рис. 1.6.

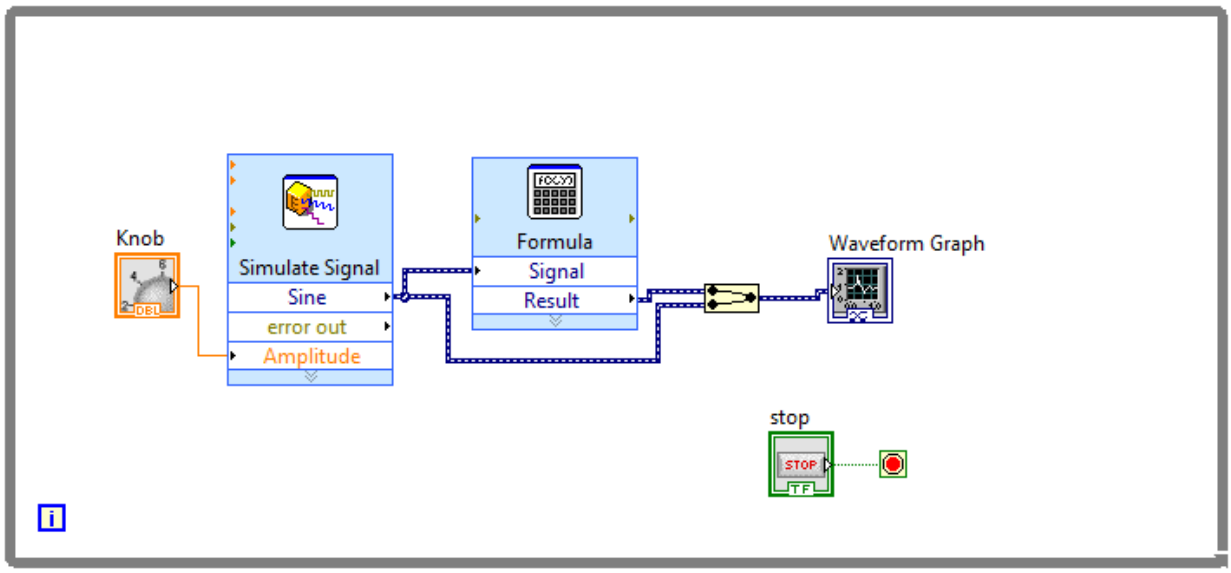


Рис. 1.6. Вигляд завершеної блок-діаграми

На панель блок-діаграм додамо ще два інструмента – Formula та Merge Signal. Для цього скористаємося пошуком викликавши меню інструментів і натиснувши на поле пошуку Search. Розмістіть знайдені елементи, як показано на рис. 1.6.

Щоб використовувати обертову ручку для зміни амплітуди сигналу, необхідно з'єднати її з входом Amplitude експрес-ВП Simulate Signal.

Для цього виконайте наступні кроки:

- На блок-діаграмі наведіть курсор на термінал елемента керування Knob (Обертова ручка). Курсор миші стане стрілкою (інструментом Позиціонування), цей інструмент використовується для вибору, розміщення або зміни розмірів об'єктів.
- Виділіть термінал Knob за допомогою інструменту позиціонування. Переконайтеся, що він знаходиться зліва від експрес-ВП Simulate Signal і всередині сірого контуру, що охоплює основну частину лицьової панелі.

Примітка: Термінали - точки введення/виведення, через які відбувається обмін інформацією між лицьовою панеллю і блок-діаграмою. Бувають, як і об’єкти, двох видів керуючі, які передають сигнал від лицьової панелі до блок-діаграми і індикаторні – передають сигнали від блок-діаграми до лицьової панелі.

- Перемістіть термінал ручки у зручне місце і зніміть виділення, клацнувши в порожньому місці блок-діаграми.
- Помістіть курсор над стрілкою на терміналі Knob, курсор автоматично змінить вигляд на котушку Wiring tool (інструмент з'єднання). Використовуйте цей інструмент для з'єднання об'єктів на блок-діаграмі. Для цього затисніть ліву кнопку миші і протягніть лінію зв'язку від виходу одного елементу до входу наступного. При фізичній можливості такого з'єднання – лінія зв'язку з'явиться. У протилежному, така лінія буде позначена пунктиром, що вказує про неможливість виконання такої операції.

Примітка: Кожен експрес-ВП забезпечений відповідною довідковою інформацією, до якої можна звернутися, натиснувши на Help у діалоговому вікні налаштувань або викликавши контекстне меню (натисканням правою кнопкою миші) експрес-ВП і вибравши пункт Help.

Тепер введемо значення у блок формули, зайдемо в його налаштування зробивши подвійний клік по блоку експрес-ВП.

Змініть текст у текстовому полі Label з X1 на Signal, щоб перейменувати вхідне значення експрес-ВП Formula. Після натискання кнопкою миші в текстовому полі String вгорі діалогового вікна Configure Formula, текст зміниться відповідно до введеної назви.

Задайте масштабний множник, вводячи символи "* 10" після слова Signal в текстовому полі String. Для цього можна використовувати кнопки введення у діалоговому вікні конфігурації або набрати з клавіатури. Якщо ви використовуєте кнопки в діалоговому вікні конфігурації, LabVIEW поміщає символи формули після змінної Signal в текстовому полі String. При використанні клавіатури, клацніть лівою кнопкою миші в текстовому полі після змінної Signal і введіть необхідну формулу.

Діалогове вікно Configure Formula (Налаштування Формули) повинно виглядати так само, як на рис. 1.7.

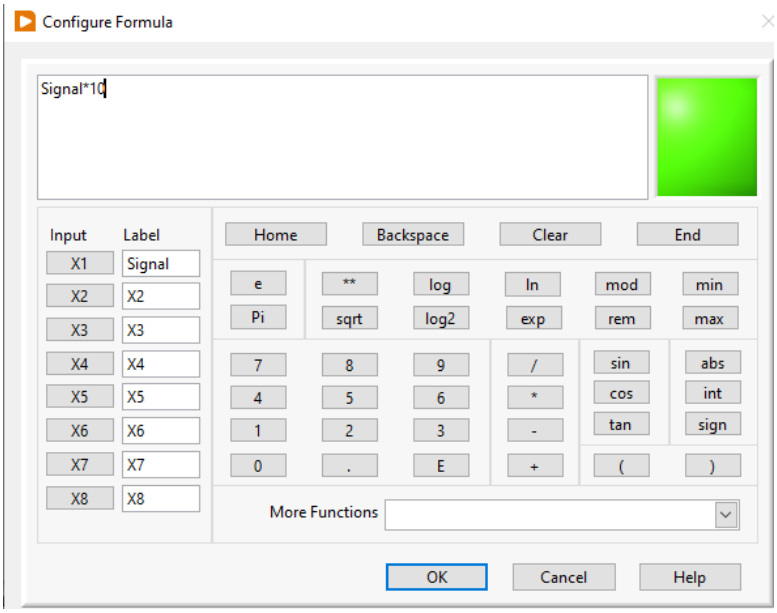


Рис. 1.7. Налаштування блоку Formula

Примітка: При введенні невірної формули в текстовому полі світлодіодний індикатор Errors (у правому верхньому куті) стане сірим і відобразить текст “Invalid Formula”

Даний блок дозволяє просто вводити формули з великою кількістю арифметичних, умовних та тригонометричних операторів та функцій, хоча може бути замінений на специфічні «одинарні» блоки для виконання будь-якої послідовності дій.

Таким самим чином зайшовши в Налаштування блоку Simulate Signal оберіть один з можливих варіантів форми сигналу (окрім DC), як показано на рис. 1.8.

Рис. 1.8. Налаштування блоку Simulate Signal. Варіанти типів сигналу для генерації

Далі з’єднайте всі блоки, як показано на рис. 1.6 та збережіть і запустіть ВП.

В результаті, остаточний варіант ВП, що відображається на лицьовій панелі після запуску має вигляд як показано на рис. 1.9.:

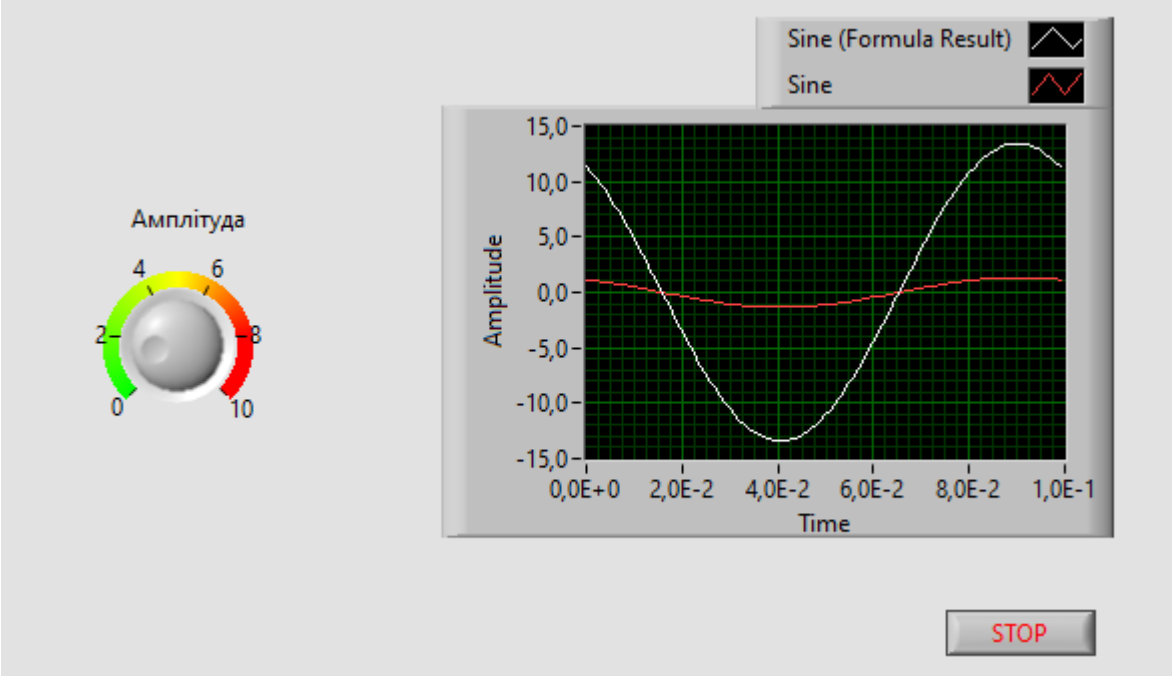


Рис. 1.9. Лицьова панель готового прикладу

За необхідності, в меню осцилограми можна змінити назви осей X та Y, а також написи «легенди» графіків.

В результаті створений віртуальний прилад дозволяє згенерувати періодичний сигнал із заданою амплітудою, маштабувати його і вивести два графіки на одному графічному вікні.

1.1.2. Перетворення сигналу та табличні данні

Тепер створимо ВП, що знешумлює гармонічний сигнал, а отримані данні виводить у вигляді таблиці.

Тут і надалі будемо розглядати лише нові функціональні блоки, оскільки основні підходи до створення і взаємодії з ВП показані вище.

Відтак, лицьова панель та блок-діаграма показані на рис. 1.10 та рис. 1.11 відповідно.

Рис. 1.10. Лицьова панель ВП для знешумлення сигналу

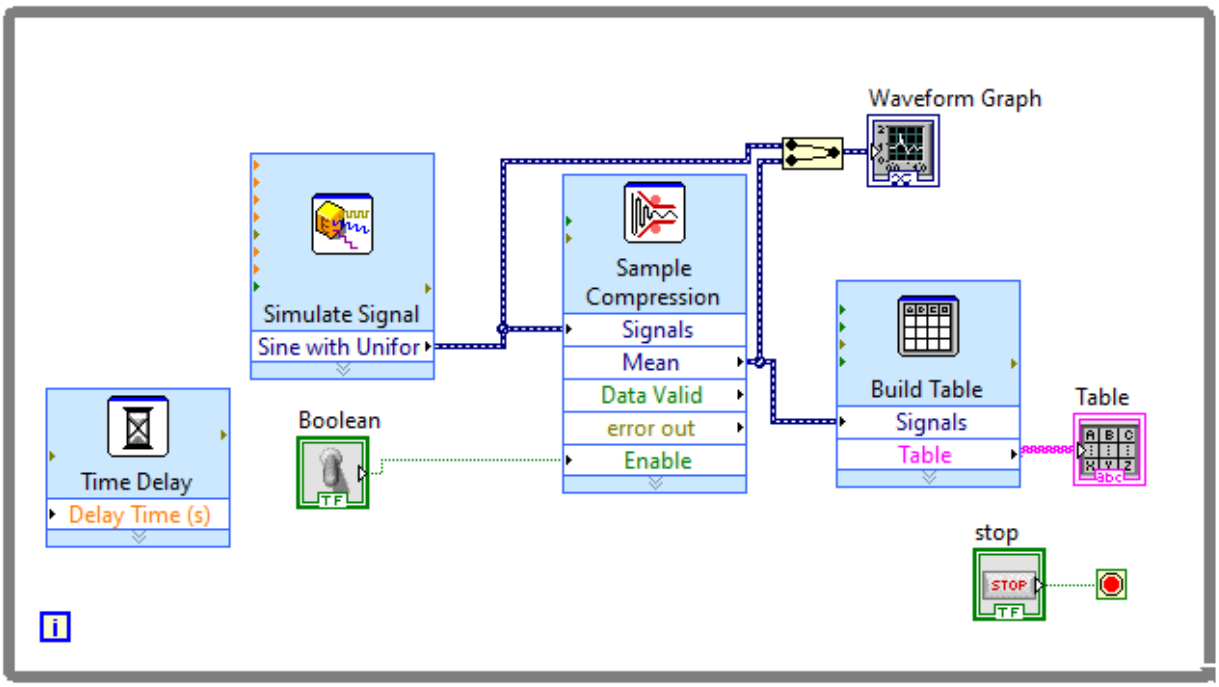


Рис. 1.11. Блок-діаграма ВП для знешумлення сигналу

Алгоритм роботи ВП наступний - ВП генерує простий гармонічний сигнал із завадою. В якості завади виступає білий шум з вказаним розкидом значень (в блоці генерації сигналів позначається як *Амплітуда шуму*), сигнал виводиться на графік та одночасно подається на блок *Sample Compression*, що буде виконувати функцію фільтруючого елементу.

Далі фільтрований сигнал додається на графік та передається на блок формування таблиці *Build Table* і далі на термінал таблиці для відображення на лицьовій панелі.

Параметри сигналу показані на рис. 1.12 і будуть змінені в залежності від варіанту.

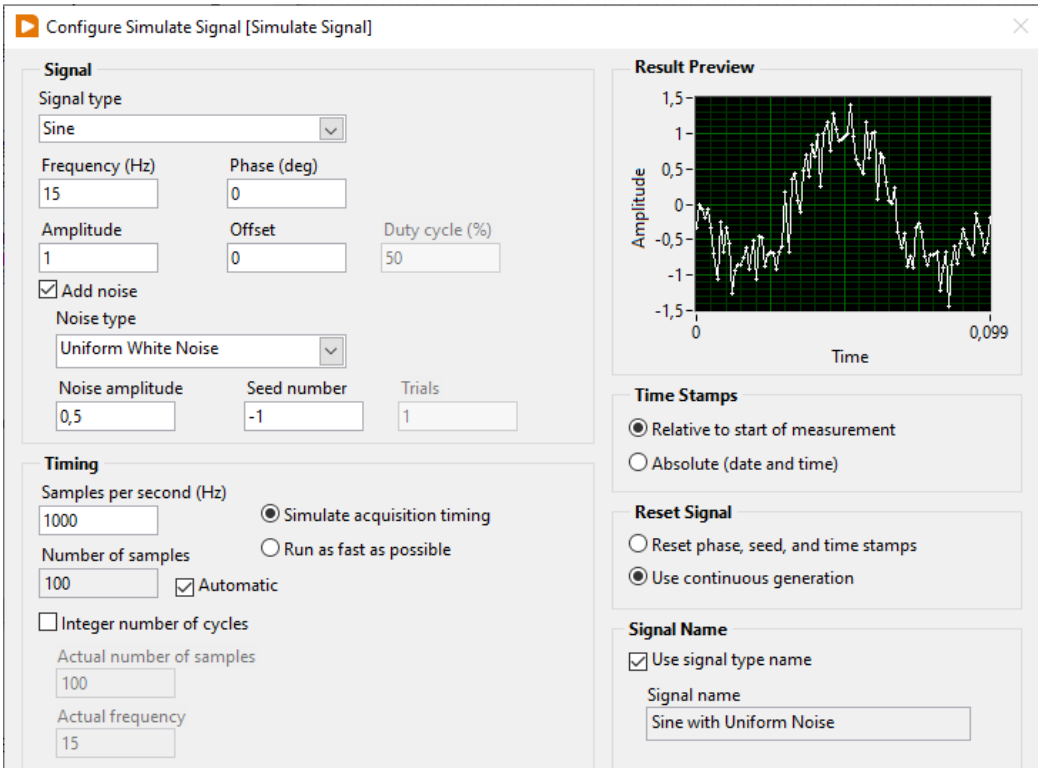


Рис. 1.12. Налаштування блока генерації сигналів

Примітка: Рекомендуємо подивитись вплив на загальний функціонал прикладу таких параметрів: Тип шуму (*Noise type*), його амплітуди, частоти вибірок (*Samples per second*) та їх кількості (*Number of samples*).

Блок *Sample Compression* призначений для зменшення кількості точок вхідного сигналу коли інформація, яка міститься у масиві або надлишкова, або її втрата призведе до більш спрощеної обробки сигналу. Основними параметрами функції є значення фактору зменшення (*Reduction factor*), по суті що є кількістю точок вхідного сигналу по яким буде проводитись зменшення і методи зменшення (*Reduction methods*), серед яких - по мінімальному або максимальному значенню, по останньому на проміжку, середнє та медіанне

значення. Для прикладу оберемо значення фактору зменшення 15, а метод – середнє (Mean), як показано на рис. 1.13.

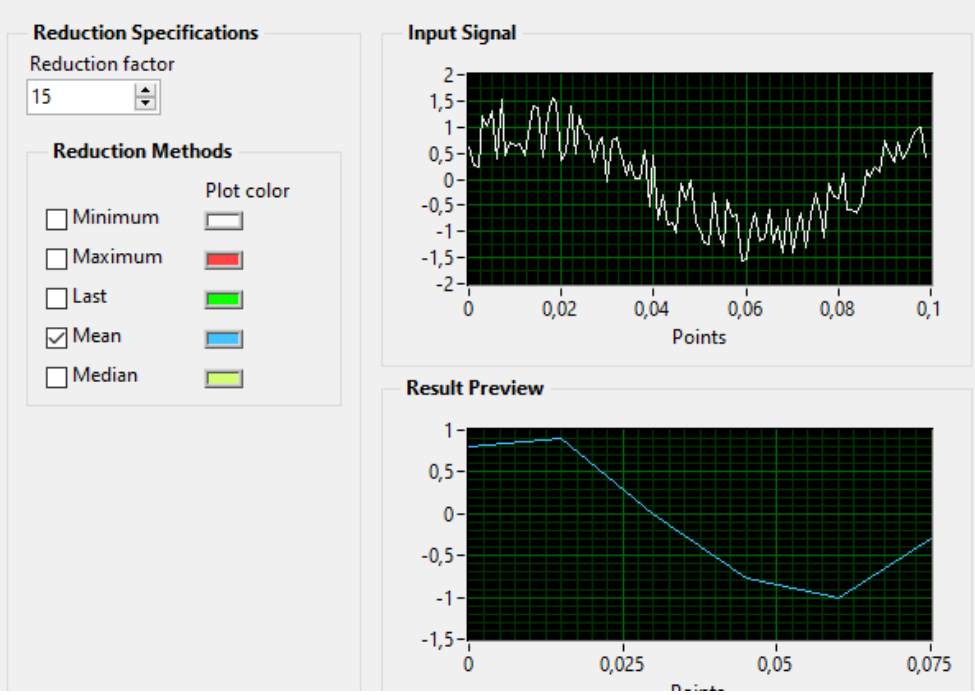


Рис. 1.13. Налаштування блока Sample Compression

Блок формування таблиці *Build Table* не потребує специфічних налаштувань, проте у опціях цього блоку можна обрати формат виведення інформації, його особливості та автоматично вивести колонку з датою отримання інформації, що може бути зручним при виконанні автоматизованих вимірювань.

Оскільки ВП використовує цикл While Loop який працює з максимально можливою швидкістю, що обмежується швидкодією ПК, то для більш плавного відображення інформації застосуємо блок Time Delay із значенням затримки 0,25 с.

1.1.3. Аналіз та збереження сигналу

У наступному прикладі ми створимо ВП, який генерує складний полігармонічний сигнал, фільтрує його, показує, чи перевищує сигнал певні граничні значення і зберігає ці оброблені дані. Після того, як ви завершите вправу, лицьова панель віртуального приладу повинна виглядати так само, як показано на рис. 1.14

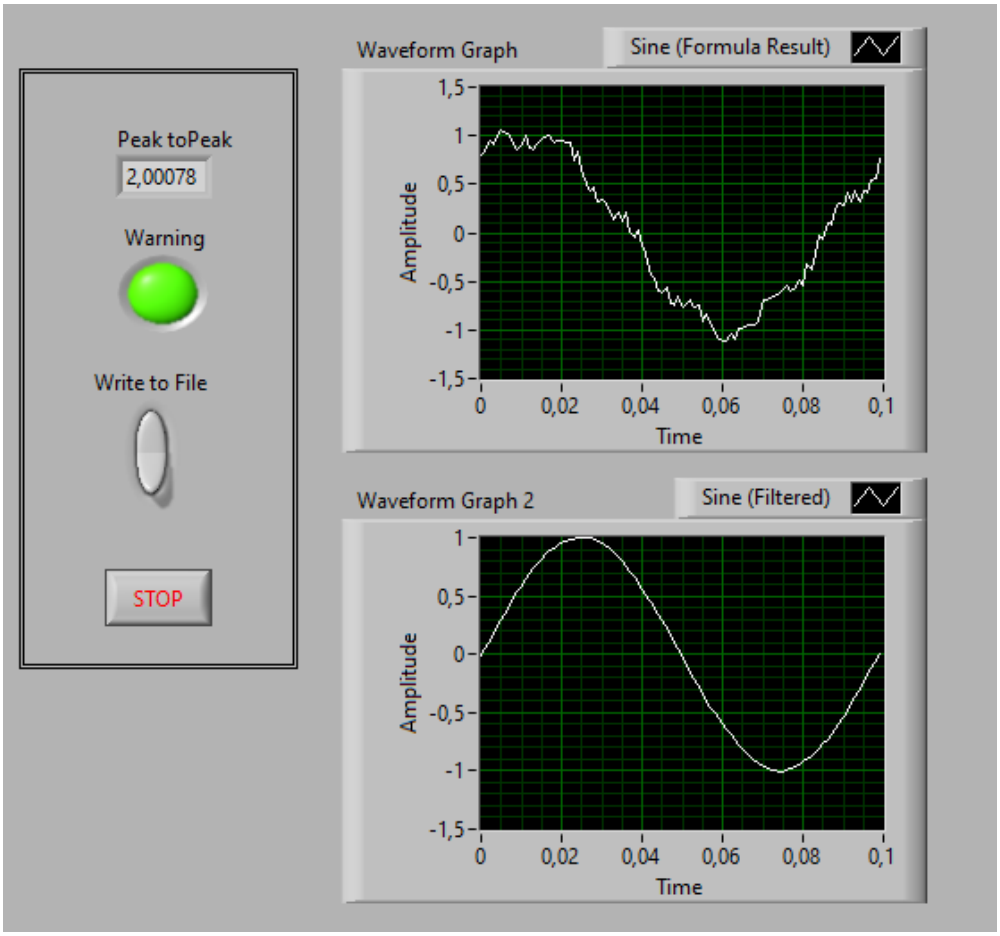


Рис. 1.14. Лицьова панель ВП для аналізу та збереження сигналів

Програма забезпечує наступний функціонал. Блоки генерації формують полігармонічний сигнал з трьох складових: синусоїду з амплітудою *A1= 1* та частотою *f1= 10.1Гц*, синусоїду з амплітудою *A2= 0.1* та частотою *f2= 60Гц*, та білим шумом із амплітудним розкидом, що дорівнює 0.1. Цей сигнал виводиться у графічне вікно. Далі, сигнал подається на блок цифрового фільтру який пригнічує частоти вище 25Гц. Вихідний сигнал подається на блок визначення пікових значень і на блок порівняння, а також виводиться у друге графічне вікно. Якщо розмах сигналу перевищує значення 2 – вмикається індикатор «Warning». В той же час, сигнал після блоку фільтра подається на блок запису у файл який активується перемикачем «Write to File» на лицьовій панелі приладу.

Загальний вигляд панелі блок-діаграм цього прикладу показаний на рис. 1.15.

Розглянемо нові функціональні блоки більш детально.

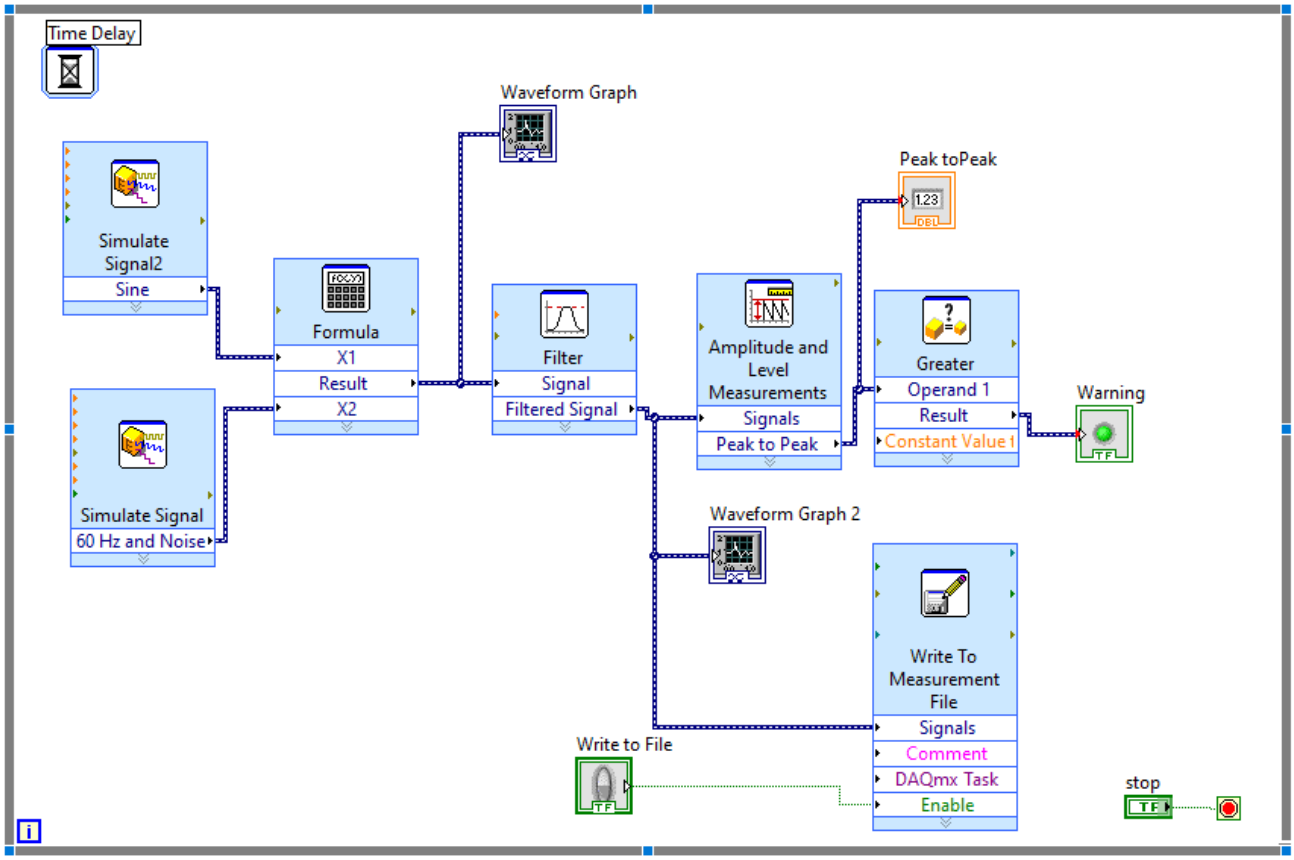


Рис. 1.15. Панель блок-діаграм ВП для аналізу та збереження сигналів

Основні параметри налаштувань блоку цифрового фільтра показані на рис. 1.16. Нагадаємо певний класифікаційний перелік характеристик цифрових фільтрів для подальшого оперування. По виду смуги пропускання (типу фільтра) можна виділити фільтри:

- фільтри нижніх частот (ФНЧ, Lowpass);
- фільтри верхніх частот (ФВЧ, Highpass);
- смугові фільтри (смугово-пропускні, Bandpass);
- режекторні фільтри (смугово-непропускні, Bandstop).

По типу імпульсної характеристики розрізняють фільтри зі скінченою імпульсною характеристикою (КІХ, FIR) та нескінченною імпульсною характеристикою (НІХ, IIR).

Фільтр зі скінченною імпульсною характеристикою (нерекурсивний фільтр) - один з видів цифрових фільтрів, характерною особливістю якого є обмеженість по часу його імпульсної характеристики (з певного моменту часу

вона стає рівною нулю). Такий фільтр називають ще нерекурсивним через відсутність зворотного зв'язку.

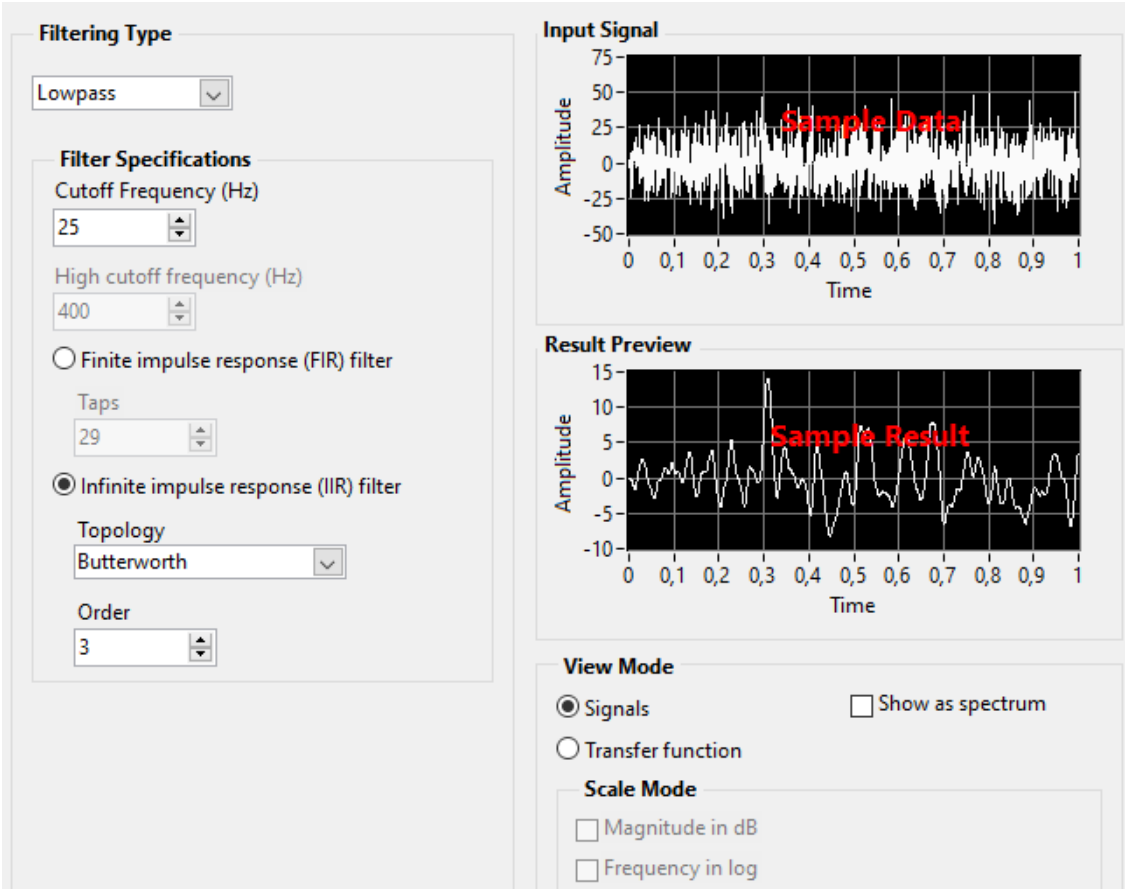


Рис. 1.16. Налаштування блоку Filter

Фільтр з нескінченною імпульсною характеристикою – фільтр, який використовує один або більше своїх виходів у якості входу, тобто утворює зворотний зв'язок. Основною властивістю таких фільтрів є те, що їх імпульсна характеристика має нескінченну довжину у часовій області, а передатна функція має дробово-раціональний вигляд.

В залежності від топології передатної функції, НІХ-фільтри умовно діляться на фільри Чебишева, Батерворта, Бесселя, Еліптичні, відмінності полягають у крутизні імпульсної характеристики в області переходу та коливаннях в зонах пропускання та пригнічення.

В даному прикладі обраний ФНЧ Батерворта з частотою зрізу 25Гц, нескінченною імпульсною характеристикою третього порядку.

Примітка: В правій частині вікна налаштувань (рис. 1.16) можна зручно перемикатись між режимами відображення, наприклад, можна обрати

відображення АЧХ (амплітудно-частотну характеристику) та ФЧХ (фазочастотну характеристику) фільтра, вивести значення амплітуди у Децибелах або відобразити вісь частот у логарифмічному масштабі.

Блок Amplitude and Level Measurment є універсальним блоком з палітри Express для попередньої найпростішої обробки даних. Він може бути замінений на інші наявні функціональні блоки, проте в даному прикладі використаємо саме його, для більш розширеного огляду палітри інструментів Express. Блок має можливість виконати наступні алгоритми обробки сигналів:

- Середнє та середнє циклічне значення (Mean DC/Cycle Average);
- СКВ (середньоквадратичне відхилення) та СКВ за період (RMS / Cycle RMS);
- Відслідковування додатних та від'ємних пікових значень (Positive peak / Negative peak);
- Розмах сигналу (Peak to Peak).

В прикладі обраний варіант відслідковування розмаху вихідного сигналу.

Блок Comparison представляє собою звичайний комбінований блок порівняння, тому детального опису не потребує.

Блок Write to Measurement File, налаштування якого показані на рис. 1.17, ще один блок інструментарію палітри Express який дозволяє доволі гнучко організувати налаштування та запис у файл необхідних даних.

За замовчування, при збереженні у вказаному місці дискового простору утворюється файл із розширенням *.lvm, з одного боку це є спеціалізованим форматом LabVIEW, з іншого – звичайним текстовим файлом з нестандартним розширенням. Окрім файлів *.lvm блок дозволяє формувати двійкові файли з підтримкою xml-скриптів та зберігати в фалах таблиці MS Excel.

Показані на рис. 1.17 налаштування дозволяють обрати опції пакетного збереження при автоматичному створенні декількох файлів із заздалегідь вказаними іменами, або заміною існуючого файлу на новий. При роботі з одним файлом є можливості дописувати блоки даних або заміняти новими. Також можна додати заголовок та часові мітки у записуваних блоках даних.

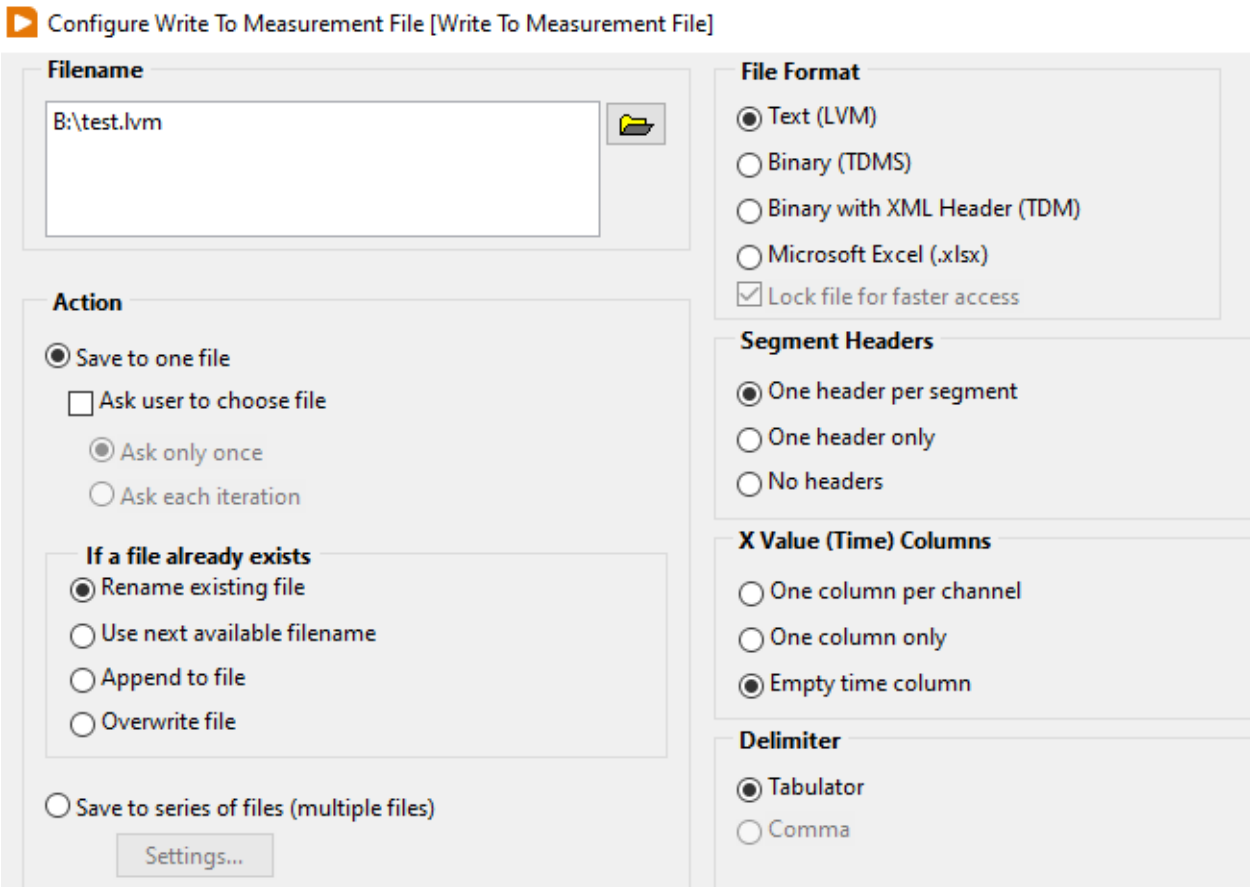


Рис. 1.17. Налаштування блоку Write to Measurement File

1.2. Завдання для виконання

Завдання 1. Створити ВП, як наведено в п.п. 1.1.1.

Для парних варіантів: обрати пилкоподібний сигнал, масштабування сигналу має бути організовано обертовою ручкою із значеннями від 0 до № варіанту +1.

Для непарних варіантів: обрати трикутний сигнал, масштабування сигналу має бути організовано обертовою ручкою із значеннями від № варіанту +1 до № варіанту +10.

Завдання 2. Створити ВП, як наведено в п.п. 1.1.2.

Змінити параметри генерованого сигналу за наступними даними: Амплітуда гармонічного сигналу A=«№ варіанту», частота f = «10*№ варіанту» (Гц), амплітуда шуму $A_ш$=«A/3». Тип шуму обирається в залежності від номеру варіанту:

- 1й - Uniform White Noise;
- 2й - Gaussian White Noise;

- 3й - Periodic Random Noise;
- 4й - Gamma Noise;
- 5й - Poisson Noise;
- 6й - Bernoulli Noise;
- 7й - MLS Sequence;
- 8й - Inverse F Noise.

9й варіант обирає перший вид шуму, 10й другий і т.д.

Підібрати параметри блоку *Sample Compression* таким чином, щоб вплив шуму у вихідному сигналі був мінімальним.

Завдання 3. Створити ВП, як наведено в п.п. 1.1.3.

Згенерувати полігармонічний сигнал за заданими параметрами, що наведені в табл. 1.1. та обрати параметри фільтра для виділення вказаної складової.

Таблиця 1.1. Параметри сигналу для завдання 3.

№ варіанту	Перша складова сигналу	Друга складова сигналу	Третя складова сигналу	Яку складову виділити
1	A_1= 1, f_1= 10 Гц	A_2= 1, f_2= 100 Гц	A_3= 1, f_3= 300 Гц	Другу
2	A_1= 0.5, f_1= 10 Гц	A_2= 1, f_2= 20 Гц	A_3= 0.1, f_3= 60 Гц	Першу
3	A_1= 1, f_1= 100 Гц	A_2= 0.1, f_2= 80 Гц	A_3= 3, f_3= 20 Гц	Третю
4	A_1= 0.5, f_1= 1 Гц	A_2= 1, f_2= 20 Гц	A_3= 0.1, f_3= 60 Гц	Другу
5	A_1= 1, f_1= 1 кГц	A_2= 10, f_2= 100 Гц	A_3= 1, f_3= 300 Гц	Першу
6	A_1= 0.5, f_1= 10 Гц	A_2= 0.1, f_2= 80 Гц	A_3= 0.1, f_3= 200 Гц	Першу
7	A_1= 1.5, f_1= 1 Гц	A_2= 1, f_2= 50 Гц	A_3= 10, f_3= 600 Гц	Третю
8	A_1= 1, f_1= 0.5 кГц	A_2= 10, f_2= 20 Гц	A_3= 1, f_3= 100 Гц	Другу
9	A_1= 0.5, f_1= 100 Гц	A_2= 0.1, f_2= 800 Гц	A_3= 0.1, f_3= 20 Гц	Першу
10	A_1=10, f_1= 10 Гц	A_2= 0.1, f_2= 100 Гц	A_3= 1, f_3= 120 Гц	Другу

Рівень шуму залишити як в прикладі (дорівнює 0.1). Запис в файл має відбуватись із вказанням часу запису і має містити інформацію про виділену складову із полігармонічного сигналу. Інші складові мають бути пригнічені не менше ніж в 10 разів у порівнянні із початковим амплітудним значенням. Частоту зрізу, тип та топологію фільтру обрати самостійно для реалізації якнайкращого результату. Вибір обґрунтувати.

1.3. Зміст звіту

1. Скріни лицьових панелей для завдань з п.п. 1.1.1-1.1.3, що демонструють працездатність віртуального приладу.
2. Скріни панелей блок-діаграм для завдань з п.п. 1.1.1-1.1.3.
3. Скріни лицьових панелей і панелей блок-діаграм з виконаними завданнями за індивідуальним варіантом.
4. Додаткові зображення, що підтверджують виконання завдання відповідно вашого варіанту (налаштування блоків і т.п.).
5. Висновки по роботі.

1.4. Контрольні запитання

1. Дайте визначення віртуальному приладу, Який зв’язок із реальними приладами? Які є особливості?
2. Для чого призначена лицьова панель віртуального приладу?
3. Для чого призначена панель блок-діаграм віртуального приладу?
4. У чому полягають переваги створення ВП із шаблона?
5. Які особливості використання блоків палітри Express? Назвіть недоліки та переваги таких функціональних блоків?
6. Як працює блок Sample Compression?
7. Опишіть основні налаштування блоків генерації сигналів?
8. Як зберегти данні у файл? Які параметри при збереженні можна вказувати?
9. Як обрати фільтр? Опишіть основні характеристики при налаштуванні блоку фільтра палітри Express.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 2
СТВОРЕННЯ ПРОСТИХ ВІРТУАЛЬНИХ ПРИЛАДІВ. ФУНКЦІЇ ВИБОРУ

Мета роботи: ознайомлення основним інструментарієм середовища NI LabVIEW створення віртуальних приладів. Елементи вибору, функція Select та структура Case.

2.1. Теоретичні відомості

2.1.1. Функціональний блок Select

Невід'ємною частиною будь-якої мови програмування є елементи розгалудження, що надають можливості вибору та варіативності, не є виключенням і середовище LabVIEW. Проте з урахуванням особливостей для таких задач реалізований різний інструментарій. Найпростішим функціональним блоком вибору є Select, загальний вигляд і позначення входів/виходів показаний на рис. 2.1. Цей функціональний блок знаходиться в меню *Programing – Comparison.*

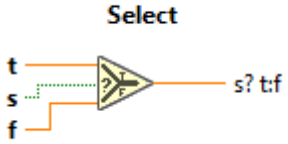
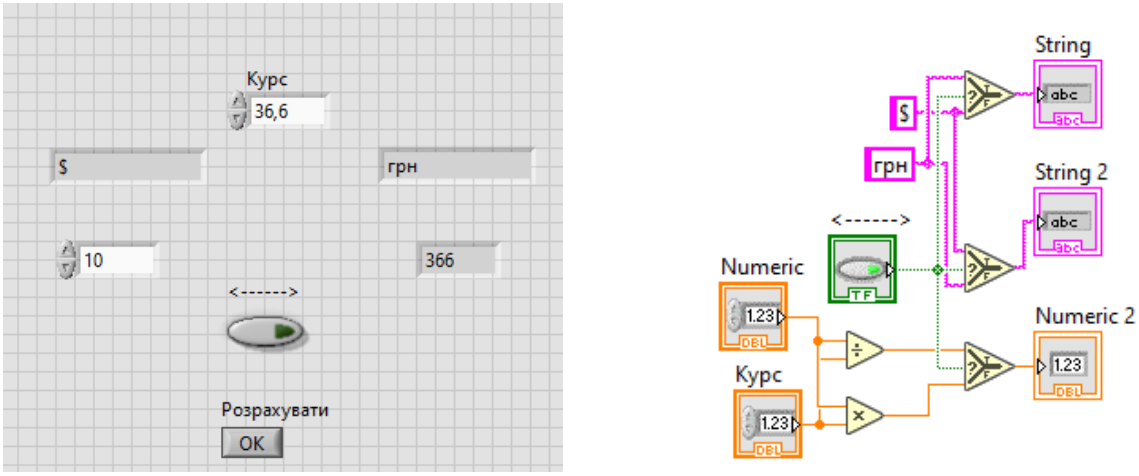


Рис. 2.1. Блок Select

Блок працює за наступним алгоритмом: на бульовий керуючий вхід **s** приходить значення True або False, в залежності від чого до виходу приєднується вхід **t** або **s**. На інформаційні входи **t** та **s** можуть подаватись данні будь-якого типу, але попарно. Тобто, по суті функція Select це аналог if-умови у мовах програмування.

Примітка: не дивлячись на те, що подача бульових типів даних на входи **t** та **s** не заборонена, але така дія немає сенсу, оскільки вихідними значеннями будуть True або False, що співпадає із значеннями на керуючому вході.

Для прикладу створимо ВП «Конвертор валют», що дозволяє проводити розрахунок при конвертуванні значень з гривень у долари та навпаки. Лицьова панель даного віртуального приладу показана на рис. 2.2, а, а блок-діаграма на рис. 2.2, б.



а) б)

Рис. 2.2. ВП «Конвертор валют»: а) Лицьова панель; б) Блок-діаграма

Для введення даних використовується керуючий числовий елемент Numeric Control, для виведення результату поля Numeric Indicator. Для виведення написів «грн» та «$» використовуються строкові об’єкти String Indicator. Ці поля доступні в меню String & Path, але дублюються в різних категоріях як Classic, Modern, System, Silver, вони мають однаковий функціонал проте відрізняються дизайном.

Якщо уважно проаналізувати блок-діаграму ВП із рис. 2.2, б, то легко помітити, що дроти змінюють свій колір в залежності від типів даних які вони передають. Така ж кольорова індикація притаманна і терміналам об’єктів. Відтак, основними і найпростішими типами даних в середовищі LabVIEW є:

- Цілочисельний (Integer I32,I16 і т.п.) позначається синім кольором;
- Дійсний, з плаваючою комою (Float, Double) – помаранчевий;
- Бульовий, логічний (Boolean) – зелений;
- Строковий (String) – темно рожевий.

Зауважимо, що оскільки враховуючи специфіку графічного програмування в середовищі LabVIEW існує багато інших специфічних типів

даних, проте про них ми поговоримо пізніше, зараз були наведені лише основні типи, що здебільшого повністю співпадають із типами даних класичних мов програмування.

2.1.2. Структура вибору Case

Другим аналогом if- умови, що найчастіше використовується при реалізації віртуальних приладів є структура вибору Case. Вона розташована у меню *Programming – Structures* і представляє собою набір субдіаграм які обираються і виконуються в залежності від сигналу на керуючому вході. Вигляд стандартної структура вибору Case показаний на рис. 2.3.

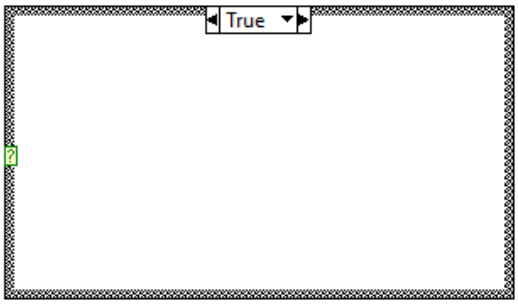


Рис. 2.3. Структура вибору Case

За замовчуванням структура Case має два варіанти субдіаграм True та False, відповідна діаграма обирається в залежності від логічного сигналу, що надходить на керуючий вхід на грані структури – Case Selector, друга субдіаграма ігнорується.

Якщо на Case Selector подати значення іншого типу даних, наприклад цілочисельного, то варіанти субдіаграм «True» та «False» змінять назву на «0, Default» та «1», що дасть можливість збільшувати кількість субдіаграм як показано на рис. 2.4.

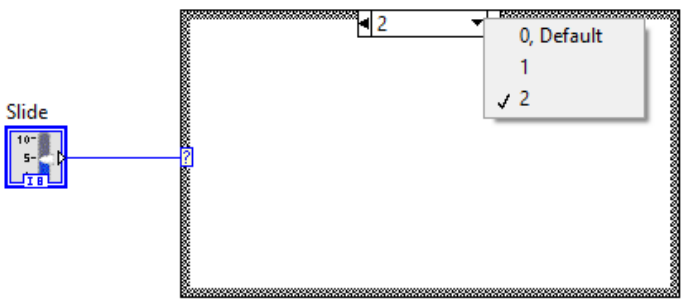


Рис. 2.4. Модифікована структура вибору Case

Для додавання субдіаграм в структуру необхідно по рамці структури Case клікнути правою кнопкою миші і в контекстному меню вибрати Add Case After або подібний варіант.

Таким чином, найближчим аналогом структури Case є інструкція Switch або if-else-if.

Для закріплення знань створимо ВП «Калькулятор» який має виконувати наступні дії: додавати, віднімати, множити та ділити два введені числа. Вміти порівнювати введені значення і видавати результат у вигляді індикації >, <, =. Для перемикання між режимами обрахунку і порівняння має бути передбачений перемикач. Якщо обраний режим розрахунку то всі індикатори вимкнені. Якщо режим порівняння – в полі результат завжди 0. Лицьова панель та блок-діаграма показані на рис. 2.5 та рис. 2.6 відповідно.

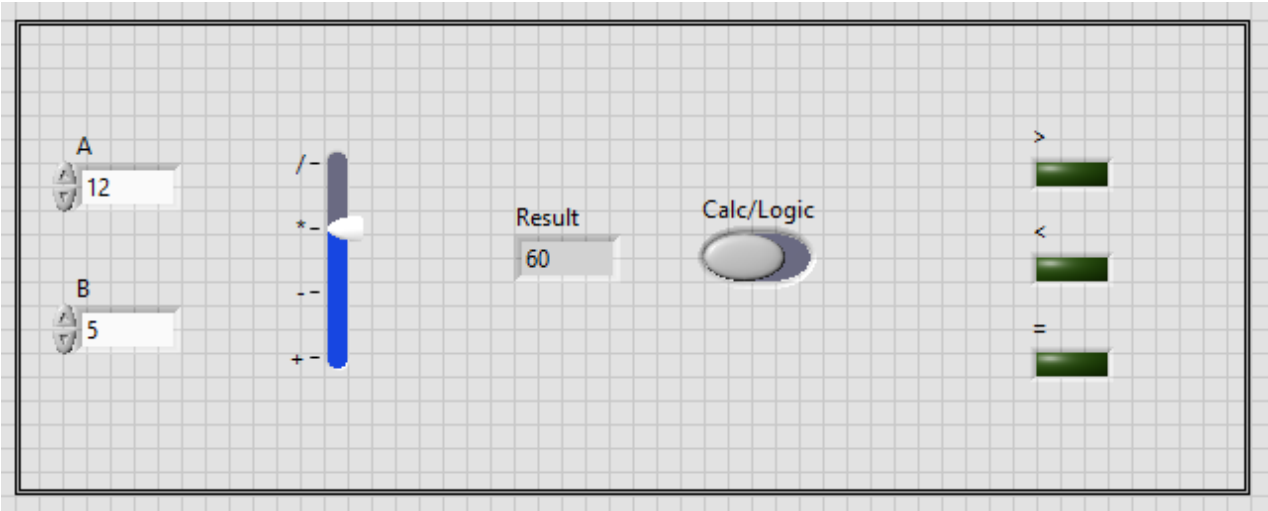


Рис. 2.5. ВП «Калькулятор»

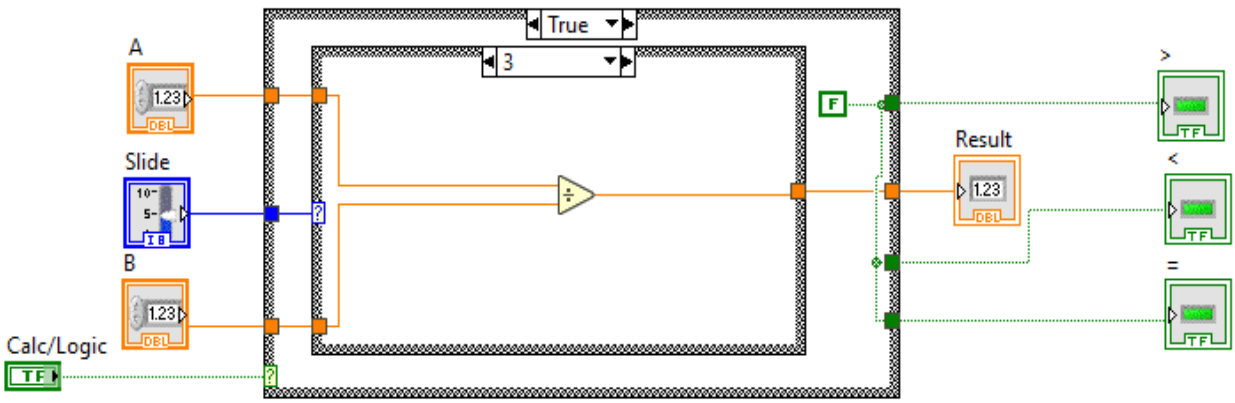


Рис. 2.6. Блок-діаграма ВП «Калькулятор»

2.1.3. Блоки властивостей

Для взаємодії з властивостями будь-якого функціонального об’єкта використовуються блоки властивостей Property Node. Не дивлячись на те, що цей блок властивостей можна знайти і в меню інструментів, має сенс використовувати контекстне меню необхідного об’єкта. Наприклад, в контекстному меню світлодіода «>» з рис. 2.5 оберемо *Create – Property Node – Visible*. На панелі блок-діаграм з’явиться блок властивостей обраного об’єкта з активною властивістю Visible. Проте ця властивість сконфігурована на видачу інформації, для її зміни треба з контекстного меню даного блоку обрати Change All to Write, що призведе до зміни режиму роботи блока. Якщо на вхід цього блока подати бульове значення False, як показано на рис. 2.7 а, то світлодіод зникне з лицьової панелі. Причому функціонувати ВП продовжить, бо світлодіод став невидимий, а не видалений. Для повернення світлодіода у «звичайний» стан достатньо змінити значення на вході блоку властивостей.

Примітка: При необхідності використовувати одразу декілька властивостей одного і того ж об’єкту, достатньо потягнути за нижню рамку блоку властивостей, а далі обрати необхідні властивості (див. рис. 2.7,б). Так само можна змінювати конкретні властивості на читання або запис (якщо таке можливо, деякі властивості не можуть бути переконфігуровані).

а) б)

Рис. 2.7. Блок властивостей об’єкта: а) Обрано одну властивість; б) Обрано декілька властивостей

2.2. Завдання для виконання

Створити ВП, як наведено в п.п. 2.1.1. та п.п. 2.1.2.

Для парних варіантів: Модифікувати ВП «Конвертор валют». Написи «грн» та «$» мають відображатись у властивостях об'єкта Caption (або Label) використовуючи блоки властивостей. Текстові поля не видаляти, а приховати відповідними властивостями.

Для непарних варіантів: Модифікувати ВП «Калькулятор». При діленні на 0 має бути відображений відповідний напис. Використовувати блок One Button Dialog. В режимі калькулятора світлодіоди порівняння мають зникати, в режимі порівняння поле результату має ставати неактивним.

2.3. Зміст звіту

1. Скріни лицьових панелей для завдань із п.п. 2.1.1. та п.п. 2.1.2., що демонструють працездатність віртуального приладу.
2. Скріни панелей блок-діаграм для завдань із п.п. 2.1.1. та п.п. 2.1.2.
3. Скріни лицьових панелей і панелей блок-діаграм з виконаними завданнями за індивідуальним варіантом.
4. Додаткові зображення, що підтверджують виконання завдання відповідно вашого варіанту (налаштування блоків і т.п.).
5. Висновки по роботі.

2.4. Контрольні запитання

1. Як працює блок Select? Які аналогії із класичними мовами програмування ви можете привести?
2. З якими типами даних працює блок Select?
3. Які основні типи даних в LabVIEW ви знаєте, та як їх відрізнити на панелі блок-діаграм?
4. Як працювати із структурою вибору Case?
5. Як змінити кількість субдіаграм структури Case та як ними керувати?
6. Для чого існують блоки властивостей та як їх створювати?
7. Чи можна одночасно керувати декількома властивостями використовуючи один блок Property Node?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 3
РОБОТА ІЗ ЦИКЛАМИ

Мета роботи: ознайомлення з циклами в середовищі LabVIEW, здобуття навичок роботи зі структурами циклів For Loop та While Loop.

3.1. Теоретичні відомості

Цикли є важливою частиною будь-якої мови програмування і однієї із найбільш часто використовуваних конструкції (поряд із умовами). Як і в класичних мовах програмування, в середовищі LabVIEW використовується два види циклів - For Loop та While Loop.

Цикл While Loop є циклом із невизначеною кількістю ітерацій який працює при виконанні певної умови, за яку відповідає окремий термінал Loop Condition. Цикл While Loop знаходиться у палітрі Programing - > Structures. Оскільки цикли в LabVIEW представлені у вигляді рамок структур, то відповідно частина блок-діаграми, що знаходиться в середині рамки цикла буде вважатись його тілом і повторюватись відповідну кількість разів.

Блок-діаграма циклу While, що у загальному вигляді показана на рис. 3.1 виконується до тих пір, поки не виконається умова виходу з циклу. За замовчуванням, термінал умови виходу має вигляд  (рис. 3.1, а).

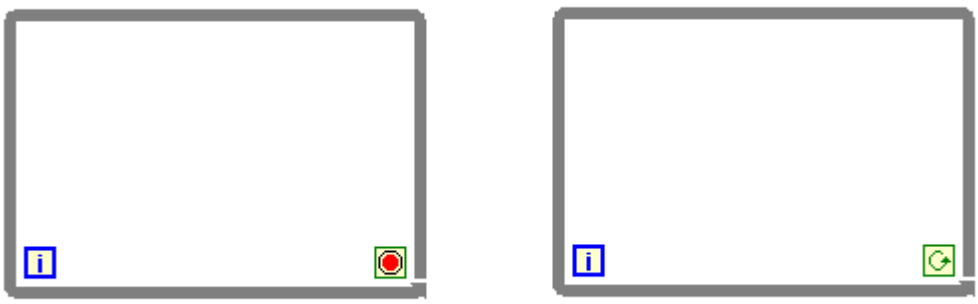


Рис. 3.1. Загальний вигляд циклів While Loop на панелі блок-діаграм:
а) Термінал зупинки в режимі Stop If True; б) Термінал зупинки в режимі Continue If True

Це означає, що цикл буде виконуватися до надходження на термінал умови виходу значення TRUE. У цьому випадку термінал умови виходу

працює в режимі Stop If True, для перемикання в альтернативний режим Continue If True (рис. 3.1, б) треба викликати контекстне меню терміналу і обрати відповідну функцію.

Обов'язковою частиною циклу є також термінал лічильника [i], містить значення кількості виконаних ітерацій. Початкове значення терміналу завжди дорівнює нулю.

Для прикладу реалізуємо ВП, який працює за наступним алгоритмом. Користувач загадує число від 0 до 10000. Програма генерує випадкове число із зазначеного діапазону. Якщо значення не співпадають, програма загадує наступне число, і так до тих пір поки не випаде шукане значення. Після цього ВП зупиняється, а на лицьовій панелі виводиться кількість ітерацій і шукане значення. Лицьова панель цієї програми показана на рис. 3.2, а блок-діаграма – на рис. 3.3.

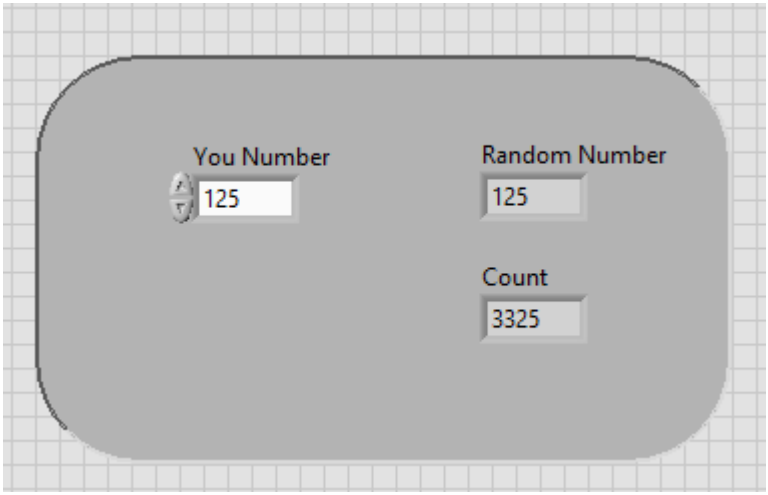


Рис. 3.2. Лицьова панель тестового ВП із циклом While Loop

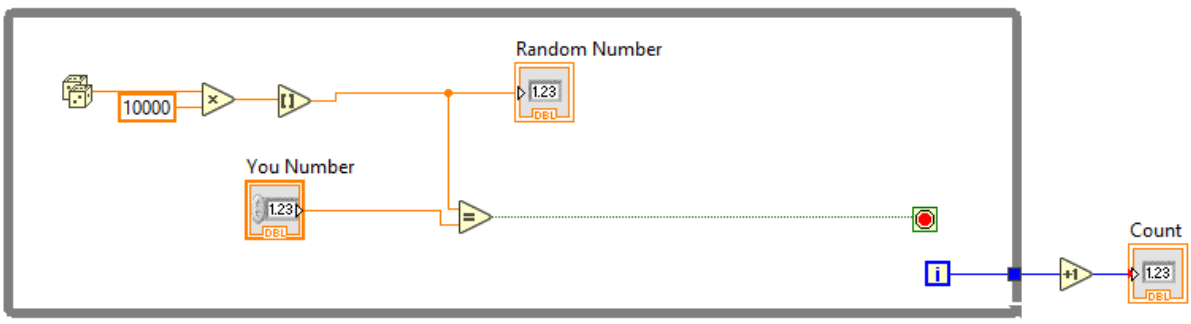


Рис. 3.3. Панель блок-діаграм тестового ВП із циклом While Loop

Примітка: Подумайте, як можна змінити програму без втрати її функціоналу змінивши режим роботи терміналу зупинки з Stop If True на Continue If True?

Примітка: Для обмеження діапазону введених значень у поле You Number достатньо встановити діапазони у меню Properties на вкладці *Data Entry* як показано на рис. 3.4.

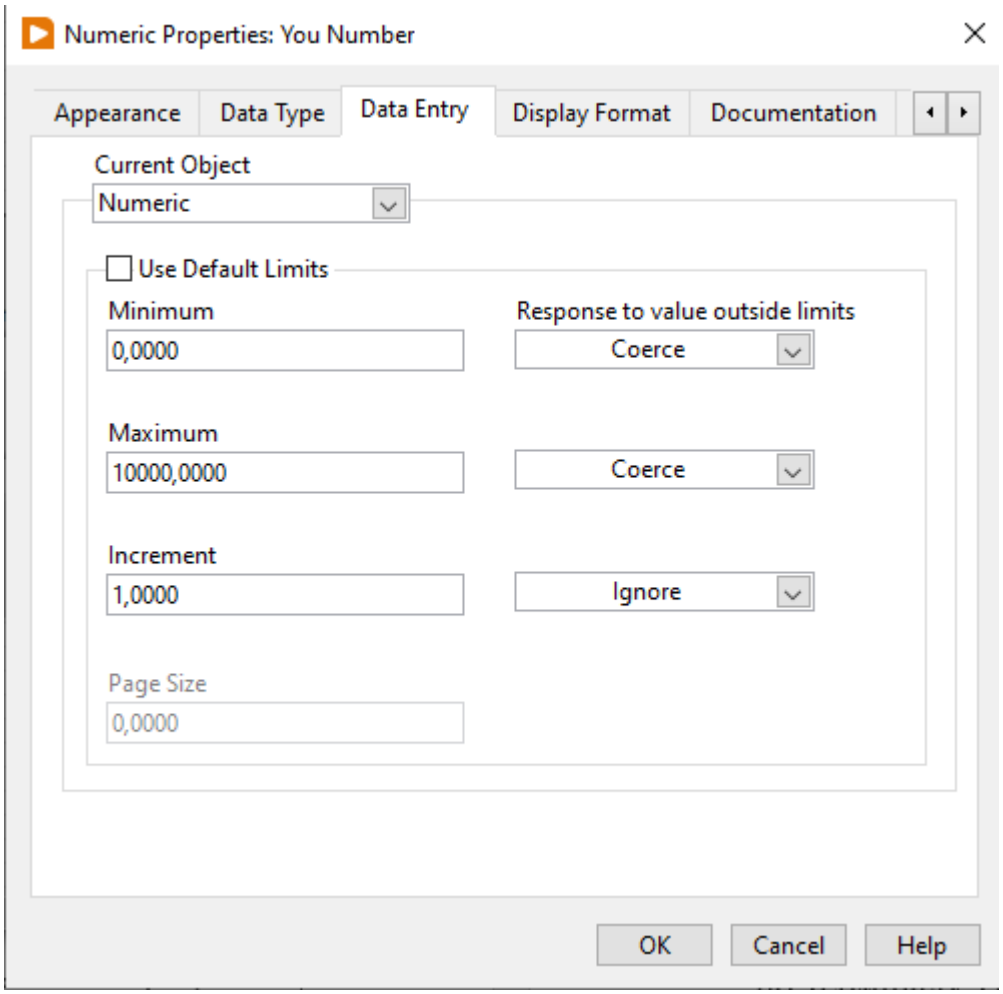


Рис. 3.4. Вкладка Data Entry меню властивостей поля You Number

Вхідні та вихідні тунелі циклів. Дані можуть надходити в цикл While Loop (або виходити з нього) через тунелі вхідних / вихідних даних циклу (вхідні і вихідні тунелі). Ці тунелі передають дані зовні в структуру і з структури назовні циклу. Вони відображаються у вигляді суцільних прямокутників на межі області циклу While Loop. Прямокутник приймає колір типу даних, переданих по терміналу тунелю. У разі якщо дані надходять в цикл

While Loop через вхідний тунель, виконання циклу починається саме з моменту надходження цих даних. У випадку, який показаний на рис. 3.3, термінал лічильника ітерацій приєднаний до вихідного тунелю циклу. Значення з цього тунелю будуть подані на термінал відображення Count тільки після завершення циклу While Loop. Такий режим функціонування вихідного тунелю має назву Last Value і є режимом роботи за замовчуванням для циклів While Loop.

Примітка: Режим роботи вихідного тунелю можна змінити у контекстному меню в підпункті Tunnel Mode.

Цикл For. Цикл For (For Loop) (з фіксованим числом ітерацій) виконує повторювані операції над потоком даних певну кількість разів він знаходиться у палітрі Programing - > Structures. Принциповою відмінністю For Loop від структури циклу While Loop є відсутність умови завершення циклу, оскільки цикл завершується автоматично після виконання заданої кількості ітерацій, та наявністю терміналу кількості ітерацій.

На рис. 3.5 та рис. 3.6 показана тестова програма яка демонструє приклад роботи із циклом For Loop, програма генерує випадкові значення (100 точок) в діапазоні від 0 до 10 та виводить їх у вигляді графіка.

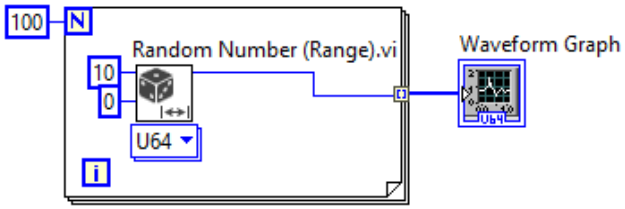


Рис. 3.5. Блок-діаграма яка демонструє роботу із циклом For Loop

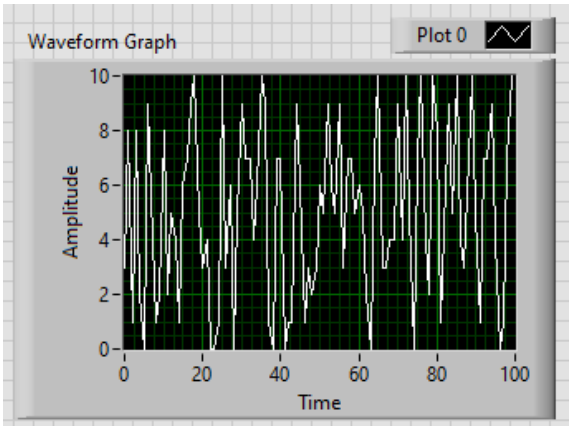


Рис. 3.6. Лицьова панель тестової програми для роботи із циклом For Loop

Ще однією відмінністю циклів For Loop і While Loop є відмінність режимів роботи вихідних тунелей, для циклу For Loop за замовчуванням встановлений режим роботи автоіндексації, який дозволяє накопичувати масив на границі циклу і після виконання всіх ітерації передавати його назовні, що і продемонстровано на рис. 3.5.

Організація доступу до значень попередніх ітерацій циклу. При роботі з циклами часто необхідний доступ до значень попередніх ітерацій циклу. В середовищі LabVIEW з врахування особливостей графічного програмування це реалізовано декількома шляхами: Shift Register (зсувний регістр) і Feedback Node (вузол зворотного зв'язку).

Зсувний регістр виглядає як пара терміналів  , які розташовані безпосередньо один проти одного на протилежних вертикальних сторонах границі циклу. Правий термінал містить стрілку «вгору» і зберігає дані при завершенні поточної ітерації. LabVIEW передає дані з цього регістра в наступну ітерацію циклу. Зсувний регістр створюється викликом контекстного меню по границі циклу і вибором пункту *Add Shift Register*.

Зсувний регістр передає дані будь-якого типу, він автоматично приймає тип перших, які надійшли на нього, даних. Дані, що передаються на термінали зсувного регістру, повинні бути одного типу (або сумісними).

Щоб ініціалізувати зсувний регістр, необхідно передати на його лівий термінал будь-яке значення ззовні циклу. Якщо не ініціалізувати зсувний регістр, він використовує значення, записане в регістр під час останнього виконання циклу або значення, яке використовується за замовчуванням для даного типу даних, якщо цикл ніколи не виконувався.

Цикл з неініціалізованим зсувними регістром використовується при неодноразовому запуску ВП для присвоєння вихідному значенню зсувного регістру значення, взятого з останнього виконання ВП. Щоб зберегти інформацію про стан між наступними запусками ВП, слід залишити вхід лівого терміналу зсувного регістру не визначеним. Після завершення виконання циклу останнє значення, записане в регістр, залишиться на правому терміналі. При подальшій передачі даних з циклу через правий термінал буде передано останнє значення записане в регістр.

Якщо треба отримати доступ до даних, що були декілька ітерацій назад, то в таких випадках використовується стек зсувних регістрів. Для цього треба потягнути за нижню границю вхідного (лівого) зсувного регістра і таким чином створити необхідну кількість комірок стеку. Так, на рис. 3.7 показана блок-діаграма із використанням стеку з доступом до трьох ітерацій.

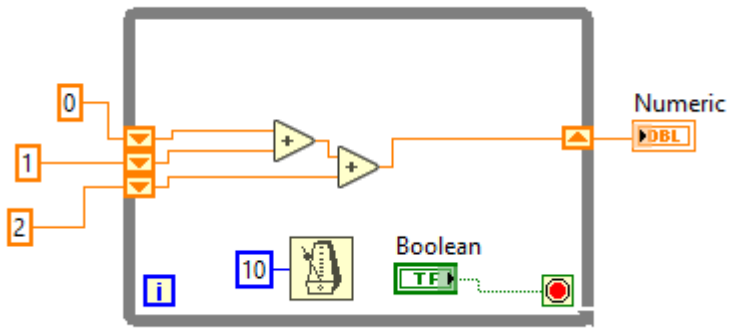


Рис. 3.7. Блок-діаграма, яка демонструє приклад роботи з стеком зсувних регістрів

Тепер розглянемо вузол зворотного зв’язку (Feedback Node). Вузол зворотного зв'язку автоматично з'являється в циклах While Loop або For Loop при з'єднанні поля виводу даних підпрограми ВП, функції або групи підпрограм ВП і функцій з полем введення даних тих же самих підпрограм ВП, функцій або їх груп (реалізація зворотного зв’язку). Як і зсувний регістр, вузол зворотного зв'язку зберігає дані будь-якого типу при завершенні поточної ітерації і передає ці значення в наступну ітерацію. Використання вузлів зворотного зв'язку дозволяє уникнути великої кількості провідників даних і з'єднань. Створити вузол зворотного зв’язку можна звернувшись до палітри *Structures - Feedback Node*. Або перетворити із зсувного регістра обравши меню Replace With Feedback Node.

Примітка: Вузли зворотного зв’язку і зсувні регістри є взаємозамінними за виключення стеку зсувного регістру. Стек не підтримується вузлом зворотного зв’язку.

Для прикладу реалізуємо програму яка по черзі виводить значення квадратів 2 до 2^{10}. Використаємо вузол зворотного зв’язку і зсувні регістри. Додатково використаємо стек зсувного регістра для отримання доступу до

даних з попередніх ітерацій. Лицьова панель і блок-діаграма показані на рис. 3.8 та рис. 3.9 відповідно.

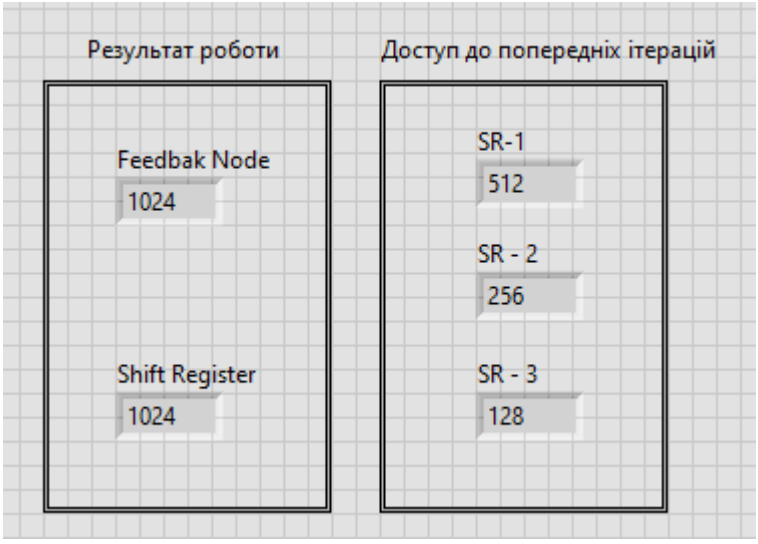


Рис. 3.8. Лицьова панель тестового прикладу, яка демонструє приклад доступу до попередніх ітерацій

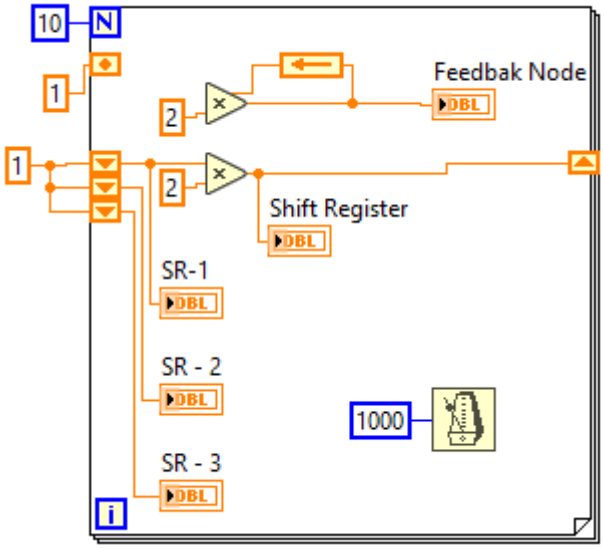


Рис. 3.9. Блок-діаграма ВП яка демонструє приклад доступу до попередніх ітерацій

3.2. Завдання для виконання

Створити ВП, як наведено в п.3.1.

Для парних варіантів: Створити ВП, який підраховує факторіал числа введеного з клавіатури. Реалізувати програму з використання зсувного регістру і, окремо, з використання вузла зворотного зв'язку.

Для непарних варіантів: Створити ВП, який використовуючи принцип дихотомії(поділу навпіл) має вгадати загадане гравцем число з діапазону від 0

до 100 за 7 ітерацій. Спілкування з гравцем реалізувати за допомогою діалогового вікна з двома варіантами відповіді. Наприклад, програма задає наступне питання: «Ваше число більше 50?» – Варіанти відповіді Так, або Ні. Використовувати зсувні регістри або вузли зворотного зв'язку.

3.3. Зміст звіту

1. Скріни лицьових панелей для завдань з п. 3.1., які демонструють працездатність віртуальних приладів.
2. Скріни панелей блок-діаграм для завдань п.3.1.
3. Скріни лицьових панелей і панелей блок-діаграм з виконаними завданнями за індивідуальним варіантом.
4. Додаткові зображення, що підтверджують виконання завдання відповідно вашого варіанту (налаштування блоків і т.п.).
5. Висновки по роботі.

3.4. Контрольні запитання

1. Опишіть відмінності циклу While Loop в середовищі LabVEW від циклів, які ви використовували в класичних мовах програмування.
2. Опишіть відмінності циклу For Loop в середовищі LabVEW від циклів, які ви використовували в класичних мовах програмування.
3. Які відмінності між циклами While Loop та For Loop ?
4. Опишіть навіщо використовуються вхідні та вихідні тунелі та які особливості їх використання у циклах While Loop та For Loop ?
5. Чи можна змінювати режими функціонування вихідних тунелей, в яких випадках це необхідно?
6. Дайте визначення зсувному регістру, в яких випадках вони використовуються?
7. Дайте визначення вузлу зворотного зв'язку, в яких випадках вони використовуються?
8. Чи є відмінності між вузлами зворотного зв'язку і зсувними регістрами?
9. Що таке стек зсувних регістрів?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 4
МАСИВИ ТА КЛАСТЕРИ

Мета роботи: ознайомлення з принципами роботи з масивами та кластерами для створення віртуальних приладів.

4.1. Теоретичні відомості

4.1.1. Масиви

Масиви об'єднують в собі елементи одного типу даних. Масив – це набір елементів певної розмірності. Він може мати одну і більше розмірностей, і до 2^{31}-1 елементів у кожному напрямку, наскільки дозволяє оперативна пам'ять.

Дані, що становлять масив, мають бути будь-якого типу: чисельного, логічного або рядкового. Масив також може містити елементи графічного представлення даних та кластери. Використовувати масиви зручно при роботі з групами даних одного типу і при накопиченні даних після циклічно повторюваних обчислень. Вони підходять для зберігання даних, отриманих з графіків, або накопичених під час роботи циклів.

Не можна створити масив, який складається з масивів. Однак можна створити масив кластерів, де кожен кластер складатиметься з одного або більше масивів.

Примітка: Усі елементи масиву упорядковані. Нумерація елементів масиву завжди починається з 0. Таким чином, індекси масиву знаходяться в діапазоні від 0 до (N-1), де N - кількість елементів масиву.

Вцілому теорія щодо масивів і їх елементів всередовищі LabVIEW збігається із визначеннями класичних мов програмування, а особливості стосуються лише графічного представлення, на що буде зроблений основний акцент в даній роботі.

На лицьовій панелі масиви виглядають як показано на рис. 4.1. Умовно можна виділити масиви індикаторного типу і керуючого. Хоча вигляд таких масивів може бути однаковим, за замовчуванням в масиві керуючого типу присутній елемент керування (рис. 4.2), що дозволяє обрати індекс для

початку відображення елементів масиву. Також цей керуючий елемент використовується для відображення слоїв тривимірних масивів Як показано на рис. 4.3.

Для створення масиву на лицьовій панелі необхідно розташувати шаблон масиву з меню *Classic/Modern – Data Containers – Array*, після чого додати об’єкт необхідного типу.

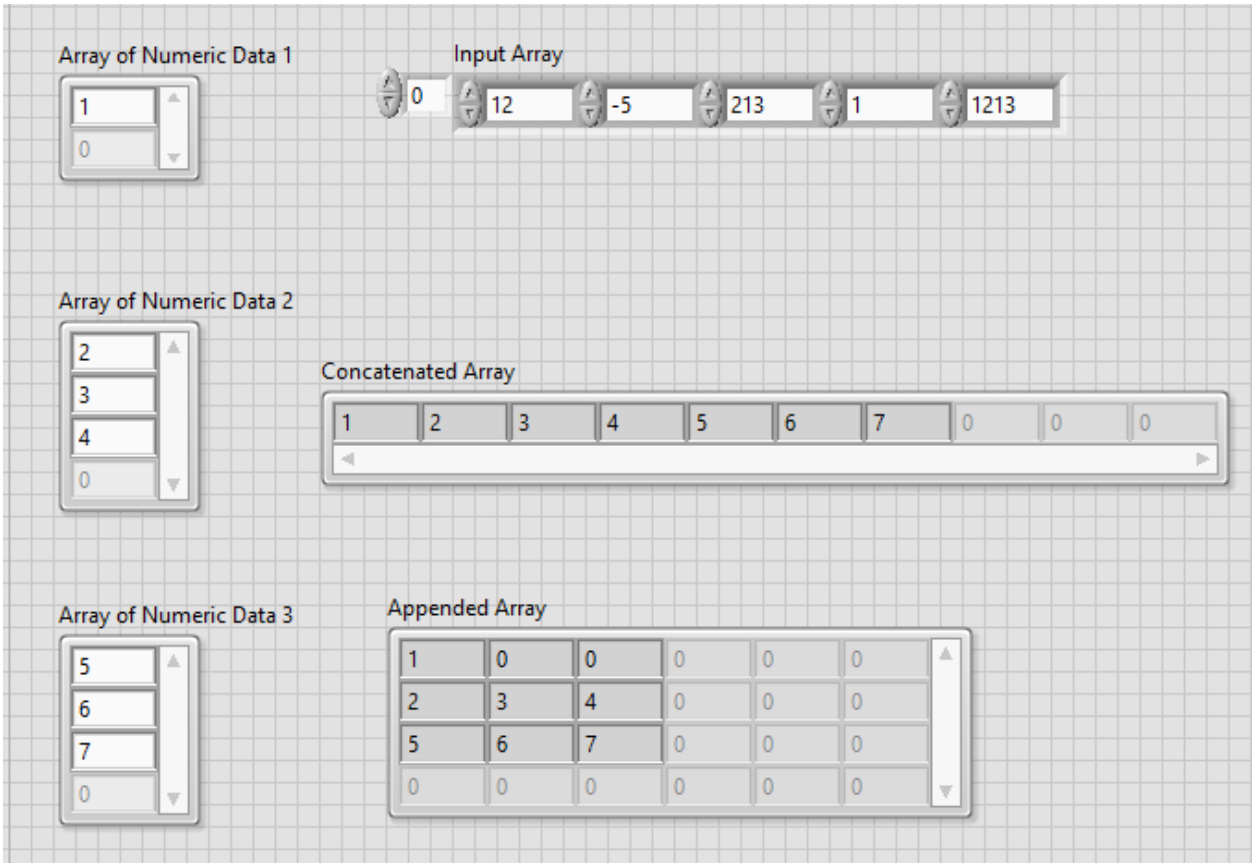


Рис. 4.1. Зовнішній вигляд та різновиди масивів на лицьовій панелі

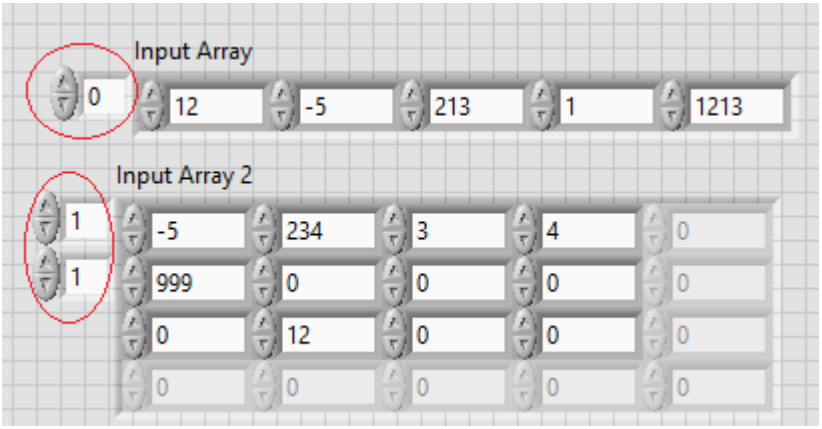


Рис. 4.2. Керуючі елементи для відображення початкового значення масиву

Примітка: Елементи керування відображенням не змінюють розмірність та кількість елементів в масиві, вони керують лише відображенням масиву на лицьовій панелі.

Примітка: Помістити об'єкт у шаблон масиву слід, перш ніж використовувати його на блок-діаграмі. Якщо цього не зробити, то шаблон масиву не буде ініціалізований і використовувати масив буде не можна.

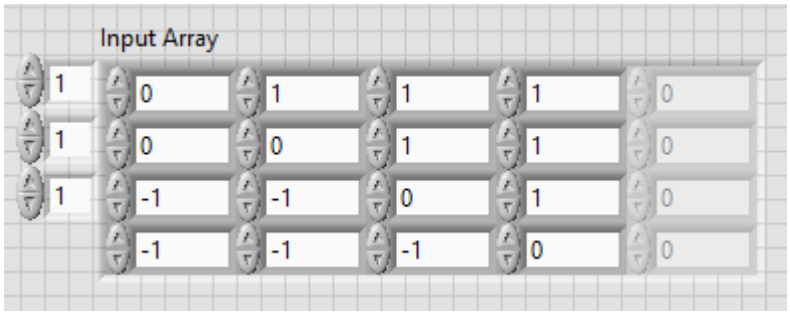


Рис. 4.3. Тривимірний масив із елементами керування відображенням

Для того, щоб гарантовано відрізнити масиви індикаторного і керуючого типів слід звернутись до блок-діаграми. Крім того, за блок-діаграмою можна по вигляду з'єднувального дроту відрізнити одновимірні і багатовимірні масиви і одинарні змінні як показано на рис. 4.4. Треба зауважити, що колір дроту та рамки терміналів буде змінюватись в залежності від типів даних з якими оперує масив.

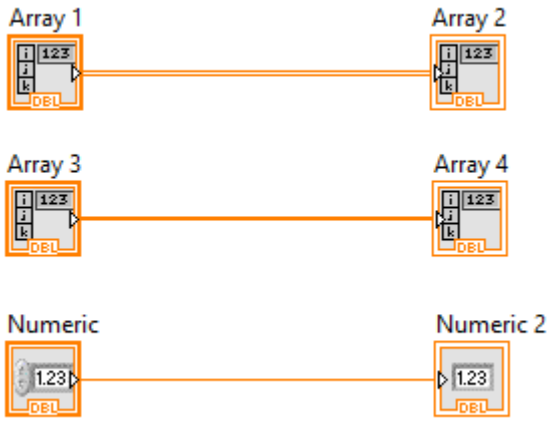


Рис. 4.4. Відмінність відображення терміналів та дротів масивів і звичайних даних

Для роботи із масивами та їх елементами в середовищі LabVIEW представлена велика палітра інструментів в меню Programing – Array як показано на рис. 4.5.

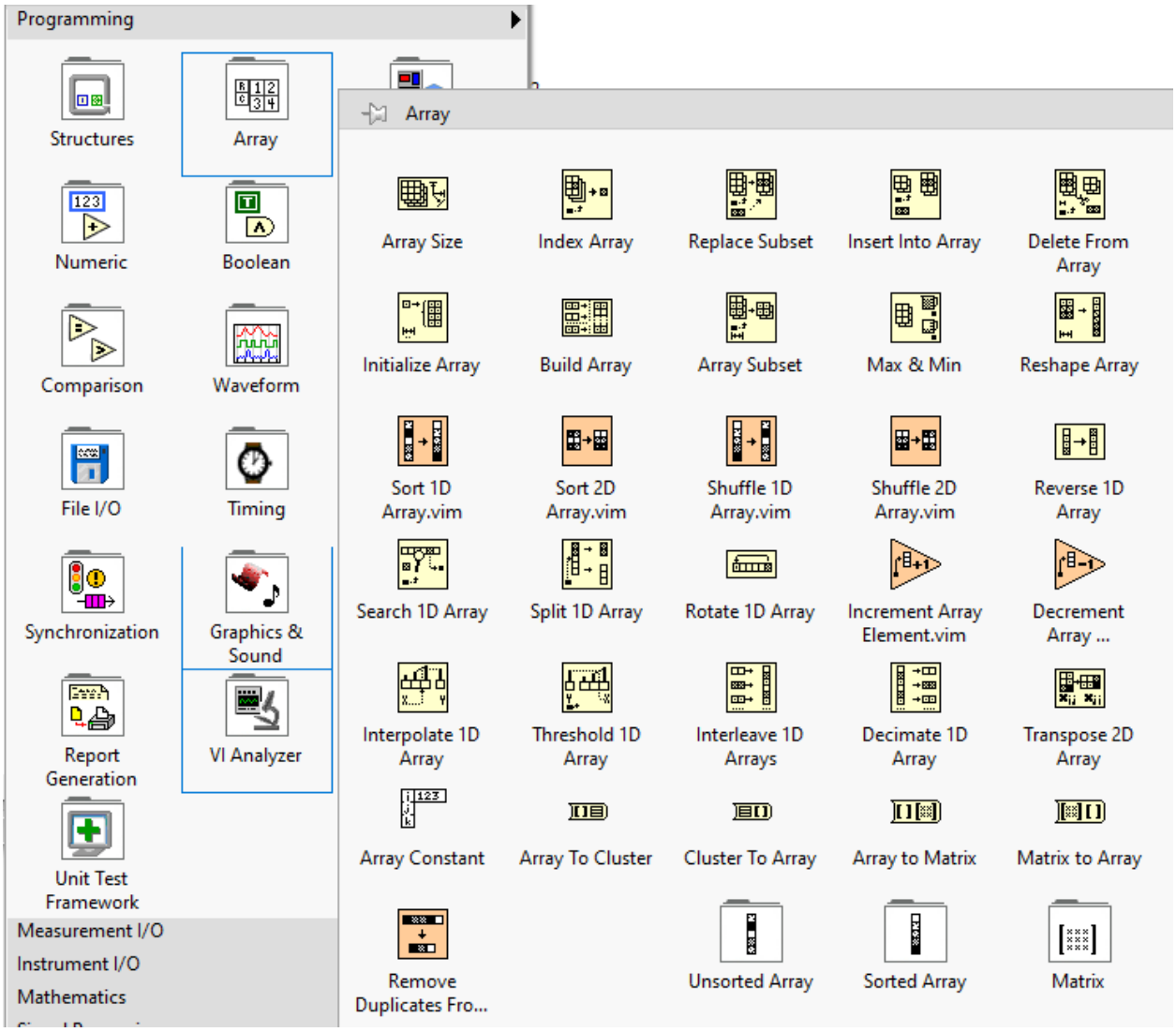


Рис. 4.5. Палітра інструментів для роботи з масивами

Для прикладу створимо ВП, який формує одновимірний масив з 10 випадкових елементів в діапазоні від 0 до 10, масштабує цей масив та виділяє підмасив вказаної довжини. Додатково відбувається запис підмасиву в текстовий файл. Лицьова панель та блок-діаграма представлені на рис. 4.6 та рис. 4.7 відповідно.

Примітка: Назви нових функціональних блоків приведені на блок діаграмі. Пропонується скористатись пошуком для їх знаходження і контекстною допомогою для більш детального опису їх входів та виходів.

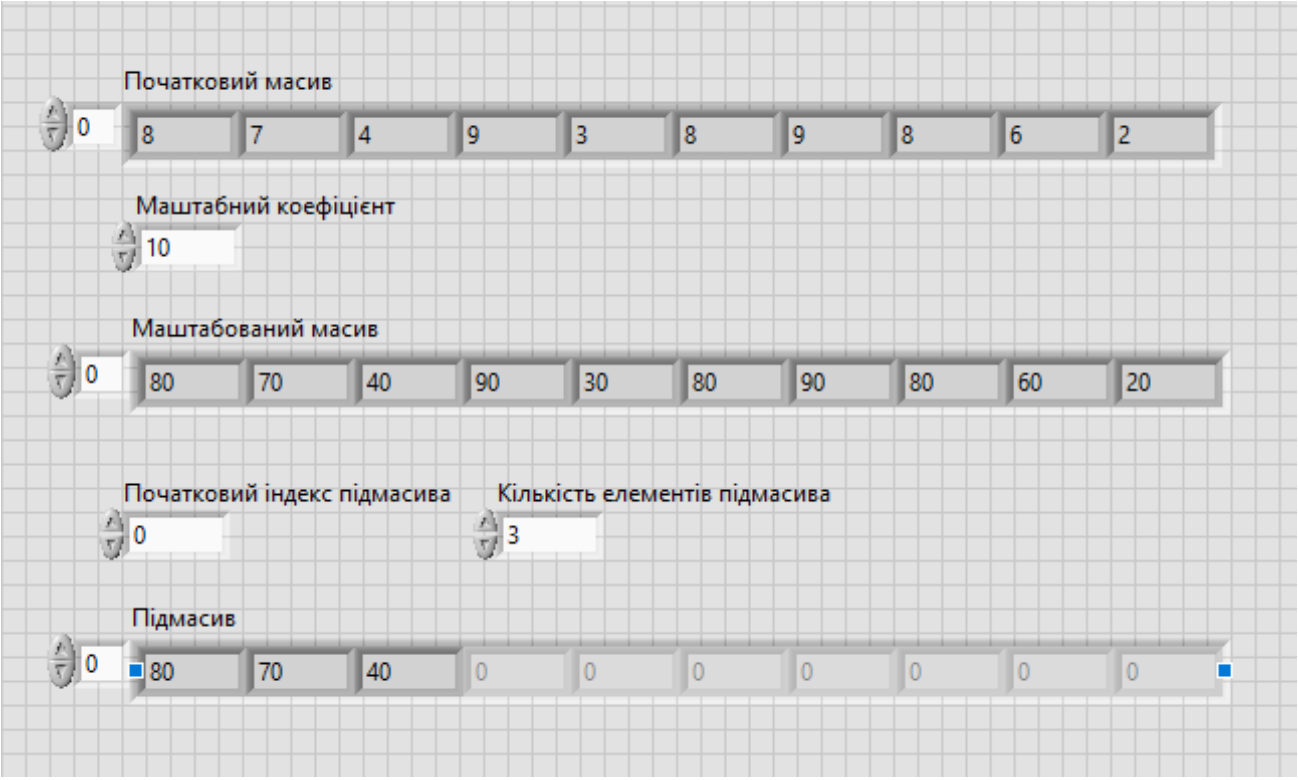


Рис. 4.6. Лицьова панель прикладу із використанням масивів

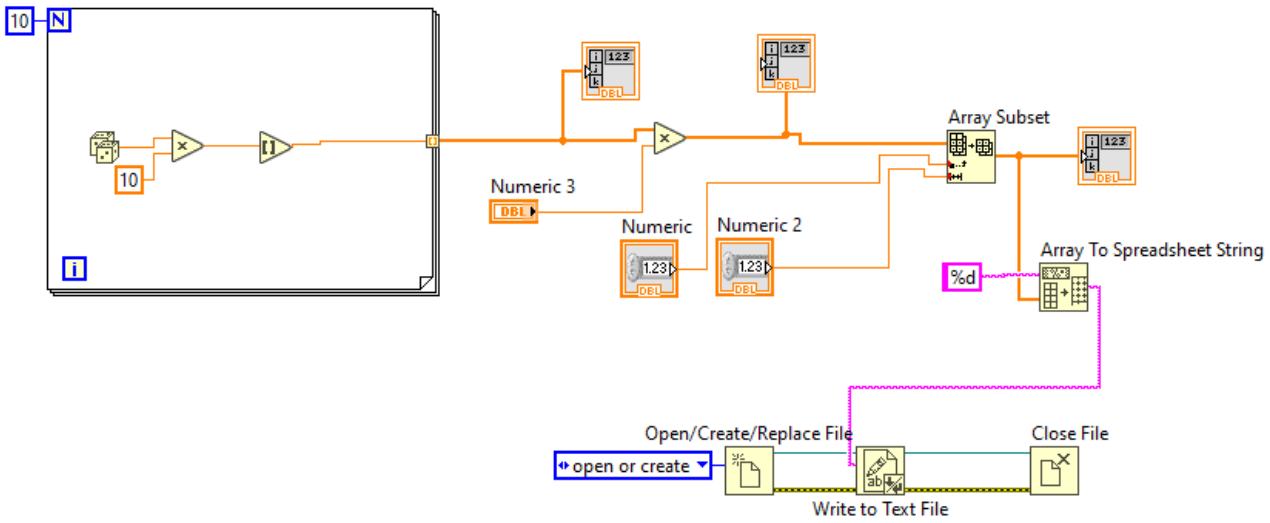


Рис. 4.7. Блок-діаграма прикладу з використанням масивів

4.1.2. Кластери

Кластери об’єкти та одночасно типи даних які об'єднують елементи різних типів. Найближчою аналогією кластерів є структури в класичних мовах програмування.

Об'єднання декількох груп даних в кластер усуває безлад на блок-діаграмі і зменшує кількість полів введення/виведення даних, необхідних підпрограмі ВП. Для старих версій LabVIEW максимально можлива кількість полів введення/виведення даних ВП дорівнювала 28. Якщо лицьова панель містить більше 28 елементів, які необхідно використати у ВП, можна деякі з них об'єднати в кластер і зв'язати кластер з полем введення/виведення даних. Як і масив, кластер може бути елементом керування або відображення даних, проте кластер не може одночасно містити елементи керування і відображення даних.

Примітка: Мається на увазі саме функціонування елемента кластера, а не його зовнішній вигляд. Відтак у індикаторному кластері об’єкт «Перемикач» буде працювати як індикатор, а не керуючий елемент.

Для створення кластерів з елементів керування і відображення даних слід вибрати шаблон кластера на палітрі *Classic/Modern – Data Containers – Cluster* і помістити його на лицьову панель. Після цього шаблон кластера слід заповнити елементами. Змінити розмір кластера можна за допомогою курсора. На рис. 4.8 показаний загальний вигляд кластера із розміщеними об’єктами.

Кожен елемент кластера має свій порядковий номер, не пов'язаний з положенням елемента в шаблоні. Першому доданому в кластер елементу автоматично присвоюється номер 0, другому елементу - 1 і так далі. При видаленні елемента порядкові номери автоматично змінюються. Подивитися й змінити порядковий номер об'єкта, поміщеного в кластер, можна, клацнувши правою кнопкою миші по краю кластера і вибравши з контекстного меню пункт *Reorder Controls In Cluster*. Панель інструментів і кластер приймуть вигляд, показаний нижче на рис. 4.9.

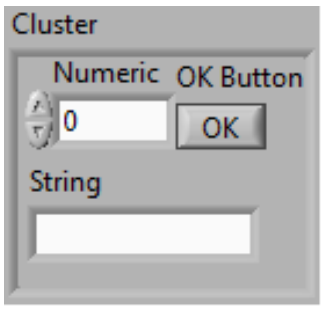


Рис. 4.8. Вигляд кластера на лицьовій панелі

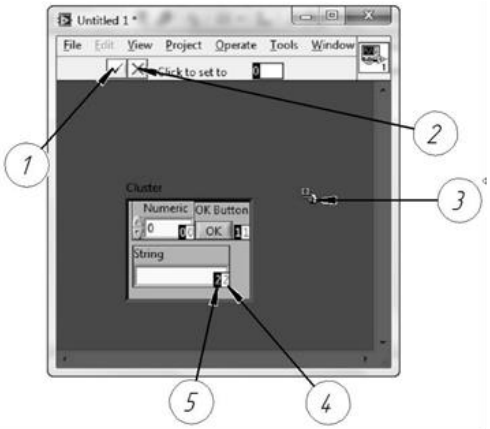


Рис. 4.9. Перевизначення порядку елементів в кластері

Де 1 - Кнопка підтвердження (Confirm button), 2 - Кнопка скасування (Cancel button), 3 - Курсор визначення порядку (Cluster order cursor), 4 - Поточний порядковий номер (Current order), 5- Новий порядковий номер (New order).

Для збирання окремих елементів у кластер використовується функція Bundle або Bundle by Name які знаходяться в меню *Programming – Cluster, Class & Variant*. Ці ж функції використовується для зміни даних в елементі вже існуючого кластера. При з'єднанні кластера з полем введення даних кількість полів введення даних функції повинна відповідати кількості елементів у вхідному кластері.

На полі введення даних *Bundle-cluster* можна подати тільки одну компоненту яка потребує заміну. Якщо відомий логічний порядок елементів, можна використовувати функцію Bundle для зміни значення елемента Command, з'єднавши елементи, як показано нижче на рис. 4.10.

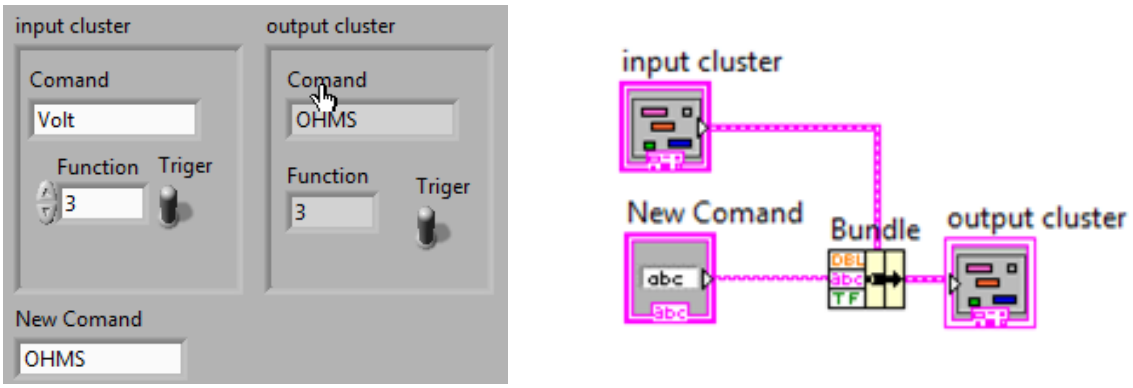


Рис. 4.10. Зміна значення одного елемента в кластері

Для заміни елемента у вже існуючому кластері використовується функція Bundle by Name. Функція Bundle by Name працює так само, як функція Bundle, але замість звернення до елементу кластера за його порядковому номеру звертається до нього за його власною міткою (іменем). При цьому можна отримати доступ лише до елементів, що мають власну мітку. Кількість полів введення даних не вимагає відповідності з кількістю елементів в кластері. На рис. 4.11 показано, як можна використовувати функцію Bundle by Name для зміни значень елементів Command і Function.

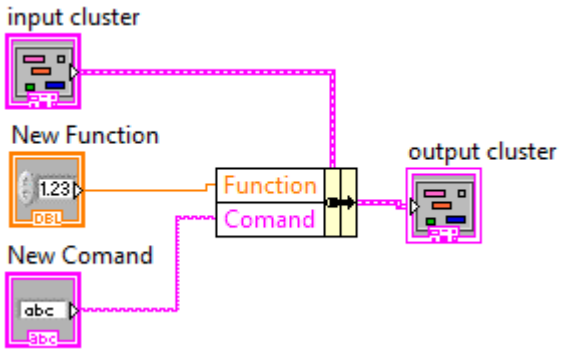


Рис. 4.11. Приклад використання блоку Bundle by Name

Для розбиття кластера на складові елементи використовуються функції Unbundle та Unbundle by Name блоки яких доступні у меню *Programming – Cluster, Class & Variant.*

Функція Unbundle використовується для розбиття кластерів на окремі елементи по їх індексу в кластері, що розбивається. Функція Unbundle by Name використовується для виділення з кластера елементів за певним іменем. Кількість полів виводу даних не залежить від кількості елементів в кластері. Приклад роботи із блоками функцій Unbundle та Unbundle by Name показаний на рис. 4.12.

Кластери констант. Оскільки кластер не лише об’єкт, а й тип даних то при необхідності можна створювати кластери констант, вибравши в палітрі меню *Programming – Cluster, Class & Variant* шаблон *Cluster Constant* і помістивши в нього числову константу або інший об'єкт даних, логічний або строковий.

Якщо на лицьовій панелі кластер вже існує, то кластер констант на блок-діаграмі, який містить ті ж елементи, можна створити просто перетягнувши кластер з лицьової панелі на блок-діаграму або клацнувши правою кнопкою миші на кластері, вибрати з контекстного меню пункт *Create -Constant.*

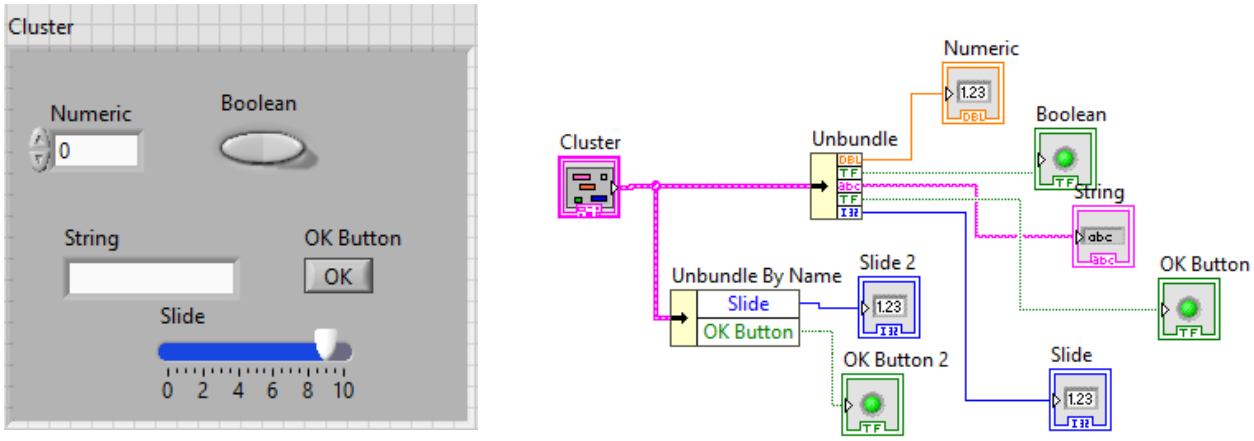


Рис. 4.12. Приклад використання блоків Unbundle та Unbundle by Name для розбиття кластера

Для прикладу, створимо ВП який буде містити три кластери як показано на рис. 4.13 та рис. 4.14. Перший кластер керуючого типу який містить чотири елементи різного типу.

Модифікований кластер – повторює елементи початкового кластера, проте змінює значення двох елементів. Третій кластер складається із двох елементів початкового кластера, інші два поля кластера виводяться окремо.

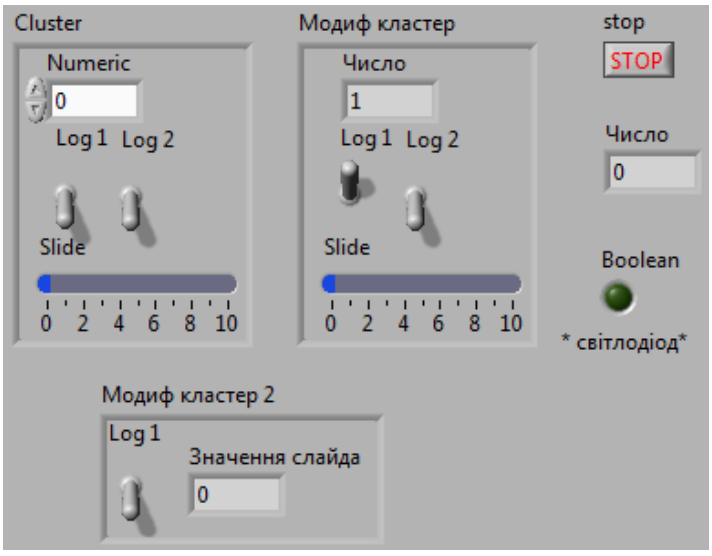


Рис. 4.13. Лицьова панель ВП по роботі із кластерами

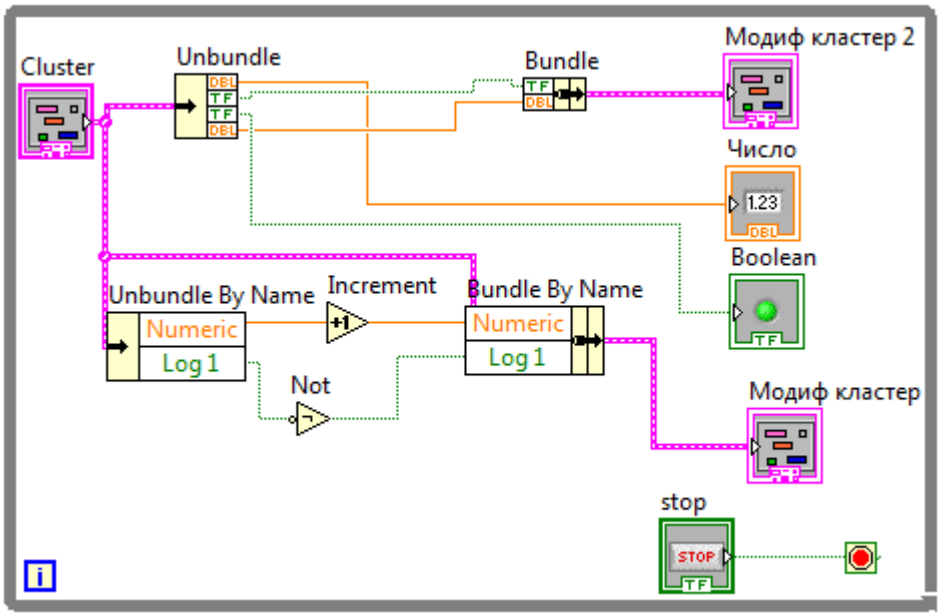


Рис. 4.14. Панель блок-діаграм ВП по роботі із кластерами

4.2. Завдання для виконання

Завдання 1. Створити ВП, який показаний на рис. 4.6.

Для парних варіантів: змінити розміри масиву, перетворивши його у тривимірний розмірністю 4x5x6.

Знайдіть мінімальні та максимальні значення.

Для непарних варіантів: на основі створеної програми змінити значення на бульові, а масив сформувати із 10 світлодіодів, що відображаються у випадкових станах.

При натисканні на кнопку зробити індикативне позначення сигналу «SOS», де 1 – це S, 0 – O.

Завдання 2. Створити ВП, який показаний на рис. 4.13.

Для парних варіантів: Модифікувати програму інтегрувавши у загальну структуру кластерів масив та кластер (хоча б із декількох елементів).

Для непарних варіантів: Модифікувати програму змінивши данні у індикаторних кластерах даними із створеного кластера констант.

4.3. Зміст звіту

1. Скріни лицьових панелей для завдань з п.п. 3.1.1. та п.п. 3.1.2., що демонструють працездатність віртуального приладу.
2. Скріни панелей блок-діаграм для завдань з п.п. 3.1.1. та п.п. 3.1.2.
3. Скріни лицьових панелей і панелей блок-діаграм з виконаними завданнями за індивідуальним варіантом.
4. Додаткові зображення, що підтверджують виконання завдання відповідно вашого варіанту (налаштування блоків і т.п.).
5. Висновки по роботі.

4.4. Контрольні запитання

1. Дайте визначення масиву, в чому відмінність з масивами у класичних мовах програмування?
2. Чи можна в комірці масиву розмістити інший масив?
3. Як відрізнити масив від звичайного елемента? По лицьовій панелі та по панелі блок-діаграм.
4. Як відрізнити індикаторні та керуючі масиви?
5. Як відрізнити масиви різної розмірності (одно- дво - багатомірні)?
6. Які функціональні блоки існують для роботи з масивами? Які ви використовували?
7. Що таке кластери? Наведіть аналогію із інших мов програмування?
8. Коли слід використовувати кластери?
9. Чи може кластер містити як керуючі так і індикаторні елементи?
10. Як можна змінити індексацію елементів в кластері?
11. Які ви знаєте блоки по роботі із кластерами?
12. У чому відмінність елементів Unbundle та Unbundle by Name?
13. У чому відмінність елементів Bundle та Bundle by Name?
14. Навіщо в блоках Bundle та Bundle by Name існує вхід Cluster та коли він застосовується?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 5

РОБОТА З РЯДКАМИ

Мета роботи: здобути навички роботи із текстовою інформацією, модифікацією рядків та пошуку в них необхідних елементів.

5.1. Теоретичні відомості

Строки або рядки – це послідовність відображуваних і не відображуваних ASCII символів. Рядки забезпечують незалежний від платформи формат обміну даними, до того ж, часто при обміні інформацією із різними чутливими елементами, інформація, навіть числова, представляється у вигляді рядка певної довжини.

Можна виокремити групу додатків використання, яких потребує вводу або виводу інформації представленої саме у вигляді рядка:

• Створення простих текстових повідомлень.

• Передача числових даних в прилади у вигляді рядків символів і перетворення рядків у числові дані.

• Зберігання числових даних на диск. Щоб зберігати числові дані у вигляді файлу ASCII, необхідно перед записом перетворити на рядки.

• Діалогові вікна інструкцій та підказок.

На лицьовій панелі рядки з'являються у вигляді таблиць, полів введення тексту і тегів. Для роботи з текстом та мітками використовуються рядкові керуючі та індикаторні елементи які розташовані в палітрі *Controls- Modern-String & Path*. Створення та редагування тексту в рядку проводиться за допомогою інструментів керування та введення тексту які автоматично змінюються при наведенні на відповідний елемент. Для зміни розміру строкового об'єкта на лицьовій панелі використовується інструмент переміщення. Для економії місця на лицьовій панелі можна використовувати смугу прокрутки. Для цього необхідно натиснути правою кнопкою миші по строковому об'єкту і вибрати в контекстному меню пункт *Visible Items - Scrollbar(Vertical/Horizontal).*

В залежності від поставленого завдання тип відображення текстової інформації в рядкових об'єктах можна змінювати. В табл. 5.1. показані типи

відображення рядка та приклади заповнення поля введення тексту. Перемикатись між режимами відображення можна в контекстному меню рядкового об’єкта.

Таблиця 5.1. Типи відображення інформації в рядкових об’єктах

Тип відображення	**Опис**	**Приклад тексту**
Режим стандартно відображення **(Normal Display)**	Відображає стандартні ASCII коди, використовуючи шрифт елемента керування. Керуючі коди для друку виводяться на екран у вигляді квадратів.	There are four display types. \ is a backslash
Режим відображення із зворотним слешем недрукованих керуючих кодів **('\' Codes Display)**	Виводить \ для всіх недрукованих керуючих кодів	There\sare\sfour\s display\stypes.\n\ !
Режим прихованого відображення тексту **(Password Display)**	Виводить * для всіх кодів текстового простору	********** ********** **********
режим відображення 16-тиричном ASCII кодів **(Hex Display)**	Виводить значення ASCII коду для кожного символу	5468 6572 6520 6172 6520 666F 7572 2064 6973 706C 6179 2074

Функції роботи з рядками. Для редагування і керування рядками на блок-діаграмі слід користуватися функціями обробки рядків, розташованими в палітрі *Functions -Programming- String*. Деякі з функцій роботи із рядками розглянуті нижче:

Функціональний блок *String Length*, що показаний на рис. 5.1 повертає значення кількості символів у рядку включаючи пробіли. Даний блок корисний при подальшій циклічній обробці текстової інформації, або при автоматизованому розбитті радків на підмасиви значень.

Функція *Concatenate Strings* (рис. 5.2) - об'єднує рядки і одномірні масиви рядків в окремий рядок. Для збільшення полів введення даних функції слід змінити її розмір потягнувши за нижню межу функціонального блоку .

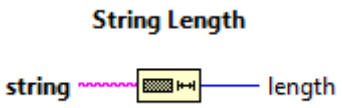


Рис. 5.1. Блок String Length

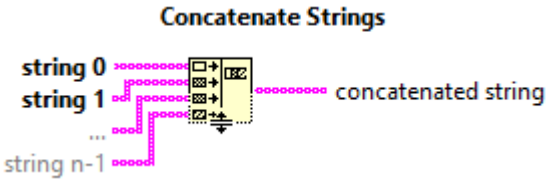


Рис. 5.2. Блок Concatenate Strings

Функція *String Subset* (рис. 5.3) - видає підрядок певної довжини length, починаючи зі значення зсуву (offset). Зсув першого елемента у рядку дорівнює 0.

Функція *Match Pattern* - шукає повторювану послідовність, подану на полі введення даних regular expression, у рядку починаючи зі значення зміщення offset, і якщо знаходить відповідність, розбиває рядок на три підрядка. Якщо відповідність не знайдено, полі виведення даних match substring є порожнім, а значення поля виводу даних offset past match (зміщення повторюваної послідовності в рядку) дорівнює -1. Загальний вигляд цього функціонального блоку показаний на рис. 5.4. Якщо співпадіння знайдене, то на виходах *before substring* і *after substring* – видається частина рядка перед і після співпадіння відповідно.

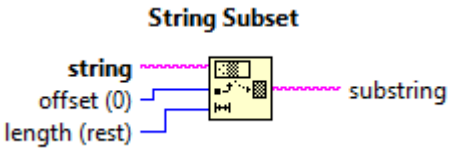


Рис. 5.3. Блок String Subset

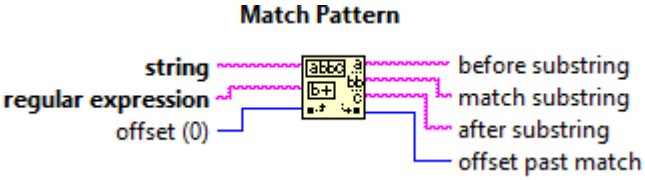


Рис. 5.4. Блок Match Pattern

Функція *Format Into String* - перетворює параметри будь-якого числового формату в рядок, а також поєднує їх з існуючим рядком (рядками).

Тобто, частково цей функціональний блок повторює функціонал блока *Concatenate Strings* проте з додатковим перетворенням чисельних значень в строку.

В залежності від формату, що поданий на відповідний вхід, зміниться формат при перетворенні та виведенні значень у якості підрядка. Частими у використанні є формати які складаються із спецсимволу «%», числа, що позначає кількість символів та тип даних з яким проводяться маніпуляції.

Наприклад %4f – поверне значення дійсного типу з точністю 4 знаки після коми. Оскільки формати значень є універсальними для середовища LabVIEW та можуть задаватись у доволі складному та параметричному вигляді, пропонується познайомитись із ними самостійно використовуючи допомогу середовища.

Примітка Для збільшення кількості вхідних текстових параметрів слід змінити розмір функції.

У наведеному нижче прикладі, що показаний на рис. 5.5, функція *Format Into String* повертає комбіновану строку, що складається з підстроки «Voltage is» та дійсного значення 1,28.

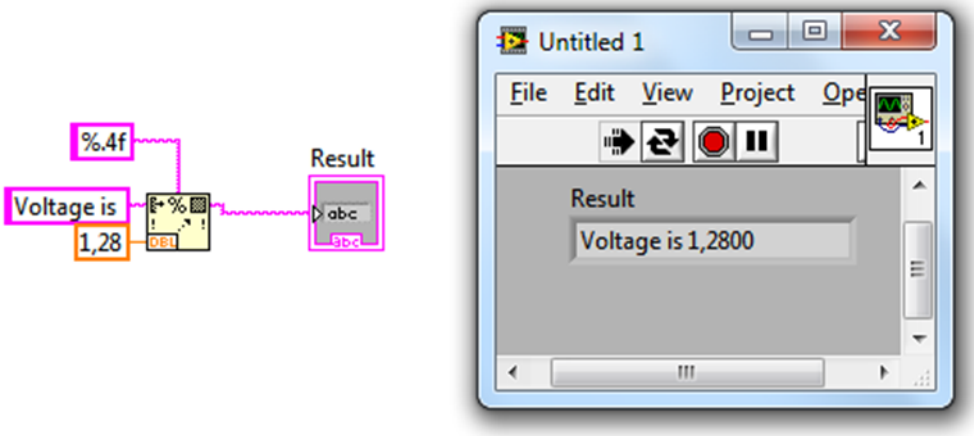


Рис. 5.5. Приклад функціонування блоку Format Into String

Треба зазначити, що для виконання тих самих функцій можна скористатись блоком з палітри Express *Build Text.*

Для перетворення рядка в числові дані слід використовувати функцію *Scan From String*. Експрес-ВП Build Text, розташований у палітрі Functions - *Express - Output*. Якщо вхідні величини мають не строковий тип даних, то вони перетворюються в рядок у відповідності з налаштуваннями цього експрес-ВП.

При приміщенні Експрес-ВП *Build Text* на блок-діаграму з'являється діалогове вікно налаштувань *Configure Build Text* (рис. 5.6). У наступному прикладі, що показаний на рис. 5.7, значення напруги подається на вхід експрес-ВП і перетвориться до формату даних з плаваючою комою з 4-ма числами після коми. Потім це значення додається до кінця рядка «Voltage is».

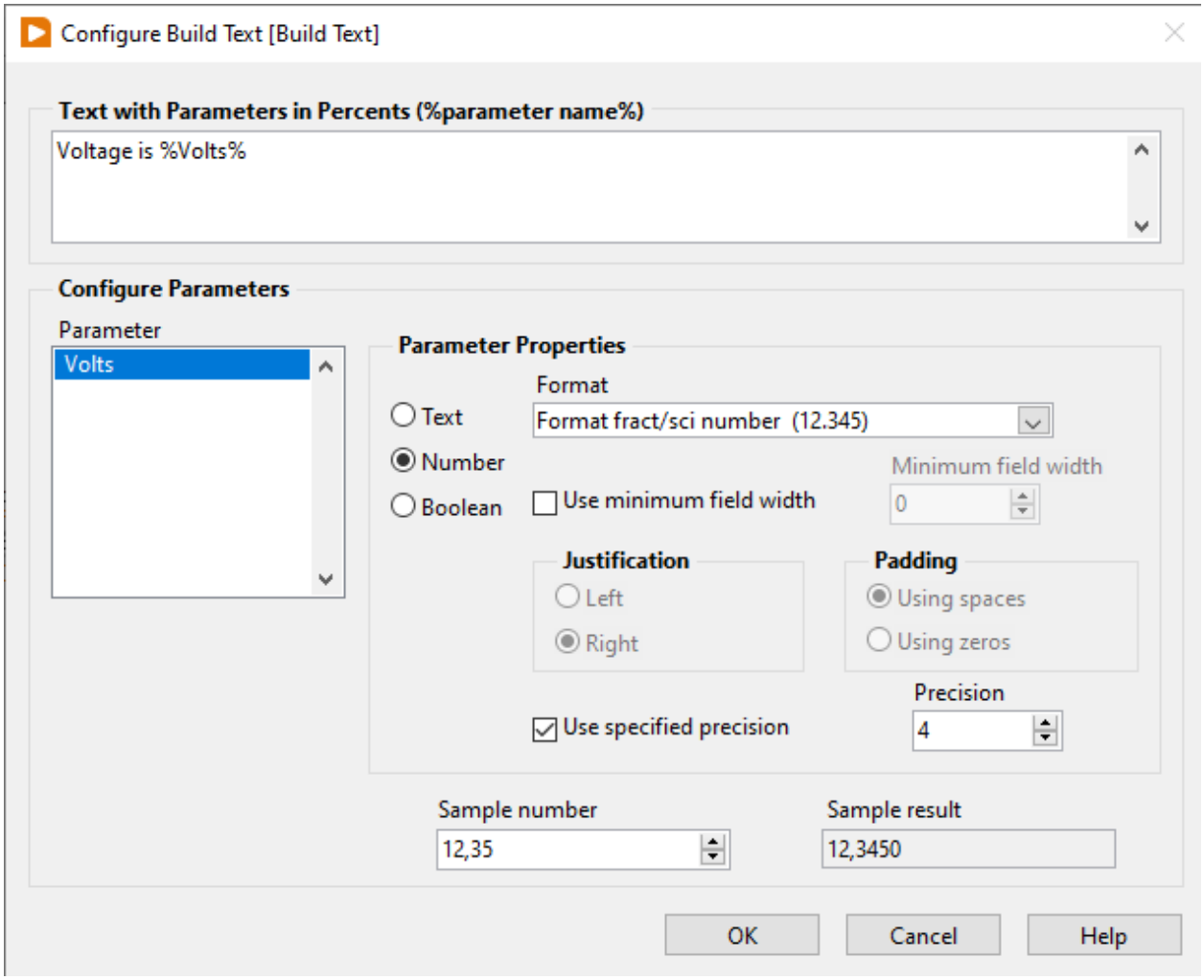


Рис. 5.6. Налаштування блоку Експрес-ВП Build Text

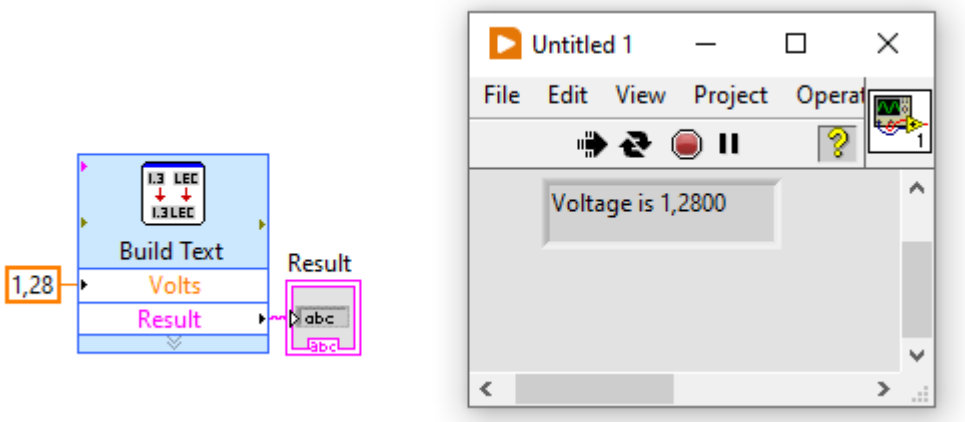


Рис. 5.7. Приклад функціонування Експрес-ВП Build Text

Функція *Scan From String* знайде і перетворить рядок, який містить допустимі числові символи, такі як «0-9, +, -, e, E» і десятковий роздільник, в дані числового формату. Функція починає перегляд рядка, який подається на поле введення даних Input String з номера символу, що задається на полі initial

Search Location. Функція може переглядати вхідний рядок різних типів даних, таких як числові або логічні дані, ґрунтуючись на форматі рядка. Наприклад, при значеннях на полях введення даних *Format String -% f, Initial Search Location - 8, Input String - VOLTS DC +1,28 E +2* функція видає результат 128, як показано нижче на рис. 5.8.

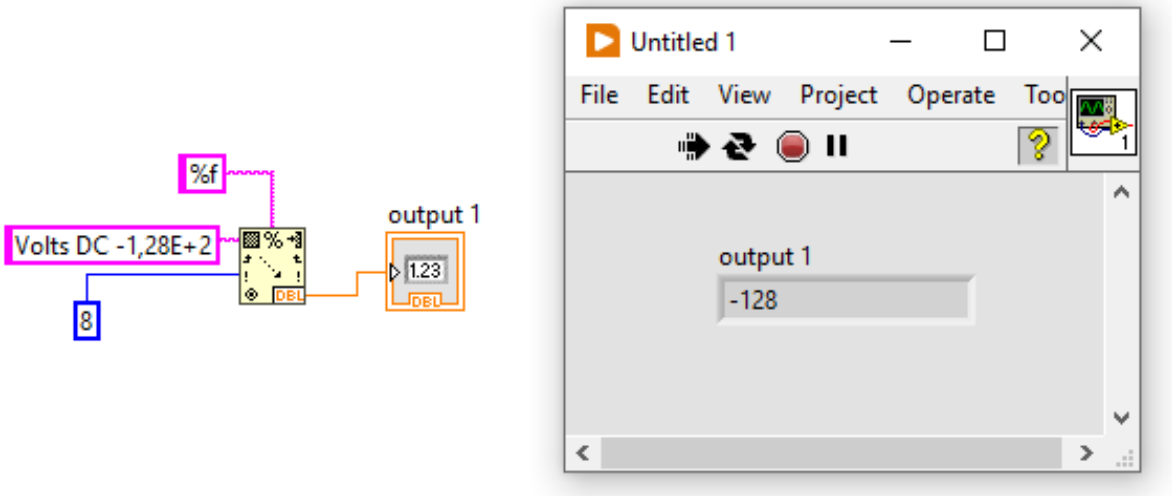


Рис. 5.8. Приклад функціонування блоку Scan From String

Для закріплення матеріалу складемо ВП, який буде використовувати описані блоки. Умовно ВП складається із двох незалежних частин. Перша частина формує результуючий рядок на базі введених строкових і чисельних даних. Друга частина – шукає співпадіння, виокремлює числове значення з рядка і перетворює його в числовий тип даних за вказаним форматом.

Для цього створимо новий віртуальний прилад та оформимо лицьову панель як показано на рис. 5.9.

Відтворювати коментарі і підписи до елементів не обов'язково.

Блок-діаграма представлена на рис. 5.10.

Примітка Значення, яке задається в текстовому полі «Строка 2» задається у форматі зворотного слешу Volts\sDc:\s\s+1,26E+1\r\n, оскільки часто рядок, що отримується від зовнішніх джерел інформації може містити недруковані символи.

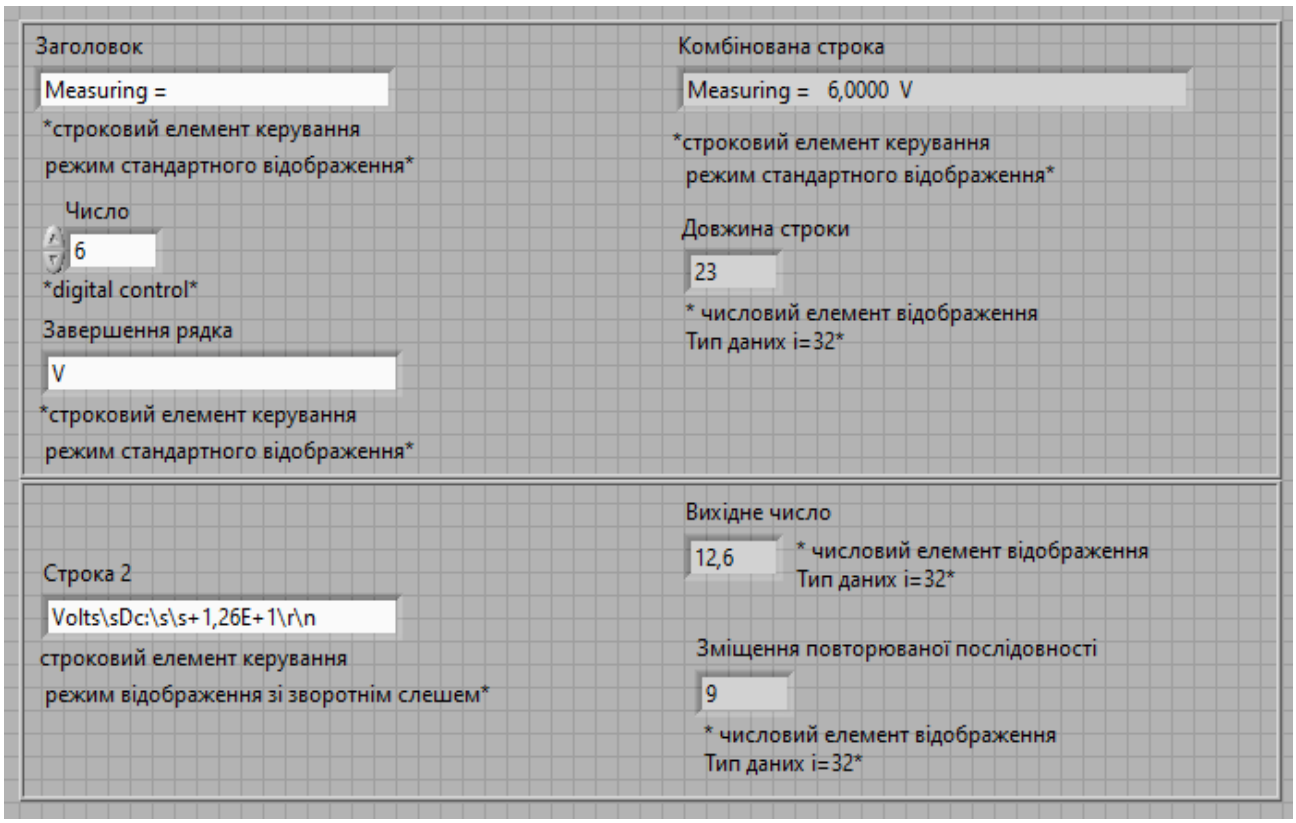


Рис. 5.9. Загальний вигляд лицьової панелі демонстраційного ВП по роботі із рядками

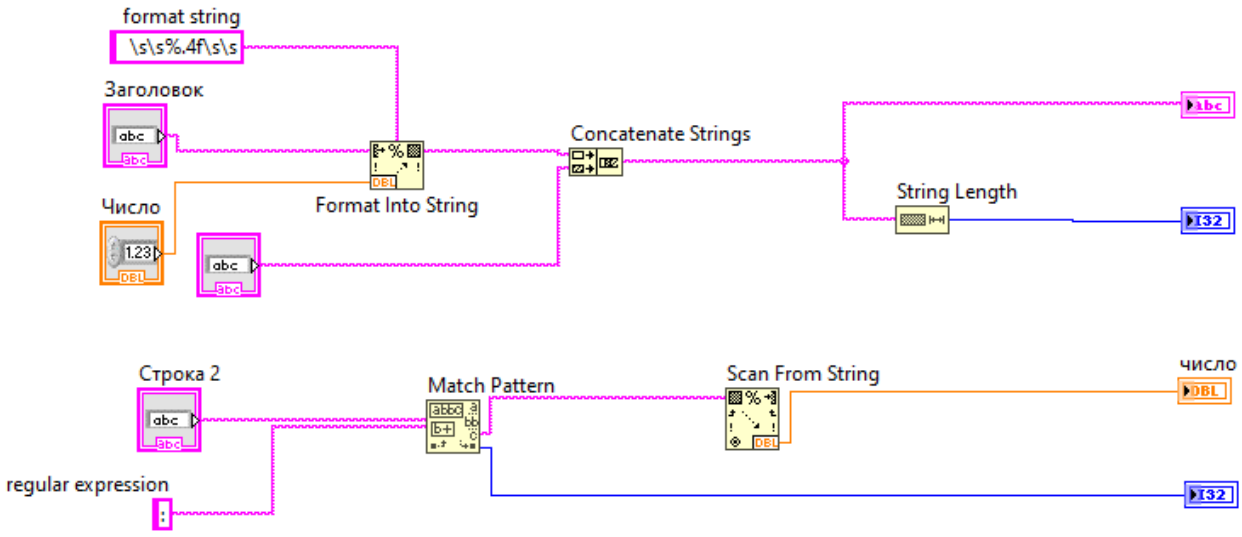


Рис. 5.10. Блок-діаграма демонстраційного ВП по роботі з рядками

5.2. Завдання для виконання

Створити ВП, який показаний на рис. 5.9. На основі створеного ВП виконати наступні завдання:

Для парних варіантів: Сформувати комбінований рядок який буде складатись з даних наступних типів: рядковий, цілочисельний, бульовий. Створений рядок вивести в об’єкт ВП та зберегти у файл.

Для непарних варіантів: З текстового файла зчитати рядок, який містить данні від умовного чутливого елемента представлені в форматі x=123 ; y = -15. Перетворити і вивести в окремі числові вікна значення при x та y, тобто наприклад 123 та -15.

5.3. Зміст звіту

1. Скріни лицьової панелі для завдання з п.п. 5.1.
2. Скріни панелей блок-діаграм для завдань з п.п. 5.1.
3. Скріни лицьових панелей і панелей блок-діаграм з виконаними завданнями за індивідуальним варіантом.
4. Додаткові зображення, що підтверджують виконання завдання відповідно вашого варіанту (налаштування блоків і т.п.).
5. Висновки по роботі.

5.4. Контрольні запитання

1. Чи є принципові відмінності рядкових типів даних в середовищі LabVIEW та класичних мовах програмування?
2. Які формати введення/виведення текстової інформації можна обрати?
3. Навіщо використовувати формат зі зворотнім слешем?
4. Чи є обмеження щодо типів даних які можуть бути виведені в текстові об’єкти? Якщо так, то в якому вигляді їх можна представити для подальшого виведення.
5. Які функціональні блоки існують для роботи з рядковими даними (назвіть і опишіть принаймні три блоки)?
6. Як виконати пошук необхідного числового значення в рядку?
7. Як знайти необхідний символ в рядку?

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 6
СТВОРЕННЯ ІНТЕРФЕЙСУ ДЛЯ РОБОТИ ІЗ ЧУТЛИВИМИ ЕЛЕМЕНТАМИ

Мета роботи: навчитись створювати універсальні інтерфейси для отримання даних від різних чутливих елементів.

6.1. Теоретичні відомості

6.1.1. Інтерфейс VISA

Взаємодія пристроїв з середовищем LabVIEW можлива багатьма шляхами, одним із універсальних є використання пакету VISA (Virtual Instrument Software Architecture). Це широко використовуваний стандартизований інтерфейс введення-виведення в області тестування і вимірювань для керування приладами з персонального комп'ютера.

VISA був розроблений компанією National Instruments (NI) в середині 1990-х для автоматизації вимірювань за допомогою віртуальних приладів В комплект з бібліотекою були включені також кілька програмних інструментів, наприклад, NI Spy - програма для протоколювання звернень до бібліотеки NI VISA та безпосередньо функції бібліотеки, на даний час основний функціонал не зазнав суттєвих змін, проте можливості протокольного обміну значно розширились.

Розглянемо типовий приклад використання розширення VISA, блок-діаграма якого представлена на рис. 6.1, а лицьова панель на рис.6.2.

Примітка Приклади можна знайти в теці прикладів в меню Help – *Find Examples…* і у відкрившомуся вікні NI Example Finder обрати необхідний приклад або скористатися пошуком в лівій частині вікна.

Дана програма демонструє використання функцій VISA для передачі значень через віртуальний COM-порт і його відображення на лицьовій панелі у вигляді текстового рядка.

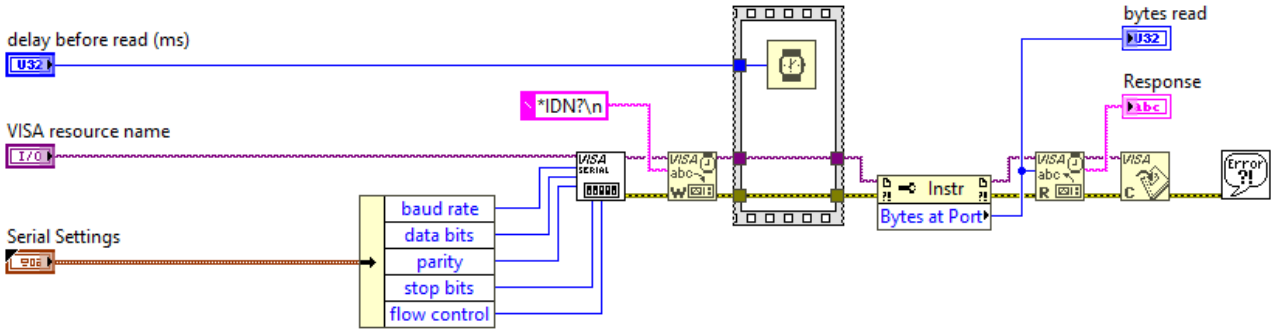


Рис.6.1. Блок-діаграма стандартного прикладу використання VISA

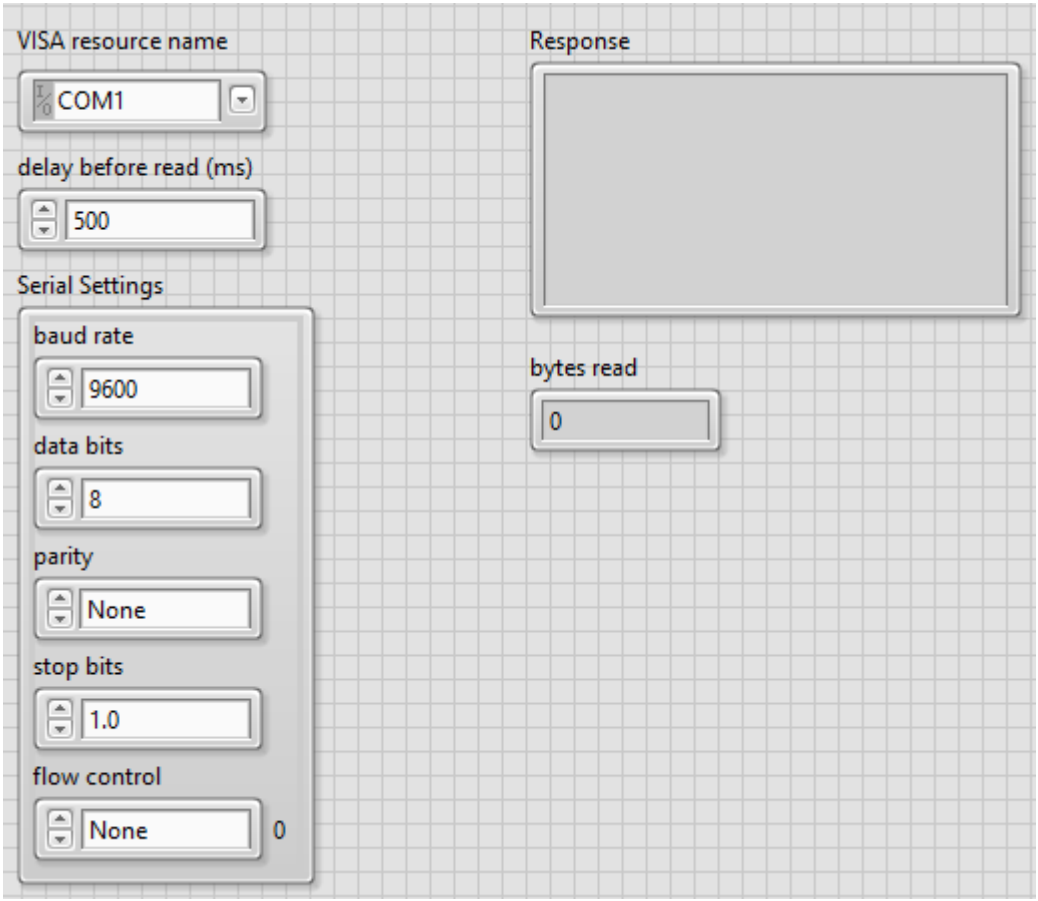


Рис.6.2. Блок-діаграма стандартного прикладу використання VISA

Примітка Наприклад для підключення платформи Arduino достатньо обрати відповідний Com-порт та встановити швидкість обміну (за замовчуванням 9600 бод). Якщо драйвер VISA не встановлений то в полі порта буде відображатись значення лише COM1, що може слугувати своєрідним індикатором коректності встановлення.

Драйвер встановлюється автоматично разом з встановленням середовища, проте якщо цього не відбулося, його можна завантажити з офіційної сторінки NI за посиланням: https://www.ni.com/en/support/downloads/

drivers/download.ni-visa.html.

Тепер більш детально розглянемо ключові керуючі блоки блок-діаграми, які розміщуються в палітрі інструментів Instruments I/O – VISA або Serial:

VISA Configure Serial Port (рис. 6.3) -ініціалізує послідовний порт, зазначений у назві ресурсу VISA із заданими параметрами.

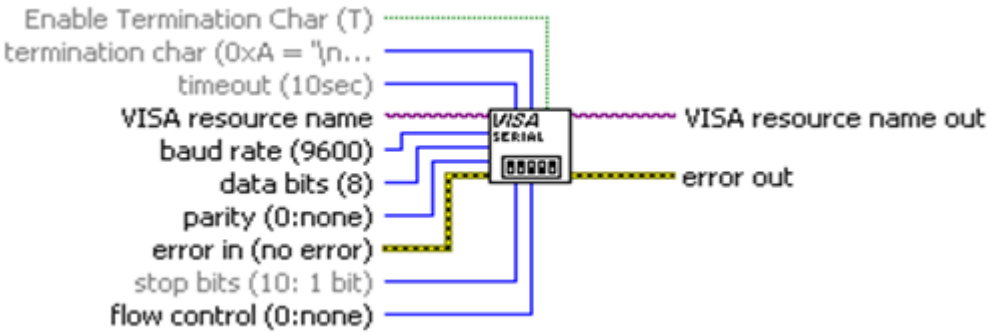


Рис.6.3. Блок VISA Configure Serial Port

Тепер розглянемо основні параметри вхідних значень та їх функціонал:

Enable Termination Char готує пристрій для розпізнавання завершального символу. Якщо встановлене значення TRUE (за замовчуванням) то перевірка наявності завершального символу буде проводитись, якщо FALSE- ні.

Termination char задається завершальний символ, при його надходженні, операція читання завершується. Значення за замовчуванням 0xA - це шістнадцятковий еквівалент символу переносу курсора на новий рядок (\n).

Timeout вказує час у мілісекундах для операцій запису та читання, значенням за замовчуванням - 10000 мс.

VISA resource name вказує номер порта, до якого буде проведене звернення, наприклад COM4.

Baud rate - швидкість передачі даних у бодах (біт/с). Значення за замовчуванням - 9600 біт/с.

Data bits - кількість бітів вхідних даних які будуть завантажені в буфер. Значення за замовчуванням - 8.

Parity вказує парність, яка використовується для кожного отриманого або переданого слова.

Stop bits вказує кількість стоп-бітів, що використовуються для позначення кінця бітового слова. Цей вхід приймає наступні значення: 10 - 1 стоп-біт, 15 – 1,5 стоп-біти, 20 – 2 стоп біти. За замовчуванням значення 10.

Flow control встановлює тип керування потоком, що використовується механізмом передачі. Цей вхід приймає наступні значення:

None (за замовчуванням) - механізм передачі не використовує керування потоком. Буфери з обох сторін з'єднання вважаються достатньо великими для утримання всіх переданих даних.

XON / XOFF - механізм передачі використовує символи XON і XOFF для виконання керування потоком. Механізм передачі керує вхідним потоком, посилаючи XOFF, коли буфер прийому майже заповнений, і керує вихідним потоком шляхом призупинення передачі.

VISA resource name out повертає ім’я ресурсу(COM-порту) яке було використано.

VISA Write Function (рис.6.4) - проводить запис даних у буфер для подальшої передачі на підключений пристрій. Входи, що дублюються за назвою з попереднього функціонального блоку, виконують ті ж функції і повторно описані не будуть.

Рис.6.4. Блок VISA Write Function

Write buffer містить дані, які потрібно записати на пристрій.

Error in описує умови помилки, які виникають до запуску цього вузла. На вхід подається стандартний кластер помилок.

Return count повертає кількість записаних байтів.

Error out містить інформацію про помилки. Повертає кластер помилок, що були сформовані цим або попередніми блоками.

VISA Read Function (рис.6.5) - зчитує вказане число байтів з буфера послідовного інтерфейса, визначеного ім'ям ресурсу VISA, і повертає дані в буфер читання.

Рис.6.5. Блок VISA Read Function

Byte count - кількість байт, яка буде зчитана із буфера.

Read buffer містить дані, прочитані з пристрою, інформація сформована у вигляді символьного масиву (рядка).

VISA Close Function (рис.6.6) – припиняє сеанс обміну інформацією із пристроєм, звільняє COM-порт зазначений у назві ресурсу VISA.

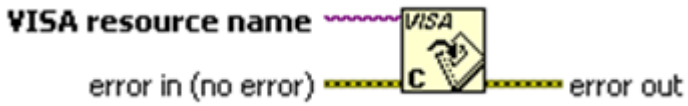


Рис.6.6. Блок VISA Close Function

Входи і виходи повторюють значення описані при розгляді попередніх блоків.

6.1.2. Ультразвуковий далекомір

У якості чутливих елементів розглянемо декілька приладів, один з них ультразвуковий далекомір HC-SR04 або його модифікації.

Загальний принцип функціонування ультразвукового далекоміра наступний: передавач випромінює декілька коротких ультразвукових імпульсів, які відбиваються від об'єкта і приймаються сенсором-приймачем. Відстань до об’єкта розраховується виходячи із часу від початку випромінювання до приходу відбитого сигналу з врахуванням швидкості звуку в середовищі.

В роботі розглядається модуль ультразвукового далекоміра на прикладі HC-SRO4, загальний вид якого наведений на рис. 6.7, а основні характеристики датчика представлені у табл. 6.1. Модуль живиться від джерела постійного струму напругою від 3.3 до 5В, для цього призначені лінії Vcc і Gnd. Інформаційними лініями є лінія Trig, яка призначенна для активації

послідовності вимірювання, та лінія Echo – призначена для передачі відповіді мікроконтролеру про відстань до об’єкта.

Рис. 6.7. Загальний вид модуля далекоміра SR-04

Якщо на сигнальну ніжку (Trig) подається імпульс тривалістю 10-15мкс, то ультразвуковий модуль буде випромінювати вісім пакетів ультразвукового сигналу із частотою 40кГц і чекати їхній відгук, що демонструє часова діаграма на рис. 6.8.

Таблиця 6.1. Основні характеристики датчика SR-04

Напруга живлення	3-6В
Споживаний струм	< 2 мА
Ефективний кут огляду	< 25 °
Діапазон вимірів	До 4м
Точність	0,3 см

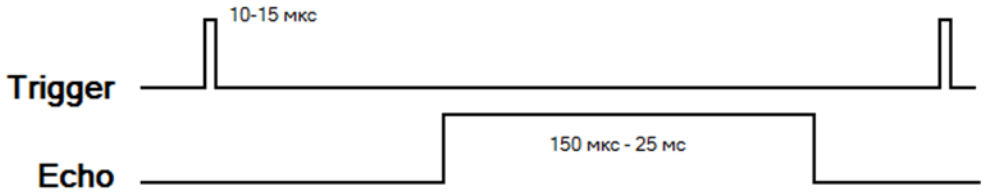


Рис. 6.8. Часова діаграма роботи

Після випромінювання сигнальних пакетів, на лінії Echo з’являється рівень логічної одиниці. При одержанні відповіді у вигляді еха, датчик знімає логічну одиницю з лінії Echo. Знаючи час випромінювання і прийому відповіді, одержуємо тривалість сигналу в «польоті». Маючи інформацію про швидкість звуку в середовищі, легко розрахувати відстань до об'єкта.

Схема підключення до відладочної плати Arduino UNO показана на рис. 6.9.

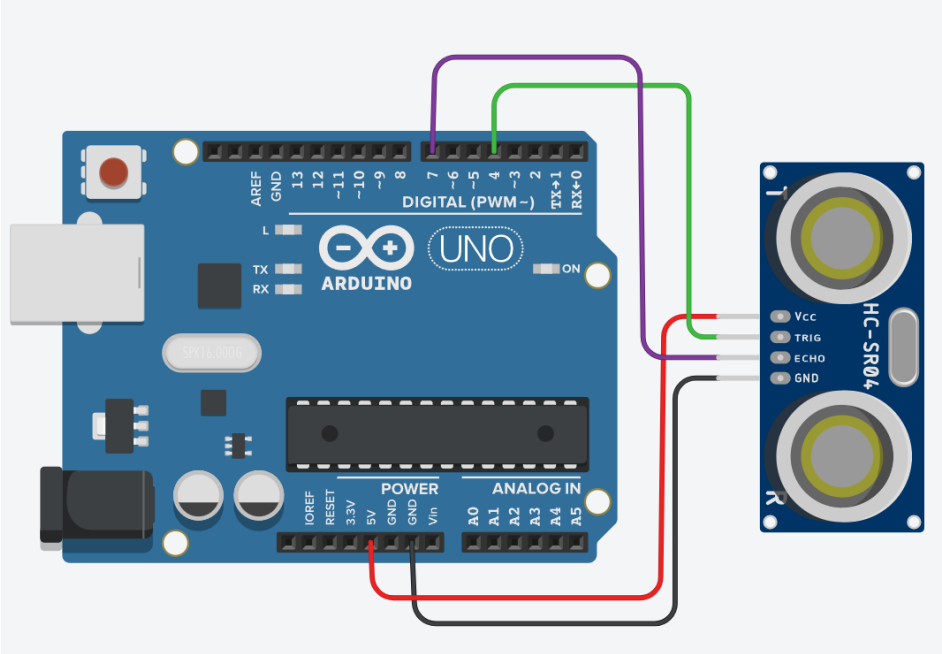


Рис. 6.9. Схема підключення модуля ультразвукового далекоміра

Для подальшої взаємодії із середовищем LabVIEW наведемо наступний код:

```
int TP = 4;//пін для лінії trig
int EP = 7;//пін для лінії echo

void setup()
{
 Serial.begin(9600);
  pinMode(TP, OUTPUT);
  pinMode(EP, INPUT);
}

void loop()
 {
 digitalWrite(TP, LOW);
  delayMicroseconds(2);
  digitalWrite(TP, HIGH);
  delayMicroseconds(10);
  digitalWrite(TP, LOW);
  long mKs = pulseIn(EP, HIGH); // час відгуку в мкс
```

```
  float cm=  mKs /29.412 / 2 ;  //швидкість звуку 340 м/с, або 29,412 мкс/см,
  //з врахуванням часу польоту в 2 сторони, ділимо навпіл
  Serial.print("Vidstan do objektu = ");
  Serial.print(cm);
  Serial.println("cm");
  delay(200);
}
```

Примітка Код використовується для середовища Arduino IDE будь-якої версії.

Наведений код в послідовний порт передає напис «*Vidstan do objektu = XXX cm*» де XXX – значення відстані до перешкоди у сантиметрах.

6.1.3. Мікромеханічний акселерометр ADXL-335

Наступним чутливим елементом який буде використаний як давач первинної інформації для побудови вимірювального каналу з інтерфейсом в середовищі LabVIEW буде акселерометр ADXL335 фірми Analog Devices, загальний вигляд плати з мікросхемою акселерометра показаний на рис. 6.10. ADXL335 – це мікроелектромеханічний (МЕМС), малоспоживаючий, повнофункціональний трьохосьовий акселерометр з аналоговим вихідним сигналом. Мінімальний діапазон вимірюваних прискорень становить ±3 g зі смугою пропускання до 1.6 кГц. Акселерометр чутливий до статичного прискорення, що викликане силою тяжіння. Отже, може бути використаний для вимірювання кутів нахилу, а також динамічного прискорення, викликаного рухом, ударами чи вібрацією.

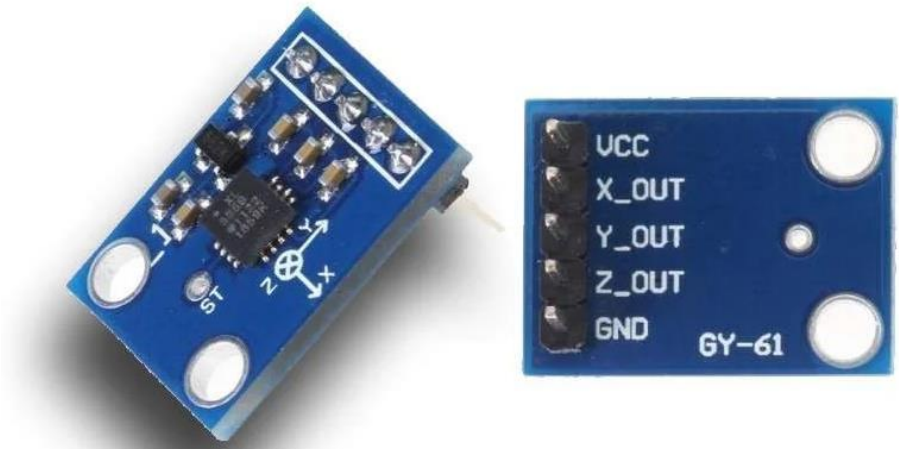


Рис. 6.10. Загальний вигляд плати акселерометра ADXL-335

Основні характеристики акселерометра ADXL335 наведені в табл. 6.2., а схема підключення до відладочної плати Arduino UNO показана на рис. 6.11.

Таблиця 6.2. Основні характеристики акселерометра ADXL335

Напруга живлення акселерометра (із стабілізатором)	1.8-3.6В (3-6В)
Споживаний струм	< 400 мкА
Діапазон вимірювань	-3.6g - +3.6g
Чутливість	300мВ/g
Ударні навантаження (граничні)	< 10000g

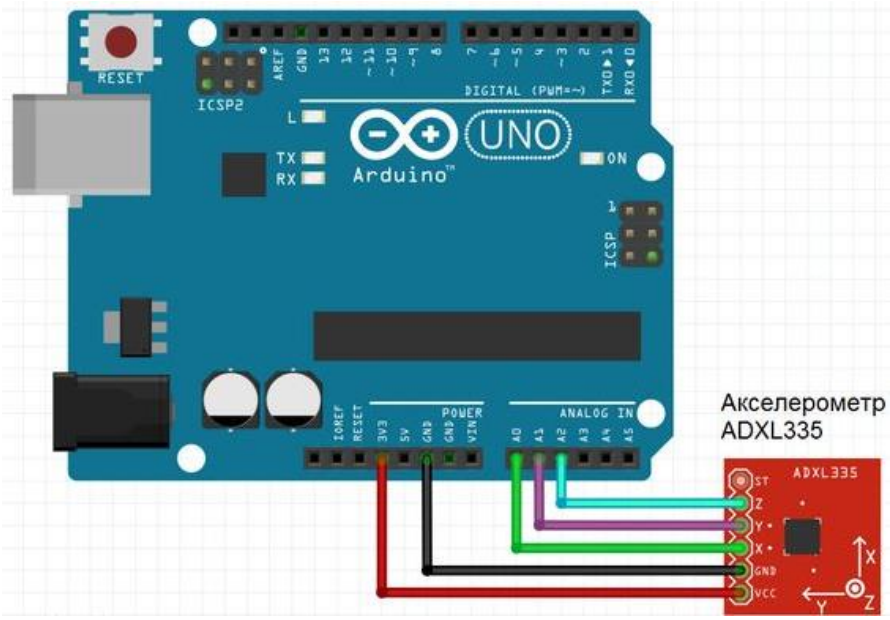


Рис. 6.11. Схема підключення акселерометра ADXL-335 до відладочної плати Arduino UNO

Далі наведемо код, що передає в Com-порт значення кута нахилу відносно осі X, який вираховується пропорційно із значення діючого прискорення g на вісь чутливості.

```
int xPin = 1;
int minVal = 266;
int maxVal = 407;

void setup()
{
  Serial.begin(9600);
}
```

```
void loop()
{
 int xRead = analogRead(xPin);
 int xGrad = map(xRead, minVal, maxVal, -90, 90);
 Serial.print("x= ");
 Serial.println(xGrad);
 delay(200);
}
```

Відповідно після виконання коду в послідовний інтерфейс буде передане значення «x = XX», де XX – кут нахилу платформи на якій умовно розташований акселерометр відносно горизонту в межах ±90°.

Примітка Для того, щоб отримати значення в вольтах або мілівольтах достатньо змінити рядок *int xGrad = map(xRead, minVal, maxVal, -90, 90);* на *int mV = map(xRead, 0, 1023, 0, 5000);*а знаючи маштабний коефіцієнт акселерометра 300мВ/g можна легко вирахувати діюче прискорення в g або м/с^2.

6.2. Завдання для виконання

Завдання 1. Використати ВП, який показаний на рис. 6.2 або створити його більш спрощений варіант який показано на рис. 6.12 та 6.13.

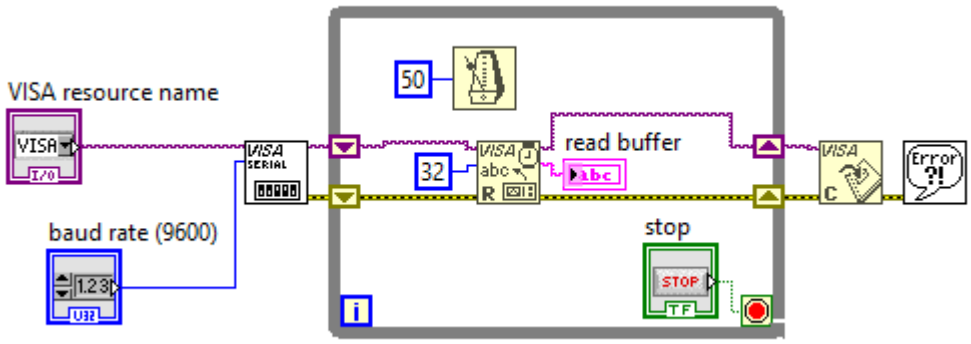


Рис. 6.12. Блок-діаграма ВП для отримання інформації від чутливих елементів

Після збірки ВП та підключення чутливих елементів (ЧЕ та відладочні плати надаються викладачем один на бригаду 3-5 студентів) перевірити коректне встановлення драйвера VISA та працездатність створених віртуальних приладів.

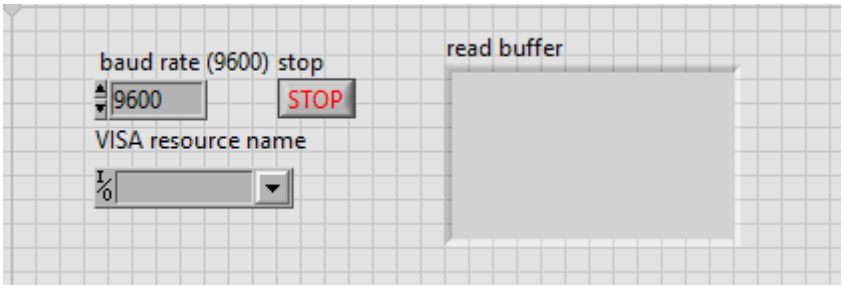


Рис. 6.12. Лицьова панель ВП для отримання інформації від чутливих елементів

Завдання 2. На основі створеного ВП виконати наступні завдання:

Для парних варіантів(або бригад якщо поділ відбувається по бригадам): Модифікувати ВП для роботи з трьома осями акселерометра ADXL-335 наступним чином: в канал зв'язку від МК приходить рядок наступного формату x=0.4; y=-1.2; z=0.8. Розбити отриманий рядок на 3 значення і вивести їх в окремі числові вікна та в таблицю із заголовками x,y,z.

Примітка Для розбиття рядка використовувати блоки з попереднього практикуму.

Для непарних варіантів (або бригад якщо поділ відбувається по бригадах): Модифікувати ВП для роботи з ультразвуковим далекоміром. Виокремити числове значення відстані з отриманого рядка (формат рядка залишається як в прикладі «*Vidstan do objektu = XXX cm*») та вивести ці значення у вигляді графіка. Результат зберегти в файл із розширенням *.xls. Якщо відстань до об'єкта менше ніж 20см на лицьовій панелі запалюється червоний світлодіод.

6.3. Зміст звіту

1. Скріни лицьових панелей і панелей блок-діаграм з виконаними завданнями за індивідуальним варіантом.
2. Додаткові зображення, які підтверджують виконання завдання відповідно вашого варіанту (налаштування блоків і т.п.).
3. Фото зібраних пристроїв за варіантами або бригадами.
4. Для парних варіантів – модифікований код для роботи з акселерометром.

5. Для непарних варіантів – прикріплений текстовий файл з результатами вимірювань відстані.
6. Висновки по роботі.

6.4. Контрольні запитання

1. Для чого призначений інтерфейс VISA?
2. Які основні блоки VISA використовуються для створення інтерфейсу для опитування чутливих елементів?
3. Чи будь-який чутливий елемент можна опитати з використанням VISA? Якщо так, то що потрібно зробити заздалегідь?
4. Які характеристики використовуваного модуля ультразвукового далекоміра?
5. Який принцип роботи далекоміра SR-04?
6. Назвіть основні недоліки та похибки ультразвукових далекомірів.
7. Для чого призначені мікромеханічні акселерометри?
8. Які основні характеристики акселерометра ADXL335 ви знаєте, які ви використовували безпосередньо при роботі з ним?
9. Чи зміниться щось принципово в створеному інтерфейсі для опитування чутливих елементів, якщо платформу Arduino замінити на іншу? Відповідь обґрунтуйте.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 7
КЕРУВАННЯ ВИКОНАВЧИМИ ПРИСТРОЯМИ КІБЕРФІЗИЧНИХ СИСТЕМ

Мета роботи: створити інтерфейс для керування виконавчими пристроями кіберфізичних систем на прикладі серводвигунів або крокових двигунів.

7.1. Теоретичні відомості

7.1.1. Серводвигуни

Основними виконавчими елементами сучасних кіберфізичних систем є актуатори різного принципу дії для формування лінійного або обертального руху певних елементів. У якості таких приводів часто використовуються серводвигуни або крокові двигуни з редуктором.

Серводвигун (сервопривід, рульова машинка) – це зазвичай двигун постійного струму з силовим редуктором. В даний час широко розповсюджені серводвигуни типів MG-90, MG-995/996r та ін. Загальний вигляд серводвигуна MG-996r показано на рис. 7.1.

Рис. 7.1. Загальний вигляд серводвигунів MG-996r

Характерною особливістю серводвигунів зазначеного модельного ряду є їхня сумісність керуючих сигналів, що дозволяє створивши код із сигналами керування маштабувати на різні проекти без особливих змін. Так, на рис. 7.2

та рис. 7.3 показані застосування серводвигунів MG-90, MG-995/996r в різних проектах.

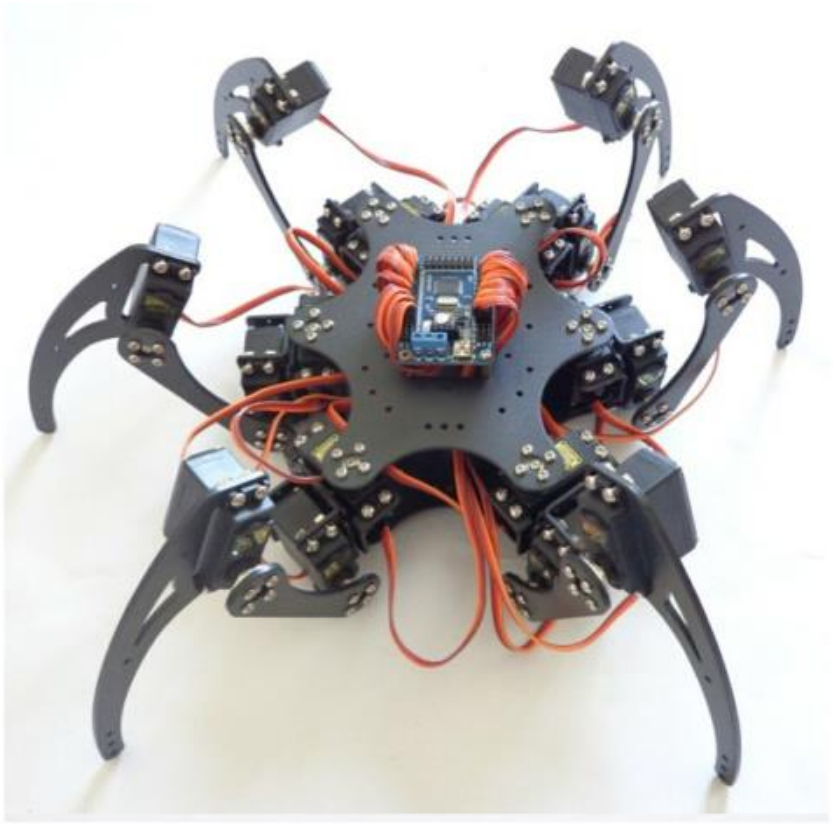


Рис. 7.2. Робот-гексапод на серводвигунах MG-995

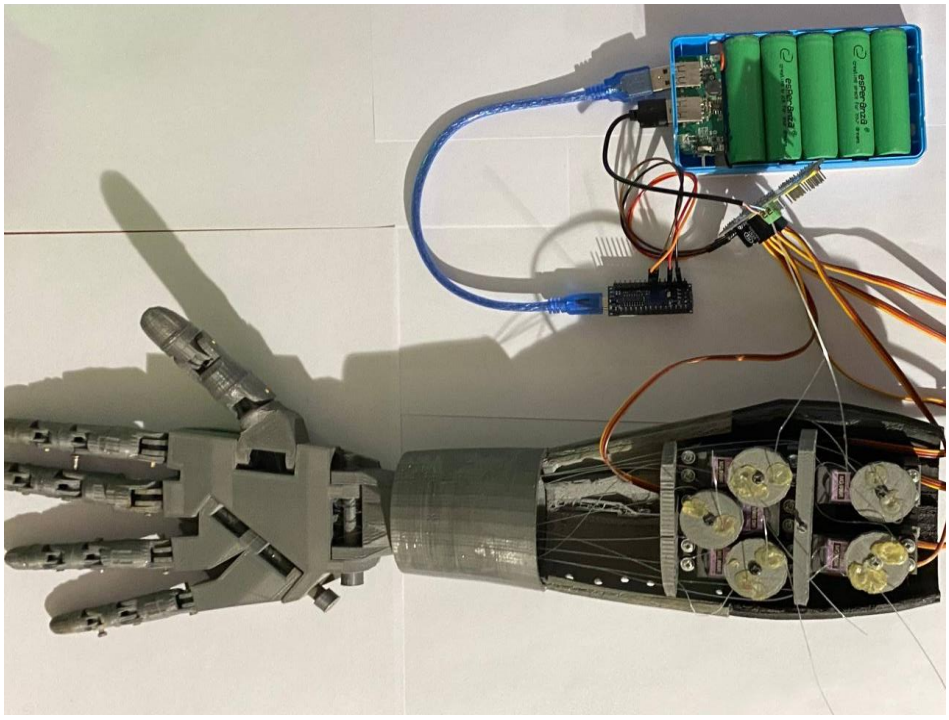


Рис. 7.3. Виконавча частина макету протезу руки на серводвигунах MG-996r

Основні характеристики найбільш розповсюджених серводвигунів наведемо в табл.7.1.

Таблиця 7.1. Характеристики серводвигунів

	MG-90	MG-995	MG-996r
Максимальний момент, кг*см	2,255	10	11
Швидкість повороту(макс), с /°	0.08с/60°	0.16с/60°	0.14с/60°
Кут повороту	<180° (360°)		
Тип керуючого сигналу	ШІМ		
Напруга живлення, В	4.8-6	4.8-7.2	
Габаритні розміри, мм	22,5x12x35,5	40,7x19,7x42,9	
Маса, г	13.4	55	
Максимальний струм, А	1,2	2,3	2,5

Схема підключення серводвигунів, що розглядаються наприклад до плати Arduino UNO показана на рис. 7.4 і однакова для будь-якої модифікації. Для підключення серводвигуна може використовуватись будь-яка цифрова лінія відладочної плати.

Примітка Оскільки при максимальних навантаженнях серводвигуни споживають значний струм (до 2,5А) то в таких випадках слід використовувати зовнішнє джерело живлення. Підключення до джерел живлення плати Arduino UNO дозволено лише в режимі холостого ходу (без навантаження) і для серводвигунів типу MG-90.

Для початку роботи із зазначеними серводвигунами в середовищі Arduino IDE передбачені приклади, що доступні в меню *File – Example – Servo* (Для версій до 2.0 середовища Arduino IDE, для останніх версій – приклади підвантажуються окремо).

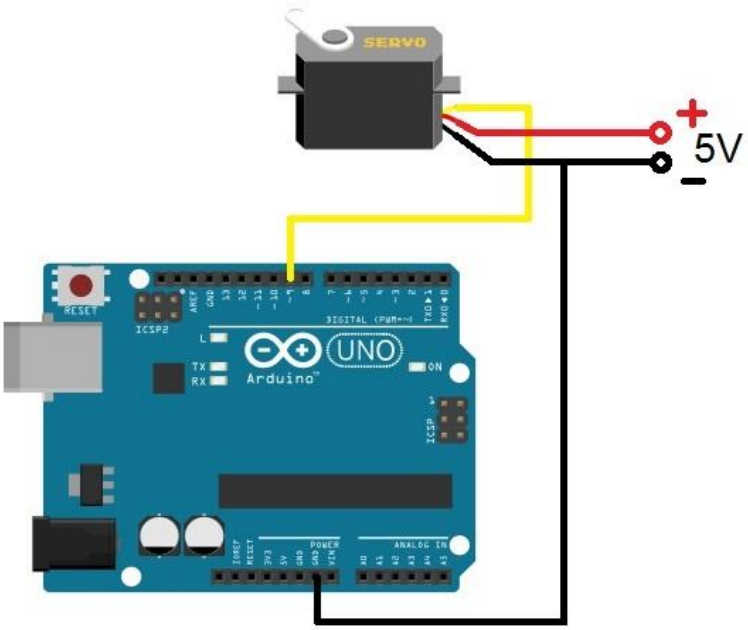


Рис. 7.4. Приклад підключення серводвигунів типу MG-90, MG-995/996r до плати Arduino UNO

Далі наведемо код, що тестово повертає вихідний вал серводвигуна з одного крайнього положення в інше:

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

Коментарі і код збережений в оригінальному вигляді як наведено в середовищі Arduino IDE.

Так, в рядку *myservo.attach(9);* необхідно вказати до якої лінії приєднаний серводвигун.

myservo.write(XXX); - вказує позицію кута повороту на який необхідно повернути вихідний вал серводвигуна, де XXX – значення кута повороту в градусах від 0 до 180.

Примітка Деякі наявні серводвигуни можуть суттєво відрізнятись за своїми характеристиками, тому вказаний в коді кут повороту може не відповідати реальному і потребує додаткової перевірки та калібровки.

Для розроблення інтерфейсу керування підготуємо технологічне ПЗ, яке буде опитувати послідовний порт і отримане значення буде сприймати як кут повороту вихідного валу серводвигуна. Наведений код треба завантажити в пам’ять мікроконтролера.

```
#include <Servo.h>
int read_input;
Servo myservo;
int val;

void setup() {
  myservo.attach(9);
  Serial.begin(9600);
  myservo.write(90);
}

void loop() {
   if (Serial.available() > 0) {
 read_input = Serial.parseInt();
 Serial.flush();
 read_input=constrain(read_input,1,180);
  myservo.write(read_input);
 delay(15);
   }
   Serial.println(read_input);
}
```

Перевірку коректності коду можна реалізувати відкривши монітор порта в середовищі Arduino IDE і послідовно вводити значення від 1 до 180. Якщо серводвигун відпрацьовує невірно (кут повороту визначається приблизно з врахуванням наявних технічних можливостей), то необхідно ввести додаткові поправочні коефіцієнти у наведений вище код.

7.1.2. Крокові двигуни

Мініатюрні крокові двигуни також користуються широким попитом при створенні кіберфізичних роботехнічних систем. Зараз використовується велика кількість різних крокових двигунів, проте принцип функціонування та керування розберемо на прикладі крокового двигуна 28BYJ-48 (24BYJ-48/30BYJ-48) з платою керування ULN2003, які показані на рис. 7.5, приклад використання у якого у якості керуючого приводу осі Z 3D-принтера показаний на рис. 7.6.

Рис. 7.5. Загальний вигляд крокового двигуна 28BYJ-48 з платою керування ULN2003

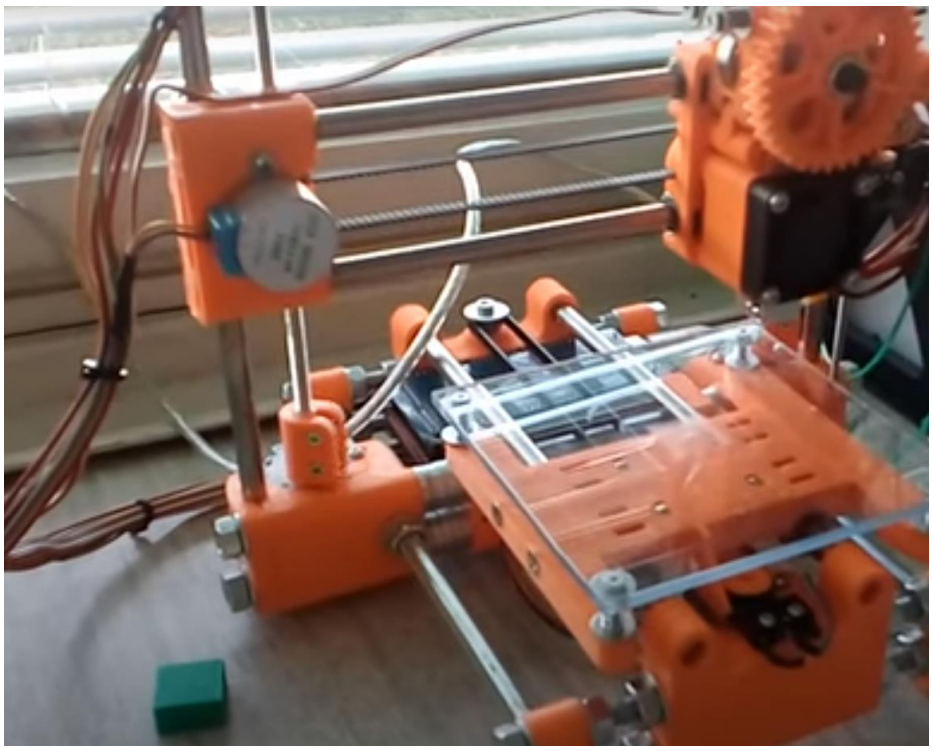


Рис. 7.6. Приклад використання двигуна 28BYJ-48 з платою керування ULN2003

При порівнянні з розглянутими серводвигунами, то кроковий двигун 28BYJ-48 має значно більший момент (~30 кгс*см), необмежений кут обертання вихідного валу, проте значно меншу швидкість і неможливість відслідковування положення вихідного валу без додавання додаткових чутливих елементів.

Принцип підключення крокового двигуна до плати Arduino UNO показаний на рис. 7.7. Для даного крокового двигуна можна використовувати живлення безпосередньо від відладочної плати (струм близько 40мА), хоча при значних навантаженнях рекомендується використовувати зовнішнє джерело живлення.

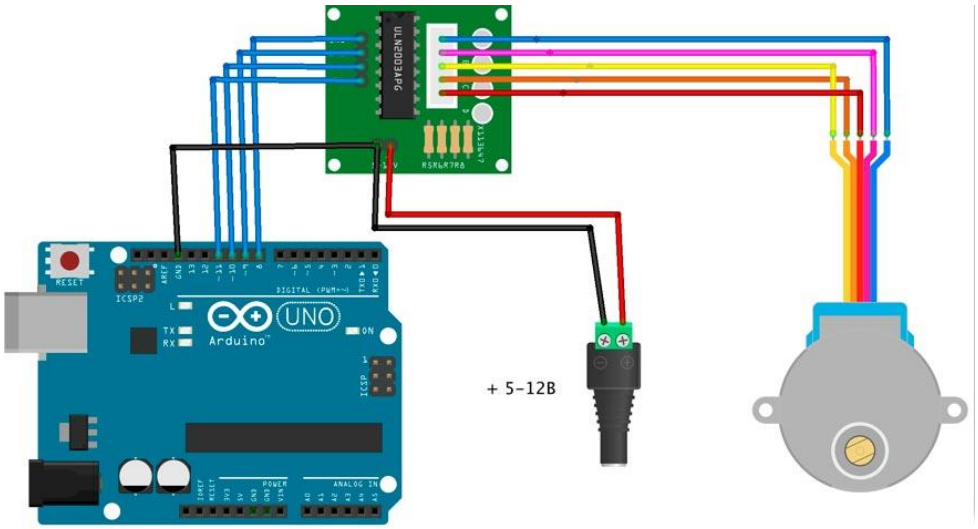


Рис. 7.7. Приклад підключення крокового двигуна 28BYJ-48 до плати Arduino UNO

Схема, що пояснює принцип роботи і послідовність подачі імпульсів на відповідні лінії наведена на рис. 7.8.

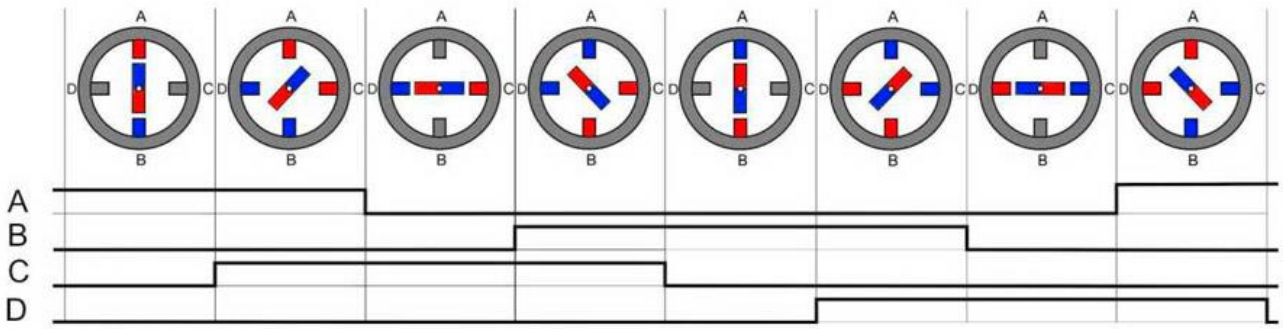


Рис. 7.8. Принцип керування кроковим двигуном 28BYJ-48

Далі наведемо код, який дозволяє обертати вихідний вал крокового двигуна за годинниковою стрілкою:

```
int motorPin1 = 11;
int motorPin2 = 10;
int motorPin3 = 9;
int motorPin4 = 8;

int motorSpeed = 5;     //змінна швидкості обертання
void setup() {
 pinMode(motorPin1, OUTPUT);
 pinMode(motorPin2, OUTPUT);
 pinMode(motorPin3, OUTPUT);
 pinMode(motorPin4, OUTPUT);
 Serial.begin(9600);

}
void loop(){
   clockwise();
}
void clockwise(){
 // 1
 digitalWrite(motorPin4, HIGH);
 digitalWrite(motorPin3, LOW);
 digitalWrite(motorPin2, LOW);
 digitalWrite(motorPin1, LOW);
 delay(motorSpeed);

**********
 // 8
 digitalWrite(motorPin4, HIGH);
 digitalWrite(motorPin3, LOW);
 digitalWrite(motorPin2, LOW);
 digitalWrite(motorPin1, HIGH);
 delay(motorSpeed);
}
```

Примітка Зверніть увагу, кроки 2-7 в наведеному коді умовно не показані, бо вони повністю відповідають діаграмі, що наведена на рис. 7.8, а сам приклад коду покликаний висвітлити лише загальний принцип керування.

7.2. Завдання для виконання

Завдання 1. В середовищі NI LabVIEW створити інтерфейс для керування двома серводвигунами типу MG-90, які використовуються як приводи керування кронштейном камери (див. рис. 7.9). Нижній серводвигун

дозволяє обертати платформу камери в азимутальній площині, а верхній – відповідає за кут нахилу в вертикальній.

Рис. 7.9. Сервопривідний кронштейн керування VGA-камерою

У створеному віртуальному приладі мають бути реалізовані два режими керування. 1й – ручний, для нього мають бути передбачені деякі елементи керування (наприклад Knob або Slide), що дозволяло б задавати кут повороту для кожного серводвигуна. І автоматичний режим, який формує певні постійні рухи платформи.

Для парних варіантів. В автоматичному режимі платформа має виконувати гармонічні коливання за двома осями, таким чином, щоб вісь візора умовної камери описувала коло.

Для непарних варіантів. Вісь камери знаходиться в умовному горизонті, а нижній серводвигун формує затухаючі коливання від максимального значення до повністю нерухомого стану.

Завдання 2. Робота із кроковим двигуном.

Для парних варіантів. Кроковий двигун робить вказану кількість обертів за годинниковою стрілкою. Кількість обертів задається в створеному ВП.

Для непарних варіантів. Кроковий двигун робить 1 оберт за годинниковою стрілкою, 2 – проти годинникової, 3 - за годинниковою, 4 - проти. Початок руху активує кнопка на лицьовій панелі ВП.

7.3. Зміст звіту

1. Скріни лицьових панелей і панелей блок-діаграм з виконаними завданнями за індивідуальним варіантом.
2. Додаткові зображення, що підтверджують виконання завдання відповідно вашому варіанту (налаштування блоків і т.п.)
3. Фото зібраних пристроїв за варіантами або бригадами.
4. Висновки по роботі.

7.4. Контрольні запитання

1. Що таке серводвигуни, який принцип їх роботи та де вони використовуються?
2. Назвіть основні характеристики розглянутих серводвигунів, в чому їх принципова відмінність?
3. Як повернути вихідний вал серводвигуна на заданий кут? Які максимальні і мінімальні значення може приймати цей параметр?
4. Який принцип роботи крокового двигуна?
5. Наведіть основні відмінності розглянутих серводвигунів і крокових двигунів?
6. Чи можна визначити кутове положення вихідного валу крокового двигуна?

ДЛЯ НОТАТОК

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Офіційна сторінка NI [Електронний ресурс] URL: https://www.ni.com/ru-ru/shop/labview.html
2. Кисельова О.Г., Соломін А.В. Технологія створення програмних продуктів. Програмування в NI LabVIEW : навч. посіб. Київ : НТУУ "КПІ", 2012. – 156 с.
3. Головня, В. М. Створення віртуальних приладів в середовищі LabVIEW [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра за спеціальністю 172 Електронні комунікації та радіотехніка освітніх програм: Інтелектуальні технології мікросистемної техніки, Інформаційна та комунікаційна радіоінженерія, Радіотехнічні комп’ютирезовані системи / В. М. Головня ; КПІ ім. Ігоря Сікорського.– Київ : КПІ ім. Ігоря Сікорського, 2023. – 142 с.
4. Офіційна сторінка Arduino [Електронний ресурс] URL: https://www.arduino.cc/