МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

В. М. Головня

СТВОРЕННЯ ВІРТУАЛЬНИХ ПРИЛАДІВ В СЕРЕДОВИЩІ LABVIEW

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник 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

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2023

Рецензент: Наливайчук М. В., к.т.н., доц., доцент кафедри програмного забезпечення комп'ютерних систем ФПМ

Відповідальний редактор: Мовчанюк А. В., к.т.н., доцент, зав.кафедрою ПРЕ

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 5 від 23.02.2023)
за поданням Вченої ради Радіотехнічного факультету
(протокол 02/2023 від 10.02.2023 р.)

Електронне мережне навчальне видання

Головня Вікторія Мілентіївна, старш. викл.

Створення віртуальних приладів в середовищі LabVIEW

Створення віртуальних приладів в середовищі *LabVIEW* [Електронний ресурс]: навч. посіб. для студ. спеціальності 172 «Електронні комунікації та радіотехніка» / В. М. Головня; РТФ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 9,92 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2023. – 142 с.

Методичні вказівки призначені для надання інформації щодо змісту та порядку виконання лабораторних робіт з дисципліни «Технології віртуальних приладів». Цикл лабораторних робіт з даної дисципліни спрямований на отримання практичних навичок роботи з середовищем графічної мови програмування *LabVIEW*, моделювання радіотехнічних пристроїв та створення віртуальних прототипів, інтегрування з різними пристроями та розробки нових.

Отримані під час виконання лабораторних робіт знання та навички дозволять в подальшому створювати, моделювати, візуалізовувати прилади різного рівня складності за допомогою віртуальних пристроїв.

Реєстр. № 22/23-475. Обсяг 6,5 авт. арк.

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

Віртуальні прилади допускають віддалене керування і спостереження через Інтернет. Вимірювальні системи на основі віртуальних приладів відрізняються своєю багатофункціональністю, гнучкістю і низькою вартістю як з точки зору устаткування, так і з точки зору витрат часу на розробку. За допомогою графічної мови програмування *LabVIEW*, можна програмувати сформульоване завдання у вигляді графічної блок-діаграми, з якої ЕОМ компілює машинний код. Завдяки цьому в *LabVIEW* на вирішення різних завдань витрачається значно менше часу і зусиль в порівнянні з написанням традиційного програмного коду.

Технологія *LabVIEW* знайшла використання в багатьох видах людської діяльності, наприклад, за її допомогою проводять виміри температури, електричних та магнітних величин, тиску, сил, просторового зсуву, механічної напруги і так далі. *LabVIEW* сприяє впровадженню комп'ютера у вимірювальні системи, полегшує проведення вимірів і дає можливість аналізувати виміряні величини, відображувати їх на графіках (базах даних) з можливістю публікації.

LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) є середовищем графічного програмування, яке широко використовується в промисловості, освіті і науково-дослідних лабораторіях як стандартний інструмент для збору даних і управління приладами.

LabVIEW − потужне і гнучке програмне середовище, вживане для проведення вимірів і аналізу отриманих даних. Це середовище, сумісне з операційними системами *Windows*, *MacOS*, *Linux*, *Solaris*.

Персональні комп'ютери є гнучкішими інструментами, ніж традиційні вимірювальні прилади, тому створення власної програми на *LabVIEW*, або віртуального приладу (ВП), є досить посильною справою. Призначений для користувача інтерфейс в середовищі *LabVIEW* інтуїтивно зрозумілий для людини з базовою технічною освітою.

ЛАБОРАТОРНА РОБОТА № 1
ЗНАЙОМСТВО З ІНТЕРФЕЙСОМ LABVIEW ТА СТВОРЕННЯ ТИПОВОГО ВІРТУАЛЬНОГО ПРИЛАДУ

Мета роботи: Здобуття практичних навиків і використання різних прийомів при розробці віртуальних приладів в середовищі *LabVIEW*.

Стислі теоретичні відомості

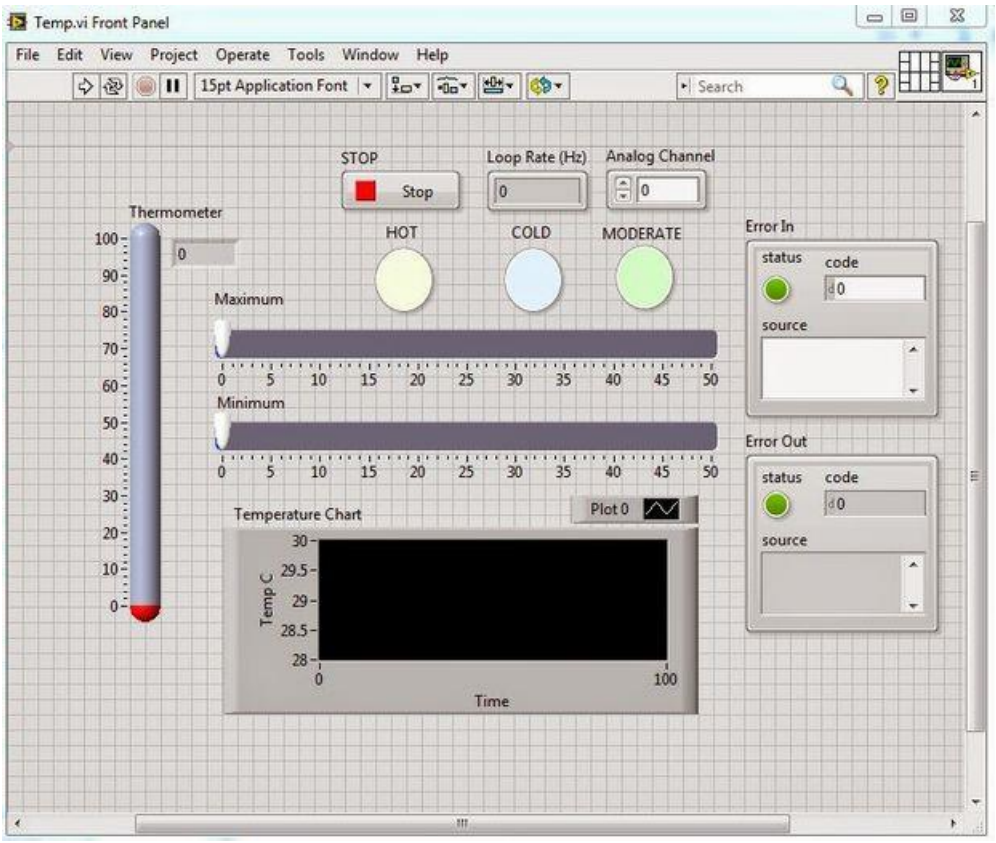
Віртуальні прилади (*VI – Virtual Instrument*). Традиційні вимірювальні прилади не дозволяють змінювати їх функціональні можливості, тому доводиться купувати всі прилади, які необхідні для вивчення якого-небудь об'єкту чи процесу. Технологія віртуальних приладів дозволяє перетворити звичайний персональний комп'ютер на пристрій з довільною функціональністю.

Комп'ютер з підключеними до нього багатофункціональними платами може бути потужною розрахунковою машиною, осцилографом, вольтметром, комутатором сигналів, частотоміром, і системою управління технологічним процесом і тому подібне.

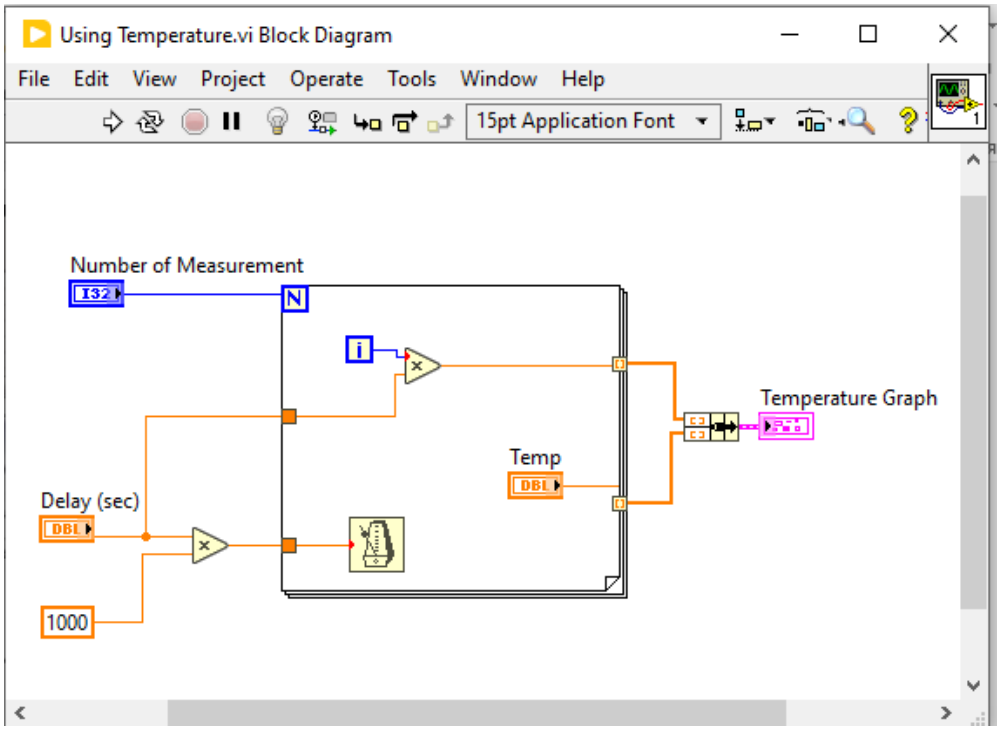
Склад бібліотек системи *LabVIEW* дозволяє в короткі терміни створювати необхідні інструменти для різних етапів досліджень, починаючи від елементарних приладів і закінчуючи керуючими інформаційно-пошуковими і аналітичними системами.

Будь-яка програма, створена в системі *LabVIEW,* називається віртуальним приладом (ВП) або віртуальним інструментом (*VI - Virtual Instrument*). Компонентами (складовими) ВП є передня панель, блок-діаграма (рис. 1.1) і піктограма/конектор.

Передня панель реалізує призначений для користувача інтерфейс з ВП, дозволяє задавати вихідні дані і відображати результати роботи ВП.



а)



б)

Рисунок 1.1 — Компоненти віртуального приладу: а) передня панель; б) блок-діаграма

Блок-діаграма є аналогом традиційної програми і реалізує функціональні можливості ВП.

Піктограма/конектор дозволяють використовувати ВП як підпрограми (*SubVI*, віртуальні "підприлади") при побудові модульних ієрархічних програм. Лише найпростіші застосування розробляються в *LabVIEW* як один єдиний ВП. Серйозні застосування є ієрархією ВП.

Таке ієрархічне застосування можна розробляти методом зверху "вниз", коли вихідне складне завдання, яке розбивається на декілька менших підзадач.

Чим детальніше продумана структура програми, чим краще описана специфікація вихідних даних і результатів роботи, тим швидше додаток буде створений, налагоджений і впроваджений.

Передня панель (*Front Panel*). Передня панель — це інтерактивний інтерфейс користувача. Саме з передньою панеллю працюватиме користувач програми, тому вона має бути зручною, інформативною і ергономічною. Передня панель може містити необхідні кнопки, тумблери, регулятори числових значень, графіки, лампи, впроваджені об'єкти. Більшість елементів передньої панелі можуть працювати в одному з двох режимів – регулятор (*Control*) або індикатор (*Indicator*).

Регулятори дозволяють користувачеві задати вихідні дані для віртуального приладу, а індикатори відображують результати роботи. При установці об'єкту на екран передньої панелі віртуальний прилад самостійно визначатиме режим його роботи. Так, наприклад, тумблер за умовчанням працюватиме в режимі "регулятор", а термометр – в режимі "індикатор".

За допомогою меню властивостей об'єкту, що викликається натисненням правої кнопки миші, розробник ВП може перемикати режим роботи, а також встановлювати інші властивості об'єкту.

Блок-діаграма (*Block Diagram*). Функціональні можливості віртуального приладу визначаються його блок-діаграмою, яка є графічною реалізацією алгоритму, блок-схеми.

Термінал (*Terminal*). Кожному елементу передньої панелі відповідає один термінал на блок-діаграмі. Термінали створюються системою

LabVIEW на блок-діаграмі автоматично, як тільки який-небудь елемент створюється програмістом на передній панелі. Залежно від налаштувань *LabVIEW* термінали відображуються або як піктограми, відповідні елементам передньої панелі, або як кольорові прямокутники різного вигляду.

Колір і зовнішній вигляд терміналу відповідає зіставленому типові даних, а назва (*Label*) терміналу – назві елементу передньої панелі. Контекстне меню (права кнопка миші) дозволяє швидко знайти елемент передньої панелі, відповідний вибраному терміналу.

Вузол (*Node*). Вузол – це аналог поняття "оператор" в текстовій мові програмування. Вузли – все те, що виконується під час роботи ВП:

• вбудовані функції *LabVIEW*,

• підпрограми (віртуальні "підприлади", *SubVI*).

Вузли бувають:

• прості (оператори $z=x+y$; $a=\cos(b)$);

• складні − конструкції програмування, наприклад, умови (оператори *if, switch, case of*), цикли (оператори *for, do-while*) і тому подібне.

Система *LabVIEW* має дуже різноманітну базу вузлів, яка дозволяє вирішувати безліч завдань. Вузол можна вибирати, користуючись розділами контекстного меню.

Можна також скористатися панеллю пошуку. Інколи це виявляється набагато зручнішим. Щоб скористатися цією можливістю потрібно натиснути кнопку ***Search*** в правому верхньому кутку діалогового вікна контекстного меню, після чого з'явиться перелік вузлів. За бажанням можна перелік представити в різних формах.

Провідник (*Wire*). Провідники – це різноколірні лінії на блок-діаграмі, що визначають передачу даних від джерела до приймача під час роботи віртуального приладу.

Колір і зовнішній вигляд провідника відповідає типу даних, переданих по провіднику. В будь-якого провідника єдине джерело даних і може бути

декілька приймачів. Провідник завжди повинен приєднуватися до необхідного контакту конектора вузла (терміналу), константи або до іншого провідника. У місці приєднання одного провідника до іншого відображується точка (якщо включений цей режим в меню налаштувань *LabVIEW*, меню *Tools>Options*, вкладка *Block Diagram*, пункт "S*how dots at wire junction*").

Провідник може мати необмежене число точок повороту, може бути будь-якої довжини (ефективність виконання програми від цього не залежить). Проте програміст повинен прагнути розташовувати термінали, вузли і провідники так, щоб блок-діаграма була наочною, простою і красивою.

Піктограма (Знак)/ конектор (з'єднувач) (*Icon/connector*). Піктограма (знак) – компактне графічне зображення вузла. Зазвичай при створенні блок-діаграми всі вузли зображуються у вигляді піктограм.

Конектор (з'єднувач) – певна конфігурація контактів, що дозволяють передати вузлу вихідні дані і отримати результати його роботи.

Конектор вузла можна відображувати за допомогою спливаючого меню властивостей вузла.

При підключенні провідників до контактів конектора *LabVIEW* виробляє перевірку типів даних, а також підказує програмістові, до якого саме контакту підключається провідник. Тому практично неможливо помилитися з підключенням провідників до вузла, програмісти рідко використовують перегляд конектора. Для всіх віртуальних приладів, які можуть використовуватися як підпрограми (*SubVI*), слід намалювати піктограму і розробити конектор.

Для завдання всіх вихідних даних і здобуття результатів роботи ВП використовуються регулятори та індикатори на передній панелі. Саме ці регулятори та індикатори можна поставити у відповідність потрібним контактам конектора.

Для розробки піктограми і конектора використовується спливаюче меню, доступне натисканням по правій кнопці миші на квадратному елементі піктограма/конектор в правому верхньому кутку вікна передньої панелі. З меню можна запустити графічний редактор для створення піктограми, а також проводити необхідні маніпуляції з конектором (вибирати шаблон конектора, повертати його і тому подібне). Контакти конектора ставляться у відповідність елементам передньої панелі за допомогою інструменту "котушка з провідниками".

Документування ВП. Графічна мова програмування "*G*", використовувана в *LabVIEW* є наочною, програма схожа на традиційну блок-схему алгоритму. Система *LabVIEW* дозволяє розробникові віртуального приладу робити будь-які текстові пояснення на передній панелі або блок-діаграмі, а також записувати основну довідкову інформацію про віртуальний прилад (меню *File>VI Properties*, вкладка *Documentation*).

Крім того, із спливаючого меню, можна вказати пояснюючу інформацію для будь-якого елементу передньої панелі. Всі ці пояснення і коментарі будуть сприйняті вбудованою довідковою системою *LabVIEW*. Вони відображуватимуться у вікні контекстно-чутливої довідки *LabVIEW* точно так, як і при використанні стандартних ВП, що входять в дистрибутив *LabVIEW*.

Також *LabVIEW* дозволяє автоматично згенерувати документацію на віртуальний прилад з його описом і всіх вхідних і вихідних параметрів. Хороші детальні коментарі дозволять полегшити та істотно скоротити терміни, потрібні для модернізації програмного забезпечення.

Порядок виконання ВП, технологія *Dataflow*. Визначає виконання віртуального приладу *LabVIEW* технологія *Dataflow*, у відповідність з якою порядок виконання програми визначає готовність потоків даних, що проходять від одного вузла до іншого. Загальні правила такі:

1. жоден вузол не може виконатися до тих пір, поки на всі контакти його конектора, до яких підключені провідники, не поступлять дані.

2. якщо дані поступають на декілька вузлів "одночасно", то і виконуються ці вузли "одночасно". Зрозуміло, в разі однопроцесорного комп'ютера, декілька дій дійсно одночасно виконуватися не можуть.

Ширше трактування другого правила таке: Якщо дані поступають на декілька вузлів "одночасно", то порядок виконання цих вузлів не визначений! В більшості випадків при обробці даних технологія *Dataflow* автоматично приводить до коректної послідовності виконання вузлів.

В разі неможливості або небажаності "провідникового" визначення порядку виконання віртуального приладу застосовується конструкція програмування послідовність (*Sequence*).

Типи даних в *LABVIEW*. В *LabVIEW* використовуються будь-які типи даних. Деякі типи даних відповідають звичайним текстовим системам програмування (ціле число, логічні дані, рядок та ін.). Інші типи даних реалізовані лише в і і призначені для надійнішої і зручнішої роботи у складі автоматизованої системи наукових досліджень.

LabVIEW працює з такими типами даних, як осцилограма (*Waveform*), сигнал (*Signal*), ресурс *VISA*, вимірювальний або управляючий канал і тому подібне. Кольором і зовнішнім виглядом терміналів і провідників *LabVIEW* підказує розробникові віртуального приладу, як і які дані обробляються блок-діаграмою, при підключенні провідників. Тому робота зі всім різноманіттям типів даних в *LabVIEW* вельми зручна.

Інтерфейс користувача. Інтерфейс користувача ВП (*VI*) подібний до інтерфейсу користувача реального приладу. Передня панель ВП (*VI*) може виглядати приблизно так, як на рис. 1.2:

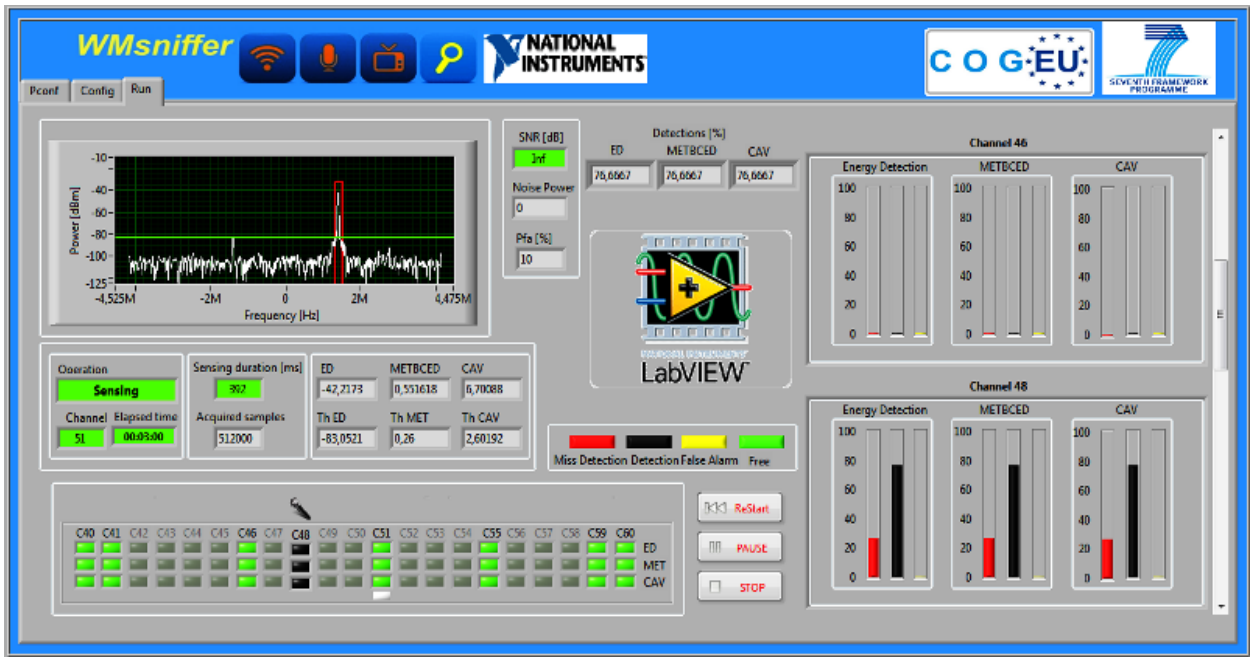


Рисунок 1.2 — Передня панель віртуального приладу

Інтерфейс користувача ВП (передня панель *VI*) − комбінація засобів управління та індикаторів. Засоби управління моделюють реальні пристрої введення даних і забезпечують їх надходження в блок-схему ВП. Індикатори, моделюють реальні пристрої виводу, які відображують дані, отримані на виході блок-схеми ВП.

Програміст додає засоби керування та індикатори на передню панель, вибираючи їх з так званої “спливаючої палітри засобів керування” (із списку наявних компонент), показаної на рис. 1.3:

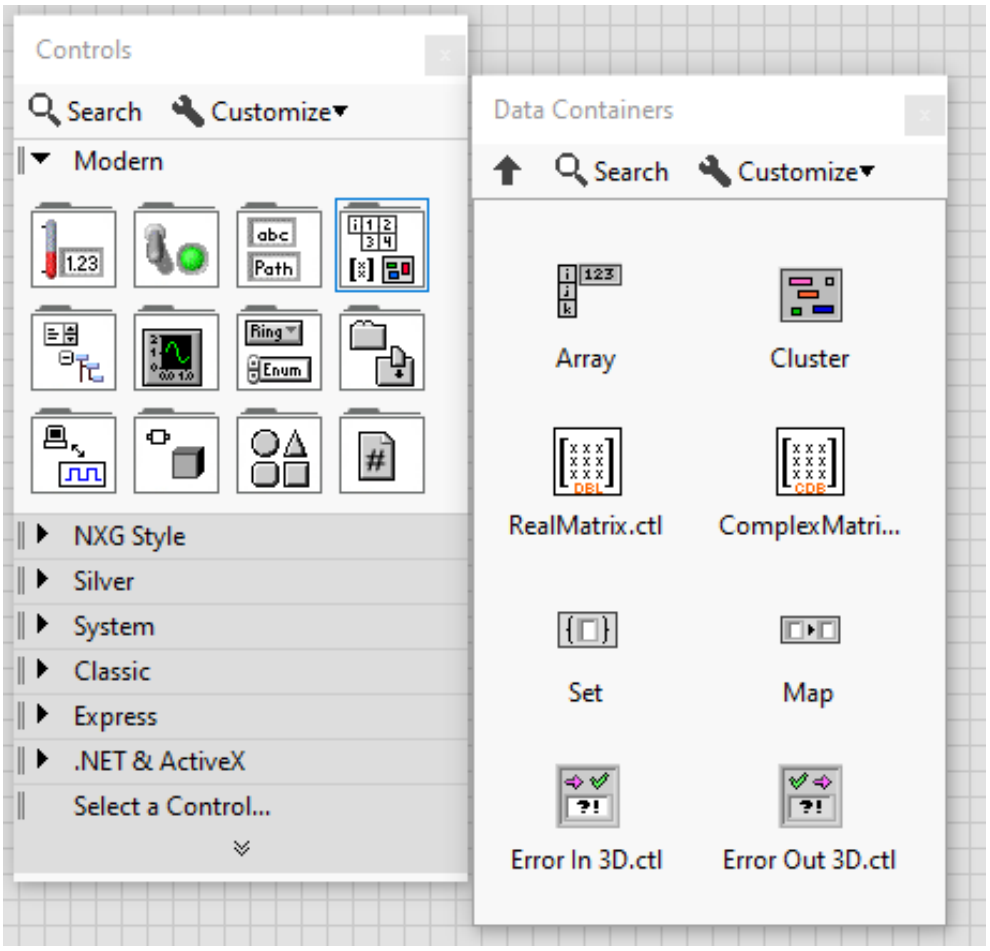


Рисунок 1.3 — Вибір індикаторів на передній панелі

Можна змінювати розмір, форму, і позицію перемикачів або індикаторів. Крім того, кожен перемикач або індикатор має спливаюче меню, за допомогою якого зазнають змін різні властивості або вибираються різні параметри редагованого об'єкту.

Стрілочний індикатор. Натисканням миші по лицьовій панелі викликаємо палітру *Controls*, на палітрі *Controls>Numeric* вибираємо натисканням миші стрілочний індикатор *Meter* і встановлюємо його на лицьовій панелі (рис. 1.4).

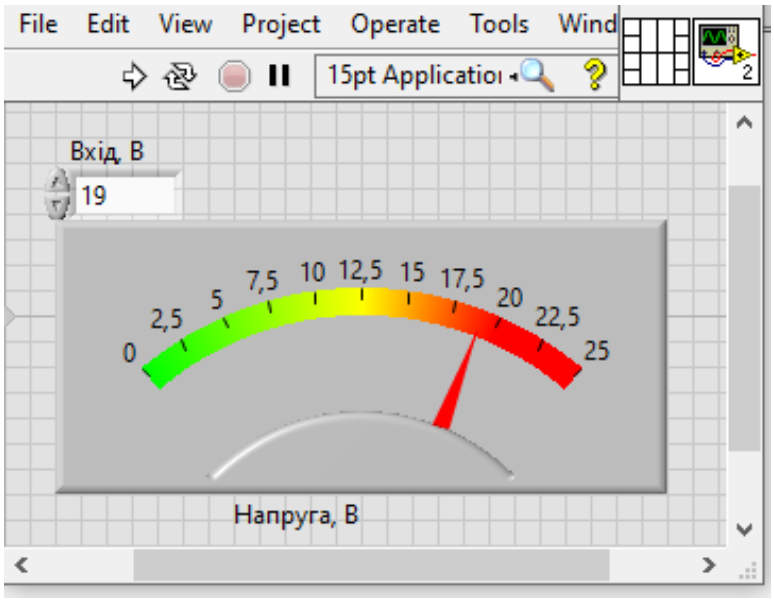


Рисунок 1.4 — Стрілочний індикатор

При цьому на блок-схемі автоматично з'явиться його термінал. Якщо межі шкали стрілочного індикатора будуть відмінні від необхідного діапазону, то їх можна замінити прямо на самій шкалі. Для цього треба подвійним натисканням миші виділити крайнє число шкали і записати туди з клавіатури нове значення. Вся шкала автоматично змінить свій масштаб відповідно до нових меж.

При оформленні лицьової панелі і блок діаграми прийнято показувати вимірювані фізичні величини та їх одиниці. Для оформлення блок-схеми і лицьової панелі, нанесення додаткових написів і коментарів, потрібно двічі натиснути мишею по вибраному місцю і ввести текст у відкрите вікно введення. Деякі декоративні елементи можна додати з палітри на лицьовій панелі *Controls>Modern>Decorations*.

Оформлення лицьової панелі і блок-діаграми − індивідуальна робота. Загальні правила для оформлення блок-діаграми: всі провідники розташовуйте так, щоб було чітко видно де вони починаються, проходять і закінчуються (мінімум пересікань). Блоки (вузли, термінали), виконувані та задані раніше, розташовуються вище і лівіше.

ПРИКЛАДИ ДЛЯ СТВОРЕННЯ ВІРТУАЛЬНОГО ПРИЛАДУ В ПРОГРАМНОМУ СЕРЕДОВИЩІ *LabVIEW*

Приклад 1.1 Складання двох чисел

На блок діаграмі будуємо ВП, який назвемо "*Sum X+Y*". Натисканням правої кнопки миші по полю блок-схеми викликаємо палітру *Functions*. Наводимо вказівник миші на потрібні блоки і послідовно переходимо до палітри *Functions>Programming>Numeric*.

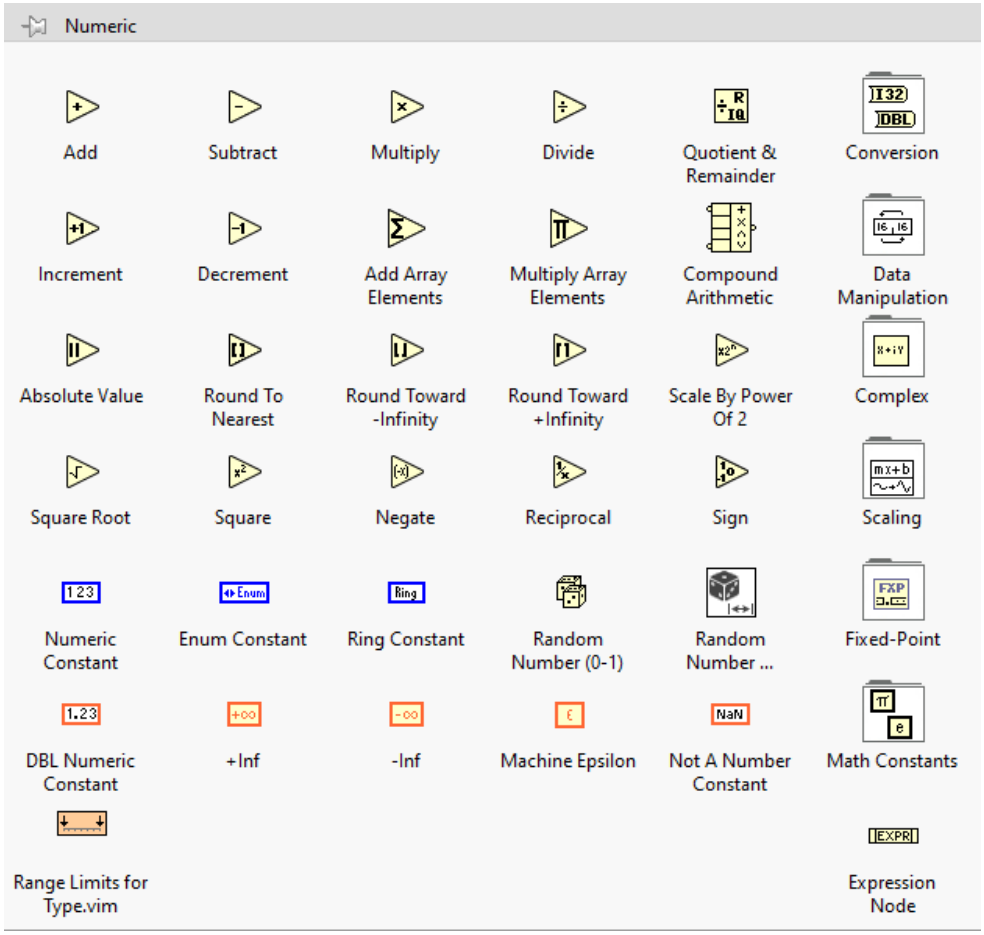


Рисунок 1.5 — Палітра *Numeric*

На палітрі *Numeric* натисканням миші вибираємо вузол складання *Add*. Далі перекладаємо вказівник миші на блок-схему (кнопка відпущена), і, натиснувши мишею по вибраному місцю блок-схеми, встановлюємо вузол. Тепер потрібно створити елементи введення двох чисел і виведення результату складання.

Найпростіший спосіб створення і приєднання стандартного терміналу введення (або виводу) до поля якого-небудь вузла − це поставити вказівник

миші на потрібне поле вузла і, коли вказівник прийме вигляд «сполучної котушки *Wiring Tool*»  , по натисканню правою кнопкою вибрати мишею у з'явившомуся меню *Create>Control* (або ж *Create>Indicator*).

На блок-схемі з'явиться термінал саме того типу, який вимагає це поле за замовчуванням, а на лицьовій панелі виникне елемент управління − *Control* (або, відповідно, індикатор) з тим же ярликом. Треба створити для двох полів входу вузла складання термінали введення − *Control* і для поля виводу − *Indicator*.

Розташування всіх елементів на блок-схемі і на лицьовій панелі можна змінити, виділяючи їх і переміщуючи при натиснутій лівій кнопці миші.

Схема готова до роботи. Перейдемо на лицьову панель, натиснувши на блок-схемі пункт меню *Window*>Show *Front Panel*, або натиснувши *Ctrl+e*. Запуск однократного виконання ВП проводиться за допомогою миші кнопкою Пуск (*Run*)  на інструментальній панелі, розташованій у верхній частині вікна програми, або ж з клавіатури, натиснувши *Ctrl+r*.

Якщо кнопка Пуск має "зламаний" вигляд  , це означає − в програмі є помилки. Треба натиснути цю кнопку, з'явиться вікно *Error list* / Список помилок. Після виправлення помилок кнопка набирає вихідного вигляду.

Для перевірки роботи програми введіть на лицьовій панелі у вікна введення два довільні числа і натисніть кнопку Пуск. На індикаторі повинен з'явитися правильний результат складання.

Внесіть тепер помилку до блок-схеми, наприклад, зітріть один з провідників на вході вузла складання. Для цього виділіть його мишею і натисніть на клавіатурі клавішу *Delete*. Натисніть Пуск, прочитайте опис з'явившоїся помилки. Натисніть далі кнопку *Show error*, поцікавтесь, який елемент з помилкою буде виділений на блок-схемі.

Відновіть стертий провідник. Для цього поставте вказівник миші на вихідне поле терміналу введення і коли вказівник прийме вигляд сполучної котушки (*Wiring Tool*)  , протягніть лінію при натиснутій лівій кнопці

миші до входу вузла складання, після чого відпустіть кнопку. Кнопка Пуск набере вихідного вигляду.

Розглянемо тепер виконання цієї ж операції із застосуванням вузла Формули (*Formula Node*), що дозволяє вводити в програми *LabVIEW* текстових операторів на мові С (табл. 1.1).

Таблиця 1.1 — Текстові оператори що використовуються в *LabVIEW*

**	Піднесення до ступеня
*, /	множення, ділення
+, -	додавання, віднімання

Для цього зберемо схему рис. 1.6. Вузол *Formula Node* викликається з палітри *Functions>Structures*. Після вибору вузла (натисканням миші) потрібно на блок-діаграмі намалювати рамку, що обмежує робочу область цієї структури.

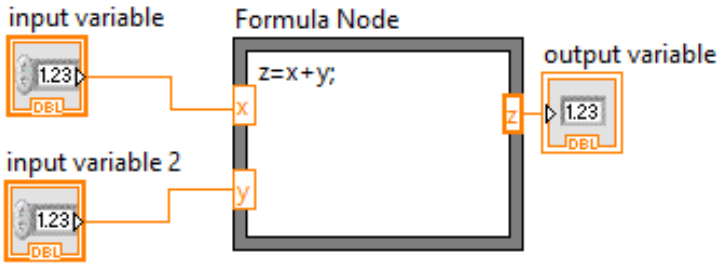


Рисунок 1.6 — Вузол *Formula Node*

Щоб створити термінали вхідних і вихідних даних, натисніть правою кнопкою миші по межі вузла. Далі в контекстному меню виберіть пункт *Add Input* (Додати вхід) або *Add Output* (Додати вихід), а потім введіть в поля даних терміналів ідентифікатори змінних для входу (*x*, *y*) і виходу (*z*) (великі і малі букви розрізняються).

Далі записуються оператори в робочу область структури. Кожен вираз повинен закінчуватися роздільником (;). Якщо будуть потрібні додаткові змінні, окрім введених для входу і виходу, то потрібно ввести їх опис за правилами мови *C*.

Приклад 1.2 Термометр

Створіть новий віртуальний прилад за допомогою команди *Blank VI* після запуску *LabVIEW*. Наведіть вказівником миші на лицьову панель і викличте палітру *Controls* натисненням правої клавіші маніпулятора. На вкладці *Numeric* виберіть регулятор *Knob*, на вкладці *Boolean* виберіть *Round Led*. Перейдіть до вікна діаграми, викличте палітру Functions і на вкладці *Comparison* виберіть значок більше (*Greater*?).

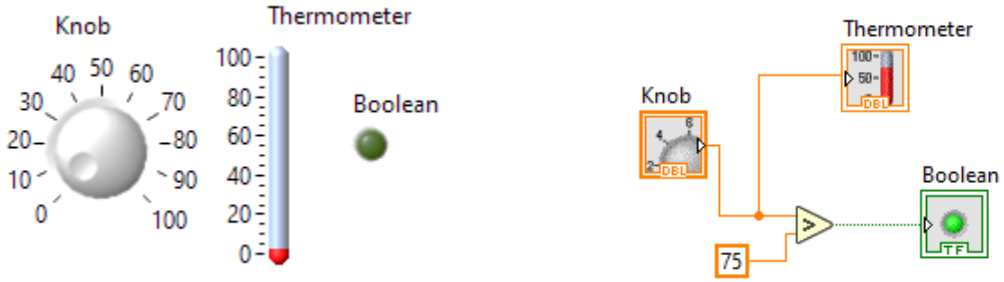
Вихід регулятора з'єднайте до одного з входів значка більше, а на інший вхід підключіть значення константи (вкладка *Numeric>Numeric Constant*) і встановіть значення константи 5. Вихід значка більше (*Greater*?) підключіть до значка *Round Led*. У Вас повинно вийти приблизно так, як показано на рис. 1.7.

а) б)

Рисунок 1.7 — Приклад віртуального приладу: а) передня панель; б) блок-діаграма

Перейдіть до лицьової панелі і запустіть програму в циклічному режимі. Змінюючи стани регулятора, переконайтеся, що світлодіод спалахує, при значеннях більш ніж 5. З меню *Numeric* лицьової панелі додайте термометр, змініть діапазон меж регулятора *Knob*, натисніть праву клавішу миші та визначте на регуляторі потрібні властивості (*Properties>Scale>Min*=0, *Max*=100).

На діаграмі підключіть вхід термометра до регулятора і змініть значення константи на вході значка більше (*Greater*?) на 75. Після цих змін ви повинні отримати приблизно наступне.



а) б)

Рисунок 1.8 — Приклад віртуального приладу з порівнянням значень температури: а) передня панель; б) блок-діаграма

Знову запустіть програму в режимі циклічного виконання, переконайтеся, що при зміні регулятора змінюються свідчення термометра, а при перевищенні рівня 75 спалахує світлодіод. Збережіть отриманий віртуальний прилад, наприклад, з ім'ям *Termometr.vi*.

Хід роботи

1.1) Виконати приклади для створення віртуальних приладів.

1.2) Отримати і ознайомитися з індивідуальним завданням.

1.3) Сформувати необхідні інструменти.

1.4) Налагодити програму і представити викладачеві для перевірки.

1.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

- звітність про виконання прикладів для створення віртуальних приладів;
- варіант і зміст індивідуального завдання;
- скріншоти складових програми і результати її виконання;
- відповіді на запитання до лабораторно роботи.

1.6) Представити звіт на захист.

Варіанти завдань

Завдання №1.1. Створити аналоговий вольтметр, діапазон відображуваних значень від 0 В до 20 В. Вхідну напругу імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення вхідною напругою рівня 18 В (червоний колір), і сигналізацію зменшення напруги до значень менше ніж 3 В (жовтий колір). Нормальні значення напруги сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку струму за законом Ома (значення опору вводиться).

Завдання №1.2. Створити аналоговий амперметр, діапазон відображуваних значень від 0 А до 5 А. Вхідний струм імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення вхідним струмом рівня 3,5 А, і сигналізацію зменшення струму до значень менше ніж 0,5 А (жовтий колір). Нормальні значення струму сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку напруги на постійному резисторі, через який протікає вимірюваний струм (значення опору вводиться).

Завдання №1.3. Створити аналоговий частотомір, діапазон відображуваних значень від 50 Гц до 550 Гц. Вхідне значення частоти імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення частотою рівня 600 Гц (червоний колір), і сигналізацію зменшення частоти до значень менше ніж 49,5 Гц (жовтий колір). Нормальні значення частоти сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку довжини хвилі вимірюваної частоти.

Завдання №1.4. Створити аналоговий фазометр, діапазон відображуваних значень зсуву фази від -90º до +90º. Вхідне значення фази імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення зрушення фази негативного значення -50 градусів (червоний колір), і позитивного значення

60 градусів (жовтий колір). Нормальні значення фазового зсуву сигналізуються зеленим кольором. Вбудувати калькулятор для переведення градусів у радіани.

Завдання №1.5. Створити аналоговий частотомір, діапазон відображуваних значень від 20 Гц до 15000 Гц. Вхідне значення частоти імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення частотою рівня 12000 Гц (червоний колір), і сигналізацію зменшення частоти до значень менше ніж 30,5 Гц (жовтий колір). Нормальні значення частоти сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку довжини хвилі вимірюваної частоти.

Завдання №1.6. Створити аналоговий ватметр постійного струму, діапазон відображуваних значень від 0 Вт до 1000 Вт. Вхідні значення постійної напруги в діапазоні від 0 В до 200 В і струму в діапазоні від 0 А до 5 А імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення потужністю рівня 910 Вт (червоний колір) і сигналізацію зменшення потужності до значень менше ніж 95 Вт (жовтий колір). Нормальні значення потужності сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку добутку струму та напруги.

Завдання №1.7. Створити аналоговий ватметр змінного струму, діапазон відображуваних значень від 0 кВт до 2,2 кВт. Вхідні значення змінної напруги в діапазоні від 0 В до 220 В, струму в діапазоні від 0 А до 10 А, зсув фаз в діапазоні від 0 градусів до 20 градусів імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення потужністю рівня 1900 Вт (червоний колір), і сигналізацію зменшення потужності до значень менше ніж 395 Вт (жовтий колір). Нормальні значення потужності сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку діючої напруги.

Завдання №1.8. Створити аналоговий вимірювач рівня рідини в

резервуарі з відображенням результату виміру на індикаторі *Tank*, діапазон відображуваних значень від 0 м до 15 метрів. Вхідне значення рівня імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення рівня 13,5 метрів (червоний колір), і сигналізація зменшення рівня до значень менше ніж 3,95 метра (жовтий колір). Нормальні значення рівня сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку об'єму рідини в резервуарі (резервуар циліндричної форми, радіус циліндра вводиться).

Завдання №1.9. Створити омметр, діапазон відображуваних значень від 0 кОм до 100 кОм. Вхідні значення опору імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення опором рівня 95 кОм (червоний колір), і сигналізацію зменшення опору до значень менше ніж 5,5 кОм (жовтий колір). Нормальні значення опору сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку тепла, що виділяється на резисторі (всі необхідні параметри вводяться).

Завдання №1.10. Створити цифровий вольтметр, діапазон відображуваних значень від 0 В до 500 В. Вхідну напругу імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. У приладі передбачити сигналізацію перевищення вхідною напругою рівня 480 В (червоний колір), і сигналізацію зменшення напруги до значень менше ніж 35 В (жовтий колір). Нормальні значення напруги сигналізуються зеленим кольором. Вбудувати калькулятор для розрахунку активної потужності (всі необхідні параметри вводяться).

Контрольні запитання

1.1) Особливості віртуальних приладів?

1.2) Основні відомості про середовище *LabVIEW*?

1.3) Яким чином здійснюється перехід між вікнами діаграми і лицьової панелі?

1.4) Як викликається палітра інструментів, палітра *Controls* і палітра *Functions*?

1.5) Призначення елементів верхнього меню в *LabVIEW*?

1.6) Призначення кнопок палітри інструментів?

1.7) Яким чином запустити програму на виконання?

1.8) Які засоби для полегшення налагодження програми є в *LabVIEW*?

ЛАБОРАТОРНА РОБОТА № 2
МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ У LABVIEW

Мета роботи: Здобуття практичних навиків написання програм в середовищі *LabVIEW* з використанням умовних операторів і циклів. Ознайомитися із способами організації розгалужень програм і циклів.

Стислі теоретичні відомості

Структури в *LabVIEW*

При розробці блок-схеми програми закладається алгоритм її роботи, порядок створюваних операцій. Якщо в текстових програмах послідовність визначається написаним текстом, то в *LabVIEW* поруч розташовані, але не пов'язані між собою, програмні блоки виконуються майже одночасно.

Тому, якщо важлива послідовність операцій, то потрібно забезпечити порядок обчислень і закласти його в структуру програми. У *LabVIEW* існує розділ, який дозволяє регламентувати цей процес. Це так звані структури. Виклик структур здійснюється з функціональної палітри меню *Structures*.

Flat Sequence (послідовна структура) − це інструмент складається з окремих сторінок (кадрів) на кожній з яких можна сформувати частину програми. Послідовність виконання фрагментів зліва направо. При цьому жодна із сторінок не пропускається. Процедура виконання завершиться, коли виконається частина програми на останній сторінці.

Stacked Sequence Structure − структура складається з однієї або декількох листів (кадрів), що виконуються послідовно, один за іншим відповідно до номера. Аналогічна за виконанням *Flat Sequence*, але істотно дозволяє заощадити місце на діаграмі.

Case Structure (структура з вибором) − це інструмент, схожий на умовного оператора текстових програм. Виконується код, що розташований на сторінці *True* або *False*, залежно від деякої зовнішньої умови.

For Loop (визначений цикл) − оператор циклу, в якому заздалегідь оговорено кількість повторюваних виконань коду.

While Loop (умовний цикл) − оператор циклу, в якому число повторень заздалегідь невідоме, є лише додаткова умова виходу з циклу.

Timed Loop − виконується частина діаграми або декілька вікон, послідовно в циклі зі встановленим користувачем періодом.

Приклад 2.1

Створимо програму, яка визначає час виконання певного циклу. У програмі використаємо послідовність з трьома кадрами. Розмістіть на блок-діаграмі структуру *Stacked Sequence Structure* з палітри *All Functions>Programming>Structures>Flat Sequence* натиснути правою кнопкою на границю та перетворити в *Stacked* "*To Stacked Sequence*".

Для створення кадрів послідовності необхідно підвести вкахівник миші на границі області структури і натиснути праву кнопку миші. У з'явившомуся меню необхідно вибрати *Add Frame After* (Додати кадр). Таким чином, створимо 3 кадри (0...2).

Змінні, які використовуються для передачі даних між кадрами — Локальні Змінні Послідовності (*Sequence locals*).

У початковий (0-й) кадр помістимо компонент *Tick Count*, який прочитує поточне значення системного таймера і повертає результат в мілісекундах. Ця підпрограма може бути завантажена з меню *Functions>Timing>Tick Count* (*ms*). Далі створимо Локальну Змінну підведенням вказівника миші до межі структури, натискуємо праву кнопку миші та у випадаючому меню обираємо *Add Sequence Local* (Додати Локальну Змінну Послідовності).

Сполучаємо виведення *Tick Count* (*ms*) з терміналом нової локальної змінної. В результаті, усередині нього з'явиться стрілка, яка вказуюча на те, що дані поступають з поточного кадру.

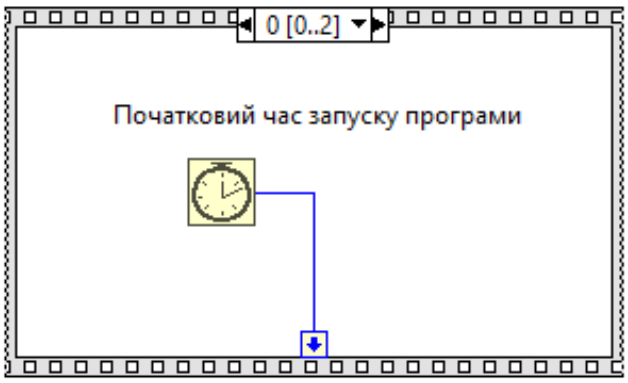


Рисунок 2.1 — Перший кадр струтури *Stacked Sequence Structure*

Створіть в другому кадрі програму по виведенню випадкових чисел у вигляді графіка за допомогою генератора випадкових чисел. Програма реалізується у вигляді:

циклу *Functions>Structures>While-Loop*, умовою виходу з якого є натиснення кнопки «зупинка»,

генератора випадкових чисел *Function>Numeric>Random Number*(0-1),

графіка *Controls>Graph>WaveForm Chart* та

кнопки зупинки *Controls>Boolean>Stop Button*.

На передній панелі помістіть графік *WaveForm Chart*. З'єднайте його термінал на блок-діаграмі з виходом приладу випадкових чисел. Кнопку *Stop Button* з'єднайте з кнопкою зупинки циклу.

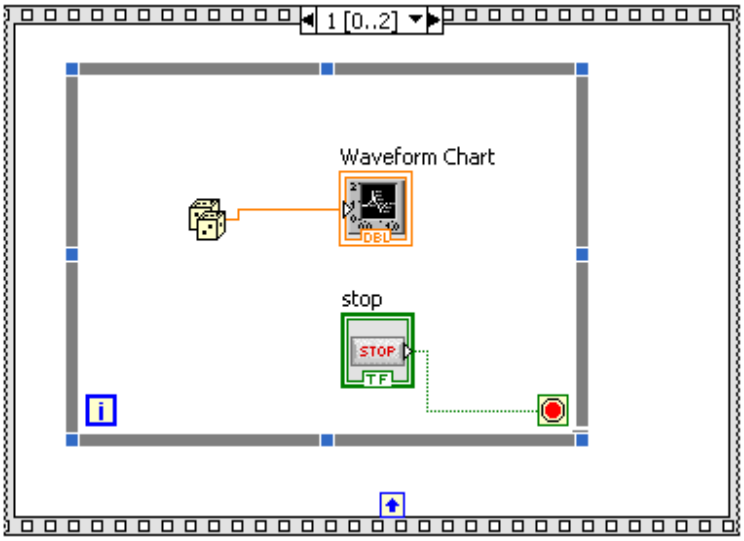
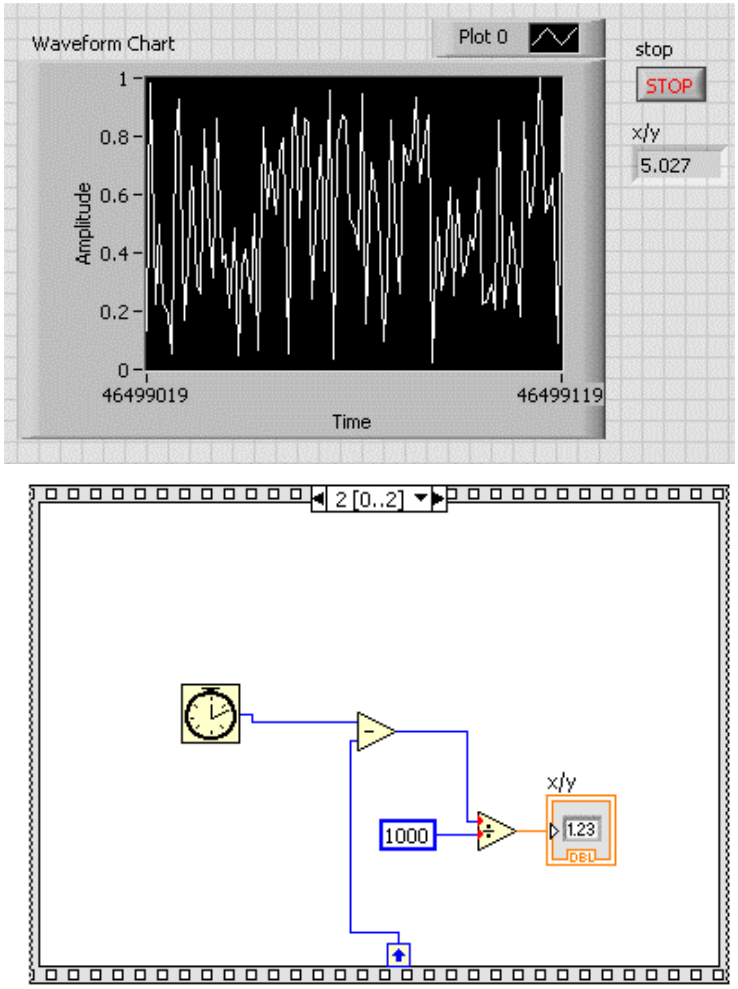


Рисунок 2.2 — Другий кадр струтури *Stacked Sequence Structure*

У останньому кадрі обчислимо різницю за часом. Для цього порівняємо поточне значення часу із значенням, отриманим в нульовому кадрі. Використаэмо значення, отримане в першому кадрі, з'єднавши Локальну Змінну (стрілку в квадратику) з відповідним виводом. Переводимо мілісекунди в секунди шляхом ділення значення на 1000 і виводимо результат на цифровий індикатор, заздалегідь встановивши його на інтерфейсній панелі.



а)

б)

Рисунок 2.3 — Третій кадр струтури *Stacked Sequence Structure*:
а) передня панель; б) блок-діаграма

Помістіть всередину функцію *Tick Count* (*ms*) з палітри *All Functions >Programming> Time*. Додайте в цей кадр послідовності локальну змінну, вибравши в контекстному меню на рамці послідовності пункт *Add Sequence Local*. З'єднайте вихід функції *Tick Count* (*ms*) з цією локальною змінною.

Приклад 2.2

Створимо прилад, що реалізовує послідовне запалення ламп - «біжучий вогник». Помістіть на передній панелі 4 індикатори типу *Round LED*. Відкрийте діалог налаштувань віртуального приладу, виберіть в контекстному меню на піктограмі ВП пункт *VI Properties*. Перейдіть до категорії Execution, і виставіть прапорець *Clear Indicators When Called*.

Розмістіть на блок-діаграмі структуру *Flat Sequence Structure*. Створіть в ній 4 кадри, та розмістіть в кожному по одному індикатору *Round LED*, і подайте на вхід кожного значення логічної істини (*true*). Створіть проміжні кадри, в яких би здійснювалася затримка виконання програми (рис.2.4).

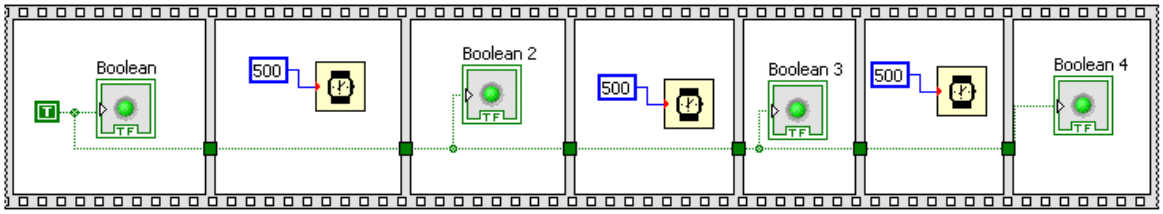


Рисунок 2.4 — Блок-діаграма до прикладу 2.2

Роботу приладу вивчіть самостійно.

Приклад 2.3 Використання оператора *For Loop*

Структура «Цикл з фіксованим числом ітерацій» (*For Loop*) еквівалентна текстовому операторові *for i* = 0 *to N* - 1 *do* ... При внесенні структури на панель блок-діаграми, її контур у вигляді прямокутника має бути розтягнутий так, щоб охопити існуючий код програми, який повинен виконуватися циклічно задане число разів, або так, щоб дозволити розмістити в ньому новий код програми.

Якщо внесена в структуру або переміщувана усередині структури, функція перетинається з її границею, то границя автоматично розширюється. Ця опція може бути відключена для цієї структури шляхом зняття відмітки рядка *Auto Grow* в контекстному меню структури або для всього додатку шляхом зняття відмітки рядка «Встановити структури з

автоматичним розширенням» (*Place structures with Auto Grow enabled*) діалогового вікна «Опції» (*Options*).

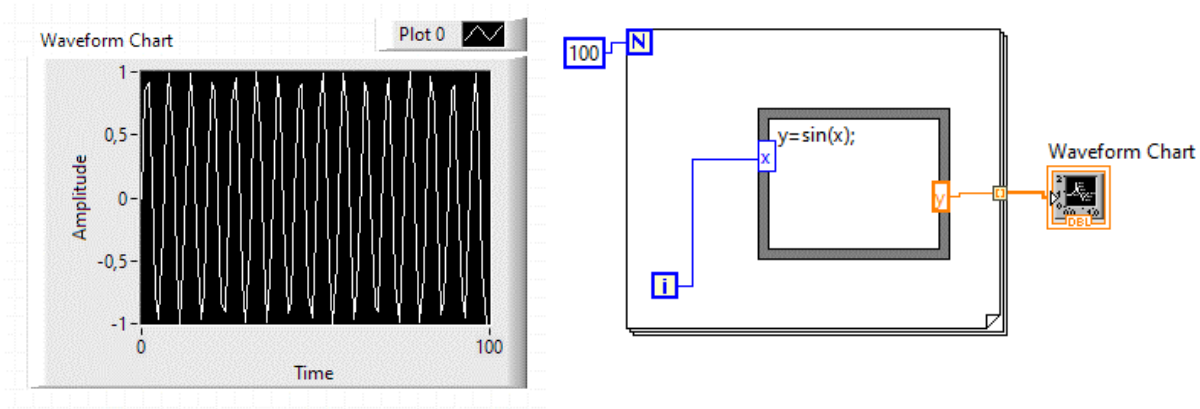
Кількість циклів може задаватися за допомогою константи або елементу управління, підключених до «терміналу числа ітерацій» (*count terminal*) (прямокутник в лівому верхньому кутку структури з буквою *N*). Поточне число завершених ітерацій циклу міститься в «терміналі лічильника ітерацій» (*iteration terminal*).

Приклад 2.4

Створіть новий порожній віртуальний прилад, вибравши команду *Blank VI* після запуску *LabVIEW*. Розмістіть на діаграмі цикл *For Loop* (палітра *Controls*, меню *Structures*), у якості опції для *N* встановіть значення 100. Розмістіть усередині циклу оператор *Formula Node* (палітра *Controls*, меню *Structures*) і впишіть всередину вираз $y=\sin(x)$.

Вхід *Formula Node* підключіть до *Loop iteration* (*i*), а вихід ***Y*** до виходу циклу. На лицьовій панелі розмістіть *Waveform Chart* і підключіть його вхід до виходу циклу. Програма розраховує 100 значень синусоїди і відображує сигнал на екрані *Waveform Chart*.

Повинно вийти приблизно наступне, рис. 2.5.



а) б)

Рисунок 2.5 — Приклад застосування циклу *For Loop*: а) передня панель; б) блок-діаграма

Запустіть програму на виконання і переконайтеся, що на екрані з'явиться синусоїда.

Використання оператора Case Structure

Структура «Варіант» (*Case Structure*) аналогічна операторам *case* або *if then-else* в текстових мовах програмування. За умовчуванням структура «Варіант» є логічною і має два варіанти - ІСТИНА (*TRUE*) і ХИБНІСЬ (*FALSE*), що обираються за допомогою терміналу селектора структури варіанту. Структура автоматично перетвориться в числову або строкову при підключенні відповідно до числової або строкової змінної терміналу селектора.

В цьому випадку структура може мати практично необмежену кількість варіантів, починаючи з нульового. За допомогою рядків «Додати варіант після» (*Add Case After*) або «Додати варіант перед» (*Add Case Before*) можна додати новий варіант або до поточного варіанту. Одночасно можна спостерігати лише один «варіант» (кадр) структури.

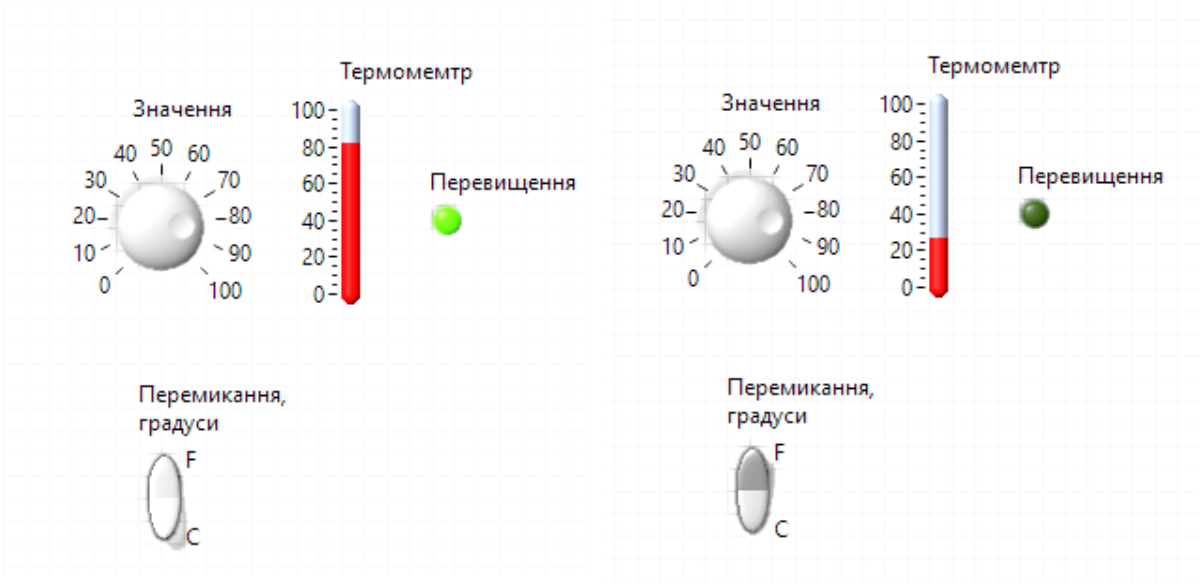
Перехід між варіантами виконується за допомогою «селектора структури варіанту», розташованого на верхній стороні рамки структури або контекстного меню структури. Для використання структури «Варіант» необхідно відзначити «варіант за умовчуванням» (*Default*).

Ввід і вивід даних в структуру «Варіант» виконується за допомогою вхідних і вихідних терміналів даних (тунелів). Створення вихідного терміналу даних на одній піддіаграмі структури приводить до його появи на інших піддіаграмах в тому ж самому місці межі структури. До підключення даних до вихідного терміналу у всіх піддіаграмах він зберігає білий колір і сприймається як похибка створення.

Приклад 2.5

Створіть і відкрийте збережений в першій лабораторній роботі віртуальний прилад *Termometr.vi*. Необхідно перетворити віртуальний прилад так, щоб він міг відображувати результат в градусах Цельсія і

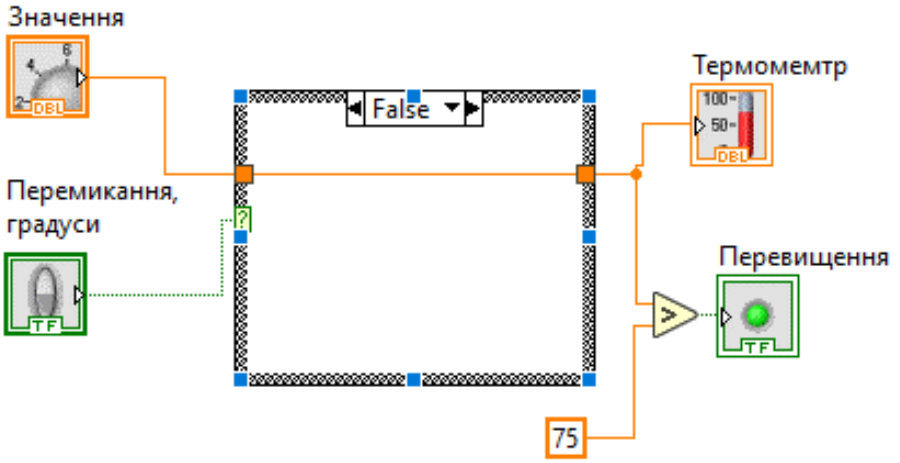
Фаренгейта (рис. 2.6). Відомо, що температура за Цельсієм пов'язана з температурою за Фаренгейтом співвідношенням *C*=(*F*-32)*(5/9).



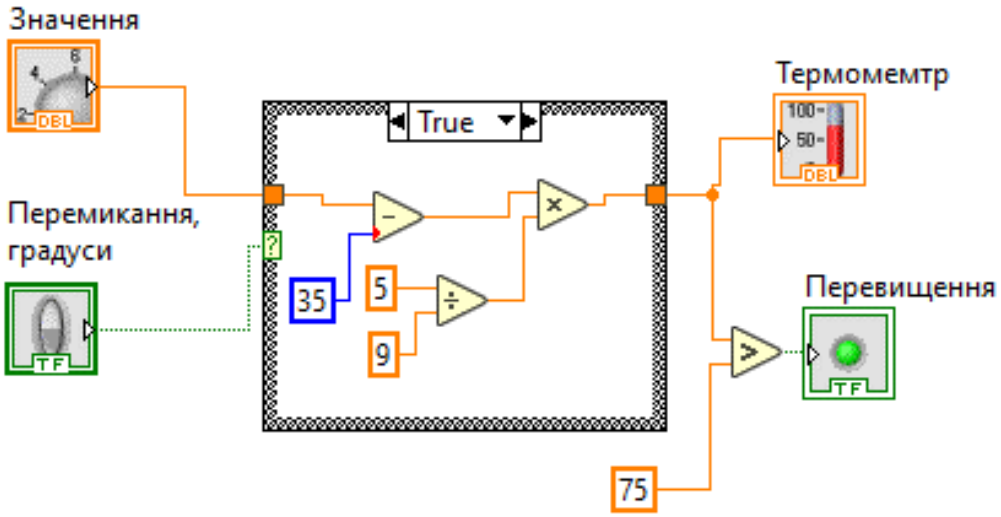
а) б)

Рисунок 2.6 — Передня панель ВП до прикладу 2.5:
а) відображення в ^{o}F; б) відображення в ^{o}C

Для зміни функціональних можливостей приладу на лицьовій панелі розмістіть перемикач *Vert Rocker*, а на діаграмі додайте *Case Structure* і селектор підключіть до виходу перемикача *Vert Rocker*, а на вхід подайте сигнал від регулятора *Knob*. На вкладці *True Case Structure* запишіть вираз перетворення температури за Фаренгейта до значення за Цельсієм. Вихід *Case Structure* підключіть до термометра. На вкладці *False* вхід і вихід з'єднайте напряму, без виконання дій. В результаті повинно вийти так, як показано на рис. 2.7.



а)



б)

Рисунок 2.7 — Блок-діагорама до ВП з прикладу 2.5: а) *False Case Structure*; б) *True Case Structure*

Запустіть програму в режимі циклічного виконання і переконайтесь, що при зміні положення перемикача змінюються результати, що відображуються на термометрі. Відкрийте на екрані одночасно вікно лицьової панелі і вікно діаграми. У вікні діаграми натисніть кнопку *Highlight Execution LJ* і простежте за проходженням даних по діаграмі. Змініть стан перемикача *Vert Rocker* і переконайтеся, що відбулося перемикання *Case Structure*, і виконання програми відбувається по альтернативному шляху.

Використання оператора *While Loop*

Структура Цикл за умови (*While Loop*) еквівалентна наступному псевдокоду: *do* {програма} *while* {умова}. Усередині структури розміщуються «термінал лічильника ітерацій» (*iteration terminal*) «*i*» та «термінал умови виходу з циклу» (*conditional terminal*).

Код програми, розміщений в структурі, виконується до подачі на термінал умови логічної змінної ІСТИНА (*TRUE*) або ХИБНІСТЬ (*FALSE*). Зміна варіанту припинення виконання виконується за допомогою рядків «Зупинити якщо істина» (*Stop If true*) або «Продовжити якщо істина» (*Continue If True*) контекстного меню терміналу умови.

Якщо термінал умови не підключений до якого-небудь виходу, то цикл не виконуватиметься. Структура *While Loop* на відміну від циклу *For Loop* виконується завжди (хоча б один раз), оскільки умова зупинки перевіряється в кінці виконання циклу.

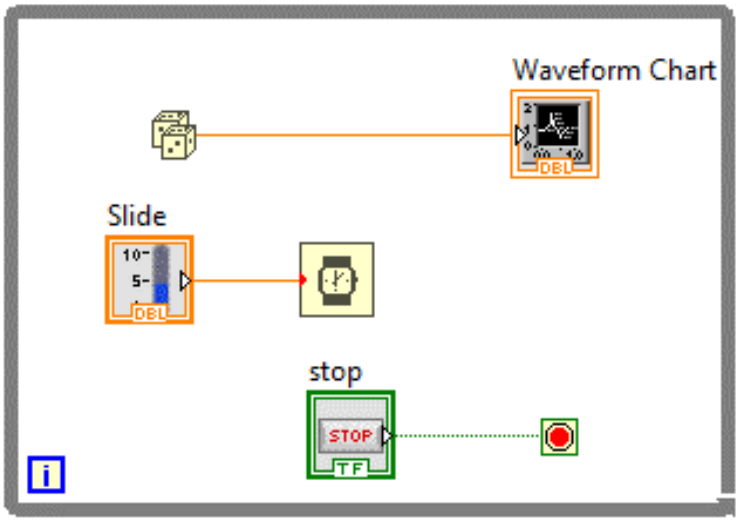
Слід пам'ятати, що умову виходу з циклу не можна задавати ззовні циклу. Якщо його значення при вході в цикл було помилкове, то цикл виконається один раз, а якщо воно було достеменне, то цикл продовжуватиметься до тих пір, поки його не зупинять кнопкою СТОП в командному рядку верхнього меню.

Приклад 2.6

Розробимо прилад для генерації і відображення випадкових чисел з використанням структури *While Loop*. Для цього створіть новий порожній віртуальний прилад, вибравши команду *Blank VI* після запуску *LabVIEW*. На передній панелі розмістіть регулятор *Vertical Pointer Slider* (*Controls*, *Modern*, *Numeric*), кнопку *Stop* («зупинка») *Stop Button* (*Controls*, *Modern*, *Boolean*) і вікно для відображення результату генерації числа в графічному вигляді *Waveform Chart* (*Controls*, *Modern*, *Graph*).

Розмістіть на діаграмі цикл *While Loop* (палітра *Controls*, меню *Structures*) і внесіть до нього піктограми *Vertical Pointer Slider*, *Waveform Chart*, *Stop Button*. Вихід *Vertical Pointer Slider* підключіть до елементу затримки (*Functions*, *Programming*, *Timing*, *Wait(ms)*). *Stop Button* підключіть до *Loop Condition*. На вхід *Waveform Chart* подайте сигнали з генератора випадкових чисел (*Functions*, *Programming*, *Numeric*, *Random Number*(0-1)).

а)



б)

Рисунок 2.8 — ВП до прикладу 2.6: а) передня панель; б) блок-діаграма

Запустіть програму на виконання, переконаєтеся, що відбувається генерація випадкових чисел, а зупинка програми можлива по натисненню кнопки Stop. Змініть властивість Loop Condition на Continue if True. Знову запустіть програму і переконаєтеся, що вона закінчує роботу після генерації одного випадкового числа.

Хід роботи

2.1) Виконати приклади для створення віртуальних приладів.

2.2) Отримати і ознайомитися з індивідуальним завданням.

2.3) Сформувати необхідні інструменти.

2.4) Налагодити програму і представити викладачеві для перевірки.

2.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

- звітність про виконання прикладів для створення віртуальних приладів;
- варіант і зміст індивідуального завдання;
- скріншоти складових програми і результати її виконання;
- відповіді на запитання до лабораторно роботи.

2.6) Представити звіт на захист.

Варіанти завдань

Завдання №2.1. Створити аналоговий вольтметр, робочий діапазон від 0 В до 2000 В. Вхідну напругу імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення напруги в «В» та «кВ». Вбудувати прилад для генерації випадкового рівня напруги кожні 2 с протягом 40 с у заданому діапазоні напруг та за допомогою тумблера подати на аналоговий вольтметр (використати цикл *For Loop*).

Завдання №2.2. Створити аналоговий амперметр, робочий діапазон від 0 А до 5 А. Вхідний струм імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення сили струму в «А» та «мА». Вбудувати прилад для генерації випадкового значення сили струму кожні 3 с протягом 30 с у заданому діапазоні вимірювання струму та за допомогою тумблера подати на аналоговий амперметр (використати цикл *For Loop*).

Завдання №2.3. Створити аналоговий частотомір, робочий діапазон від 1000 Гц до 55000 Гц. Вхідне значення частоти імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення частоти в «Гц» та «кГц». Вбудувати прилад для генерації випадкового значення частоти кожні 2 с у заданому діапазоні вимірювання частоти та за допомогою тумблера подати на аналоговий частотомір (використати цикл *While Loop*).

Завдання №2.4. Створити аналоговий фазометр, робочий діапазон зсуву фази від -90 до 90 градусів. Вхідне значення фази імітувати за

допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення фази в «градусах» та «радіанах». Вбудувати прилад для генерації випадкового зсуву фази кожні 2 с у заданому діапазоні вимірювання та за допомогою тумблера подати на аналоговий фазометр (використати цикл *While Loop*).

Завдання №2.5. Створити аналоговий частотомір, робочий діапазон від 200000 Гц до 1500000 Гц. Вхідне значення частоти імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення частоти в «кГц» та «МГц». Вбудувати прилад для генерації випадкового значення частоти кожні 3 с протягом 60 с у заданому діапазоні вимірювання та за допомогою тумблера подати на аналоговий частотомір (використати цикл *For Loop*).

Завдання №2.6. Створити аналоговий ватметр постійного струму, робочий діапазон від 0 Вт до 1000 Вт. Вхідні значення постійної напруги в діапазоні від 0 В до 200 В і струму в діапазоні від 0 А до 5 А імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення потужності в «Вт» та «кВт». Вбудувати прилад для генерації випадкового рівня напруги кожні 2 с протягом 40 с у заданому діапазоні напруг та за допомогою тумблера подати на аналоговий ватметр (використати цикл *For Loop*).

Завдання №2.7. Створити аналоговий ватметр змінного струму, робочий діапазон від 0 кВт до 2,2 кВт. Вхідні значення змінної напруги в діапазоні від 0 В до 220 В, струму в діапазоні від 0 А до 10 А, зсув фаз в діапазоні від 0 до 20 градусів імітувати за допомогою довільних регуляторів

з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення потужності в «Вт» та «кВт». Вбудувати прилад для генерації випадкового значення сили струму кожні 4 с у заданому діапазоні зміни струму та за допомогою тумблера подати на аналоговий ватметр (використати цикл *While Loop*).

Завдання №2.8. Створити аналоговий вимірювач рівня рідини в резервуарі з відображенням результату виміру на індикаторі *Tank*, робочий діапазон від 0 м до 15 метрів. Вхідне значення рівня імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення довжини в «метрах» та «дюймах». Вбудувати прилад для генерації випадкового рівня рідини кожні 2 с протягом 30 с у заданому діапазоні вимірювання та за допомогою тумблера подати на аналоговий вимірювач рівня рідини (використати цикл *For Loop*).

Завдання №2.9. Створити омметр, робочий діапазон від 0 кОм до 100 кОм. Вхідні значення опору імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення опору в «Ом» та «кОм». Вбудувати прилад для генерації випадкового рівня опору кожні 3 с у заданому діапазоні вимірювання та за допомогою тумблера подати на аналоговий омметр (використати цикл *While Loop*).

Завдання №2.10. Створити цифровий вольтметр, робочий діапазон від 0 В до 5 В. Вхідну напругу імітувати за допомогою довільних регуляторів з вкладки *Numeric* меню *Controls*. Забезпечити зміну функціональних можливостей приладу на лицьовій панелі за допомогою перемикача *Vert Rocker* та *Case Structure*. Прилад повинен відображати значення напруги в

«мВ» та «В». Вбудувати прилад для генерації випадкового рівня напруги кожні 3 с протягом 60 с у заданому діапазоні вимірювання та за допомогою тумблера подати на цифровий вольтметр (використати цикл *For Loop*).

Контрольні запитання

2.1) Як за допомогою структури послідовності можна забезпечити порядок виконання програми?

2.2) Назвіть та охарактеризуйте основні структури в *LabVIEW* для організації розгалуження програми?

2.3) Чим відрізняються структури *Flat Sequence* і *Stacked Sequence* одна від одної?

2.4) У який момент дані, поміщені у вихідний тунель в першому кадрі багатокадрової послідовності, вийдуть за межі структури?

2.5) Яким способом можна передати дані в наступний кадр в структурі Stacked Sequence?

2.6) У чому полягає основна відмінність циклів *For Loop* і *While Loop*?

2.7) Як задається умова в структурі *Case*?

2.8) За допомогою яких засобів оператор може змінювати вхідні дані в програмах?

ЛАБОРАТОРНА РОБОТА № 3
ПРОГРАМУВАННЯ ОПЕРАЦІЙ ТА ГРАФІЧНЕ ПРЕДСТАВЛЕННЯ ДАНИХ В LABVIEW

Мета роботи: Вивчення практичних прийомів і особливостей використання різних засобів відображення графічної інформації в *LabVIEW*.

Стислі теоретичні відомості

Засоби відображення графічної інформації в *Labview*

LabVIEW володіє потужним апаратом графічного представлення інформації. У *LabVIEW* є різні види графіків: *Waveform Chart, Waveform Graph, X-Y Graph, Mixed Signal Graph* та інші.

Waveform Chart відноситься до найпростіших засобів відображення інформації. Користувачеві достатньо підключити *Waveform Chart* до виходу своєї програми. В ньому вісь X організована дуже примітивно, типу самописця, це так званий стрічковий графік.

Waveform Graph виводить сигнал на відображення не по окремим точкам, а лише масивом точок. Це графік, що використовує масив сигналу. Користувач може відразу використовувати цей графік.

X-Y *Graph* (графік функціональної залежності $y = f(x)$) вимагає для своєї роботи створення окремого масиву відображуваного сигналу (Y) і окремого масиву його аргументу (X). Такий графік не можна безпосередньо підключати до джерела сигналу, потрібна певна організація масивів X і Y. Виклик і установка графіків здійснюється на лицьовій панелі з набору *Controls>Graph*.

Використання *Waveform Chart*. Це найбільш простий графік. Підключений до будь-якого джерела цифрової інформації, він дозволяє в реальному масштабі часу миттєво відобразити числове значення. Приклад самописця, в якому стрічка рухається з деякою швидкістю по горизонтальній осі, а графік відображує положення "пера" на вертикальній осі. *Waveform Chart* єдиний вид графіків, який може відображувати

інформацію по окремим точкам, відображувати масиви, причому принцип самописця залишається.

Розміри графіка (так само як для інших засобів відображення графічної інформації в *LabVIEW*) можна змінювати курсором «позиціювання». Розміри панелі (разом зі шкалами, маркерами, мітками шкал) і розміри вікна (з відображуваними сигналами) графіка можна змінювати окремо один від одного. За умовчуванням при зміні розміру панелі розмір вікна теж змінюється. При цьому також змінюють своє положення всі встановлені атрибути графіка. Зміна самого вікна на інші елементи не впливає.

Для графіків можна задавати межі по осях текстовим курсором. Підведіть курсор до мінімальної або максимальної межі, натисніть на нім лівою кнопкою миші (ЛКМ), і введіть необхідне значення межі. Основна робота з графіком і його атрибутами здійснюється через контекстне меню самого графіка і контекстні меню його атрибутів, встановлених на передній панелі.

Графік *Waveform Chart* використовує три різні режими відображення даних: панорамування діаграми (*strip chart*), тимчасова розгортка (*scope chart*) і тимчасова розгортка з маркером (*sweep chart*).

Режим за умовчуванням − панорамування діаграми (*strip chart*). Вибір режиму здійснюється натисканням правою клавішею миші (ПКМ) по діаграмі, потім вибором пункту *Advanced>Update Mode* з контекстного меню.

Режимом панорамування діаграми (*strip chart*) є екран, що прокручується зліва направо, подібно до паперової стрічки. Режими тимчасова розгортка (*scope chart*) і тимчасова розгортка з маркером (*sweep chart*) подібні до екрану осцилографа і відрізняються більшою швидкістю відображення даних в порівнянні з режимом панорамування діаграми (*strip chart*).

У режимі тимчасова розгортка (*scope chart*) після досягнення правої границі поле графіка очищається і рисування діаграми починається з лівої

границі. Режим тимчасова розгортка з маркером (*sweep chart*), на відміну від режиму тимчасова розгортка (*scope chart*), не очищає поле графіка, а поточне значення діаграми позначається вертикальною лінією − маркером.

Приклад 3.1. Використання *Waveform Chart*

За допомогою *Waveform Chart* розробимо новий віртуальний прилад. Створіть новий порожній віртуальний прилад, вибравши команду *Blank VI* після запуску *LabVIEW*. Розмістіть на лицьовій панелі 5 вікон з *Waveform Chart*, вибравши їх з меню *Controls>Graph* і кнопку *Stop Button* (*Controls*, *Modern*, *Boolean*). Для першої панелі виберіть властивість − *strip chart*, для другої − *scope chart*, а для третьої − *sweep chart* (рис. 3.1).

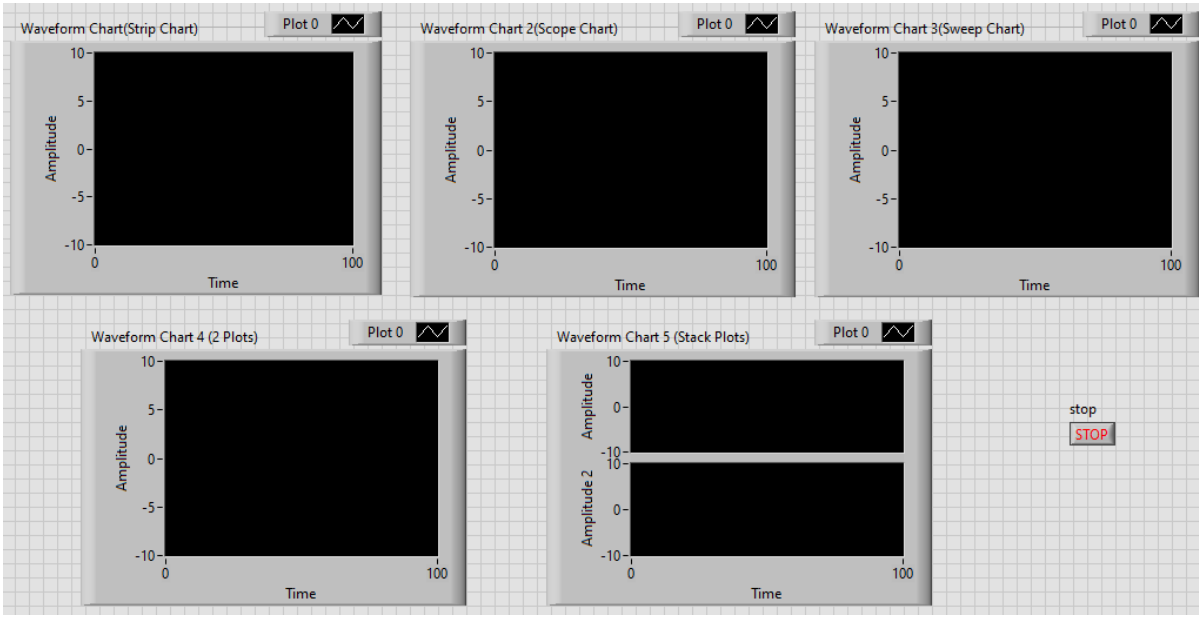


Рисунок 3.1 — Передня панель ВП до прикладу 3.1 з використанням *Waveform Chart*

Для п'ятої панелі у властивостях виберіть опцію *Stack Plots*. На панелі діаграм розмістіть цикл *While Loop* і помістіть піктограми всіх п'яти *Waveform Chart* у тіло циклу.

Кнопку *Stop* підключіть до умови закінчення циклу. Для генерації відображуваних сигналів, скористайтеся номером ітерації і циклу *While Loop*.

З палітри *Functions>Mathematics>Elementary>Trigonometric* виберіть функції 

і 

(синус і косинус відповідно). Для уповільнення роботи

циклу вставте блок затримки  з палітри *Functions>Programming>Timing*.

На 1, 2 і 3 вікно *Waveform Chart* подайте сигнал з виходу формувача синусоїди, а на вхід 4 і 5 вікон *Waveform Chart* подайте сигнал від формувачів синусоїди і косинусоїди пропущений через функцію Об'єднання (*Bundle*) (*Functions>Claster, Class, & Variant*) (рис. 3.2).

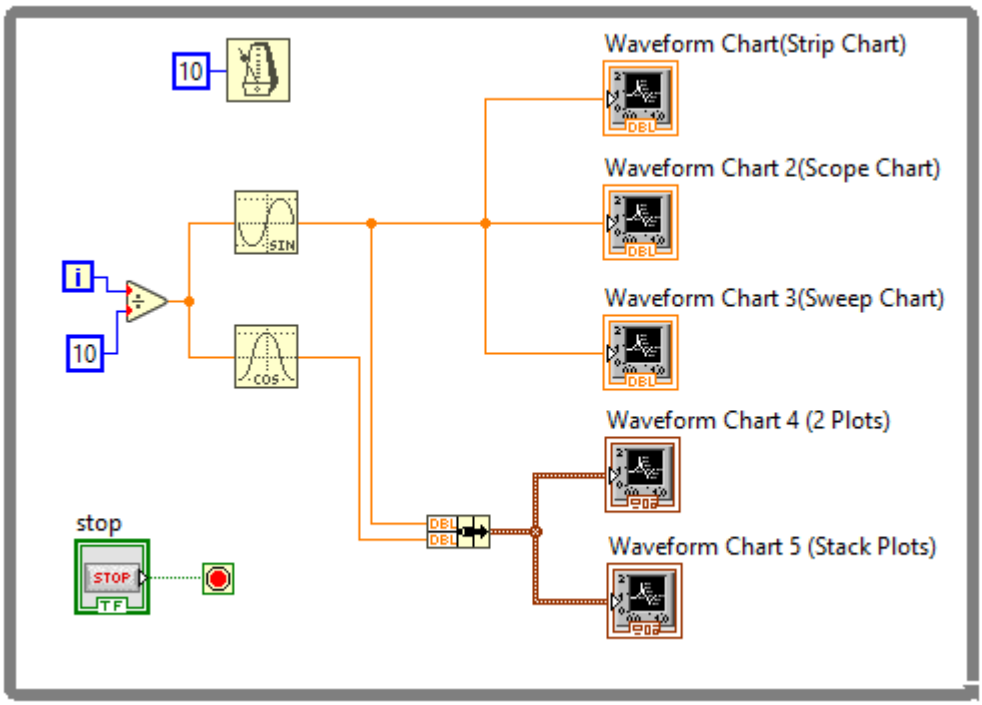


Рисунок 3.2 — Блок-діаграма ВП до прикладу 3.1 з використанням *Waveform Chart*

Налаштуйте програму і запустіть її, проаналізуйте відмінності в представленні інформації в різних режимах роботи *Waveform Chart*.

Особливості використання Waveform Graph і X-Y Graph

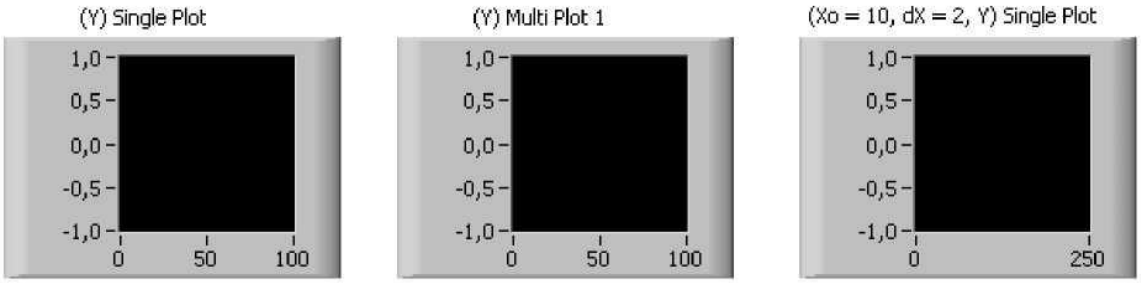
Графік осцилограм (*Waveform Graph*) відображує лише однозначні функції, такі як $y = f(x)$, з рівномірно розподіленими точками по осі X. Двокоординатний графік осцилограм (*X-Y Graph*) відображує будь-який набір точок, будь-то рівномірно розподілена вибірка чи ні.

Графік осцилограм (*Waveform Graph*) і двокоординатний графік осцилограм (*X-Y Graph*) автоматично підтримують режим відображення множини осцилограм.

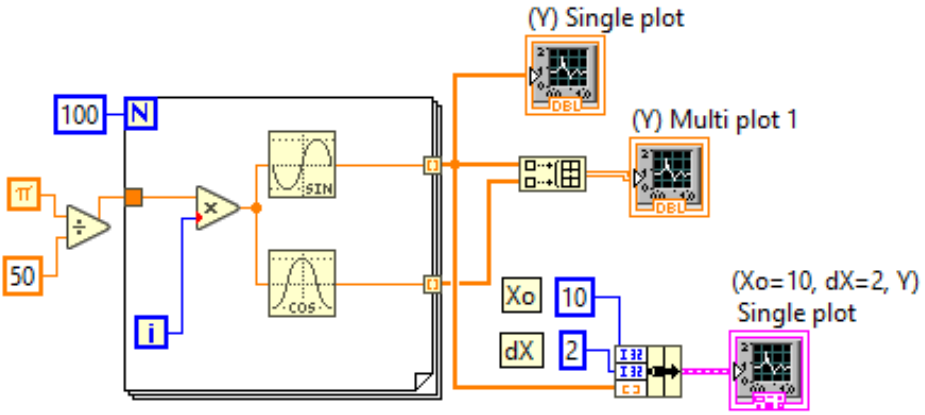
Одиночний графік осцилограм (*Waveform Graph*) працює з одновимірними масивами і представляє дані масиву у вигляді точок на графіку, з приростом по осі X рівним «1» і початком в $x=0$. Графіки також відображують дані кластерів, зі встановленим початковим значенням x, Ax і масивом даних за шкалою Y.

Приклад 3.2. Використання *Waveform Graph*

Використання *Waveform Graph*. Для цього створіть новий віртуальний прилад. На лицьовій панелі розмістіть три вікна *Waveform Graph*. На панелі діаграм розмістіть цикл *For Loop* у тілі якого розташуєте функції  і  (синус і косинус відповідно), які в циклі сприйматимуть номер ітерації циклу і формуватимуть один елемент (рис. 3.3).



а)



б)

Рисунок 3.3 — ВП до прикладу 3.2 з використанням *Waveform Graph*:
а) передня панель; б) блок-діаграма

Всього в циклі використайте 100 ітерацій, вхідним для циклу значенням використайте дробову частину (наприклад, одну п'ятдесяту від числа π). Таким чином, будуть сформовані два масиви чисел, що описують один період синусоїди і косинусоїди (кожен складається із ста значень).

Сигнал з виходу формувача синусоїди подайте на вхід першого вікна *Waveform Graph*, на вхід другого вікна *Waveform Graph* подайте сигнали з виходів формувача синусоїди і косинусоїди, заздалегідь пропустивши їх через функцію Побудова масиву (*Build Array*) (*Functions>Programming>Array*).

На вхід третього вікна подайте сигнал з виходу формувача синусоїди заздалегідь об'єднавши цей сигнал за допомогою функції Об'єднання *Bundle* (*Functions>Programming>Claster, Class, & Variant*) з початковим значенням по осі $X0$ (наприклад, $X0=10$) і кроком зміни даних по осі AX (наприклад, $AX=2$).

Налаштуйте програму і запустіть її, проаналізуйте зміни в зображенні графіка в третьому вікні якщо змінюватимуться значення $X0$ і AX.

Двокоординатні графіки *X-Y Graph*

Двокоординатні графіки *X-Y Graph* призначені для відображення багатозначних функцій в Декартовій системі координат (замкнуті криві, розподіл осцилограми в часі зі змінною часовою базою) і є універсальними.

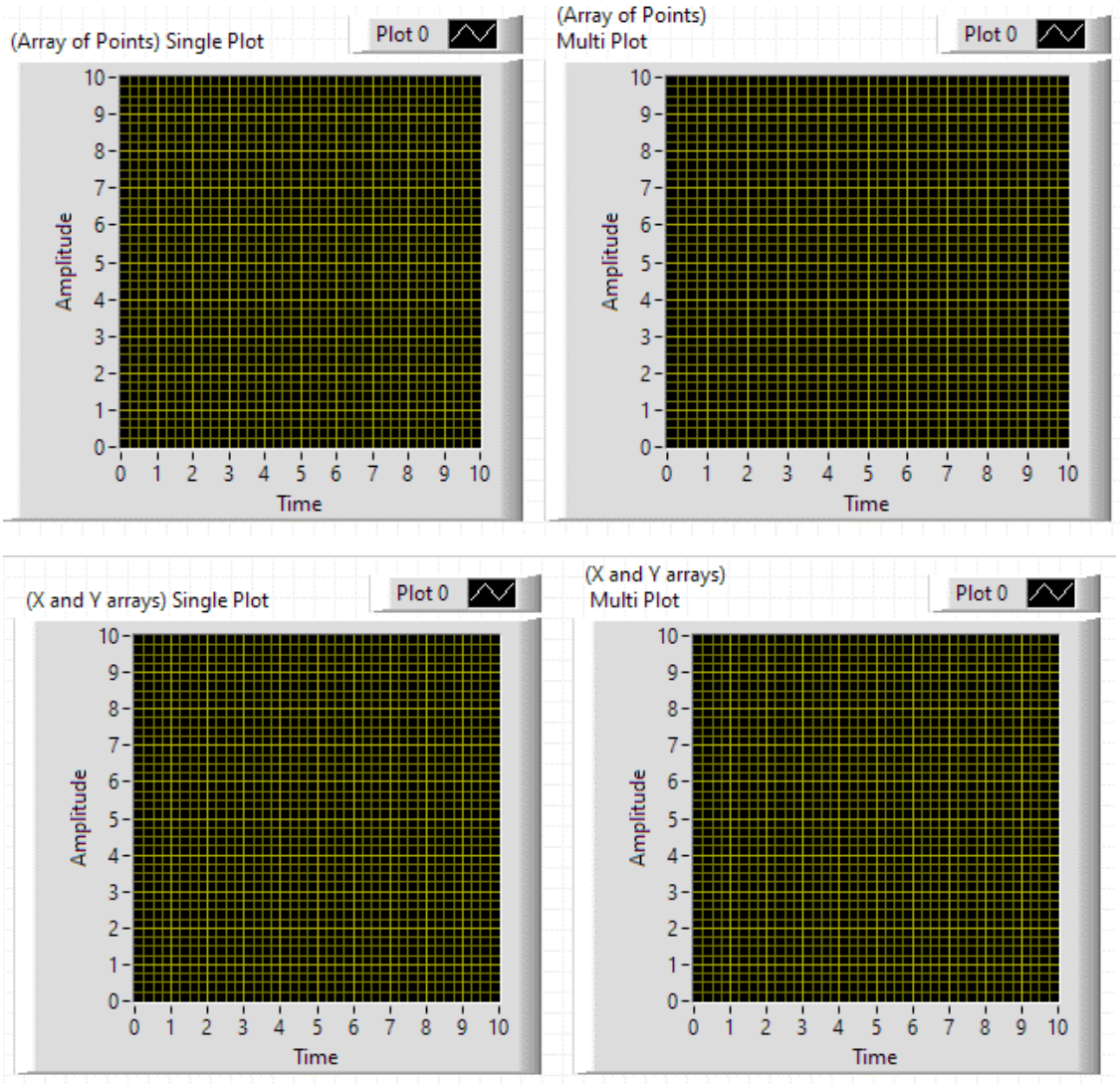
Одиночний двокоординатний графік осцилограм (*X-Y Graph*) працює з групами даних, що містять масиви x і y. Двокоординатний графік осцилограм (*X-Y Graph*) також сприймає масиви точок, де точки − групи даних, що містять значення по шкалах x і y.

Двокоординатні графіки множини осцилограм працюють з масивами осцилограм, в яких осцилограма даних − кластер, що містить масиви значень x і y. Двокоординатні графіки множини осцилограм сприймають також масиви множини осцилограм, де кожна осцилограма є масивом точок. Кожна точка це група даних, що містить значення по x і y.

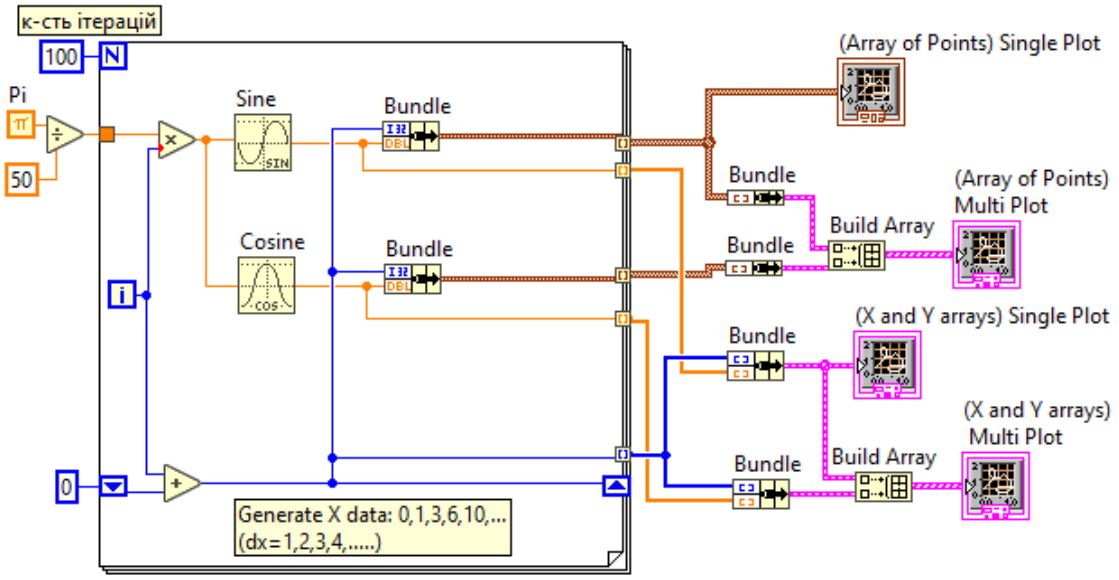
Приклад 3.3. Використання *X-Y Graph*

Розглянемо способи побудови графіків з використанням двокоординатного графіка (*X-Y Graph*), для цього розмістіть на передній панелі нового віртуального приладу чотири вікна *X-Y Graph*.

На панелі діаграм розмістіть цикл *For Loop* у тілі якого розташуйте функції  і  (синус і косинус відповідно), які в циклі сприйматимуть номер ітерації циклу і формуватимуть один елемент. Всього в циклі передбачте 100 ітерацій, у якості вхідних значень циклу використайте дробову частину (наприклад, одну п'ятдесяту від числа π) (рис.3.4).



а)



б)

Рисунок 3.4 — ВП до прикладу 3.3 з використанням *X-Y Graph*:
а) передня панель; б) блок-діаграма

Таким чином, будуть сформовані два масиви чисел, що описують один період синусоїди і косинусоїди, кожен містить сто значень. Окрім цього, в циклі сформуйте масив чисел із змінним кроком. Для цього встановіть на лівій або правій границі циклу курсор миші і викличте контекстне меню, в якому виберіть опцію додавання регістра зсуву *Add Shift Register*.

Підключіть всі елементи у відповідності до рис. 3.4 з використанням елементів *Build Array* і *Bundle*.

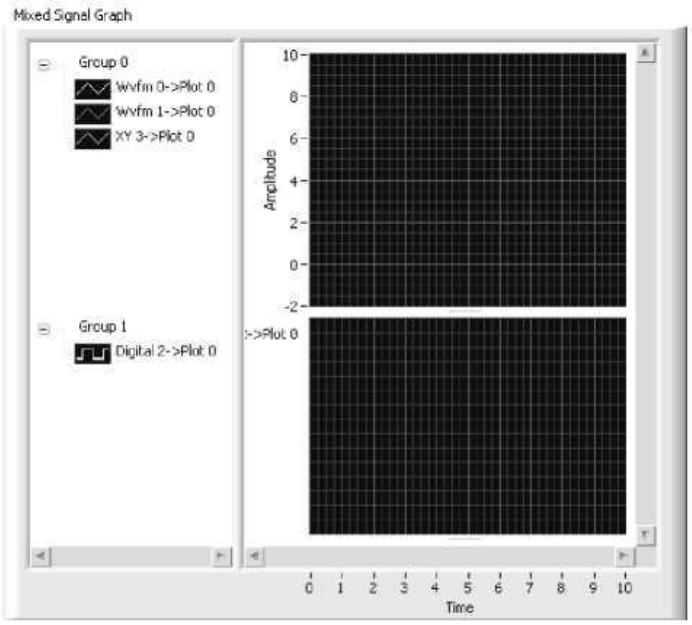
Запустіть розроблений віртуальний прилад на виконання.

Використання *Mixed Signal Graph*

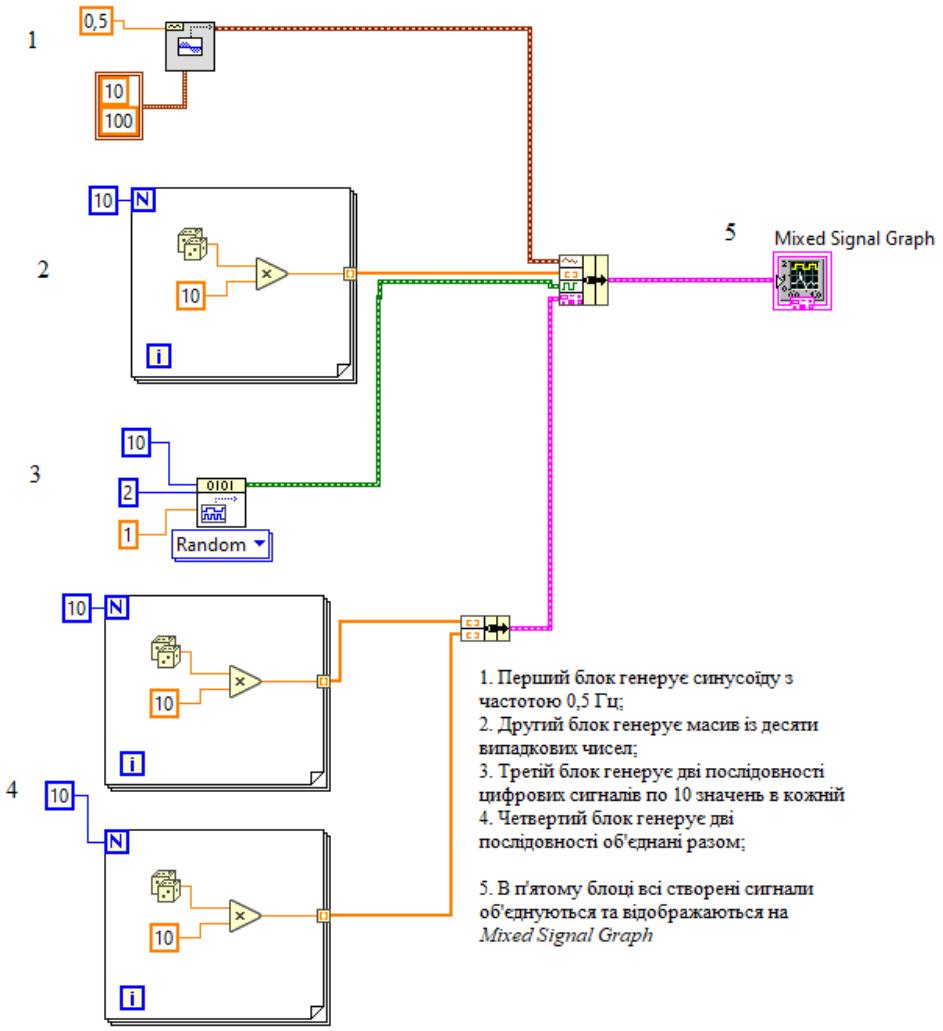
Mixed Signal Graph дозволяє в одному вікні виводити як цифрову, так і аналогову інформацію, організовану багатьма способами.

Приклад 3.4. Використання *Mixed Signal Graph*

Створіть новий віртуальний прилад, розмістіть на його лицьовій панелі *Mixed Signal Graph*, а на панелі діаграм зберіть схему, що складається з елементів представлених на рис.3.5.



а)



б)

Рисунок 3.5 — ВП до прикладу 3.3 з використанням *Mixed Signal Graph*

Окрім вже відомих елементів на схемі використайте додатково такі елементи. У першому блоці елемент формування синусоїдальних сигналів на вхід «*frequency*» якого подано значення 0,5 Гц, а на вхід sampling info подані значення: частота формування 10 відліків в секунду, загальна кількість згенерованих відліків в сигналі 100 (за умовчуванням обидва значення 1000). Цей елемент (*Sine Wfm*) розташований в палітрі *Functions>Signal>Processing>Wfm Generation*. У третьому блоці використайте елемент генерування цифрових сигналів (*Pattern Generator*) розташований в палітрі *Functions>Programming>Waveform>Digital Wfm>Pattern Gen*.

На входах цього елементу встановлена кількість формованих відліків (10), число формованих цифрових сигналів (2) і частота формування сигналу (1, за умовчуванням 1000 Гц). Властивість ramp (*Pattern Generator*) замініть на random (формуватимуться випадкові цифрові сигнали).

Сформуйте і запустіть на виконання запропонований прилад. Переконаєтеся, що на графіках представлені всі 5 сигналів (три аналогових − 1, 2 і 4 блоки, і два цифрові сигнали − 3 блок).

Приклад 3.5. Використання генераторів сигналів

Використання генераторів сигналів при проектуванні віртуальних приладів. Для формування випадкових сигналів використайте розділ контекстного меню до блок діаграми *Programming>Waveform>Analog Waveform* (*Analog Wfm*)*>Generation,* а саме застосуйте базовий функціональний генератор *Basic Function Generator* (*Basic FuncGen*) і вузол для перетворення формату виходу генератора *Get Waveform Components* (*Get Wfm Co*…).

3.5.1 Створіть віртуальний прилад, який містить два формувачі сигналу. Сигнал може бути синусоїдою, трикутної або прямокутної форми та пилкоподібним. Вихід одного з генераторів використайте для розгортки по осі X , а іншого подайте на вісь Y індикатора *XY Graph*. Вихідні сигнали

генераторів також додайте на графічний індикатор *Waveform Graph*, щоб спостерігати їх форму.

Запустіть програму, переконайтеся у її працездатності.

3.5.2 Створіть новий віртуальний прилад. На передній панелі розмістіть два вузли вибору режиму генератора (*Controls>Ring* & *Enum>menu Ring*) і змініть їх мітки відповідно до рис 3.6. Розмістіть також 3 регулятори (*Horizontal Pointer Slide*), мітки яких змініть відповідно до рисунка.

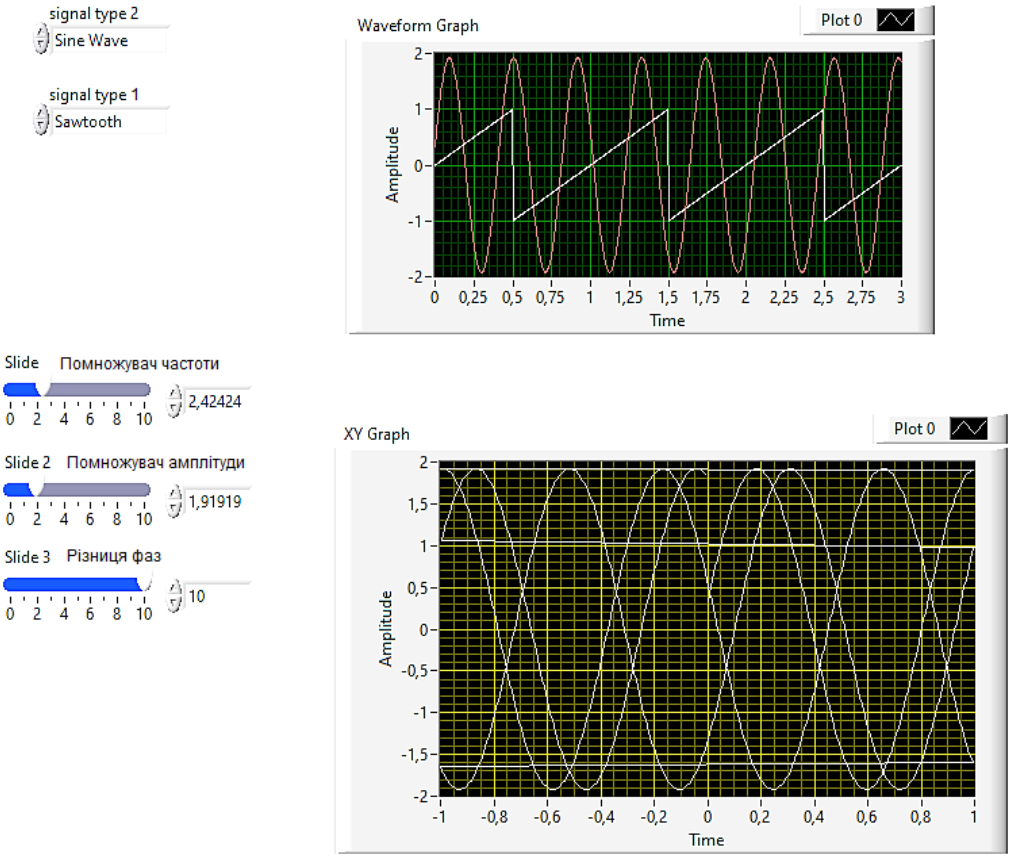
Виведіть числові значення регуляторів (контекстне меню регулятора *>Visible Item>digital Display*). Розмістіть два графічні індикатори *Waveform Graph* (зверху) і *XY Graph* та перейдіть на блок діаграму. На блок діаграмі зручно розмістіть регулятори, додайте 3 константи 3, 1 і 0, розмістіть 2 вузли множення і 1 суматор. Проведіть з'єднання відповідно до рисунка.

Розмістіть на схемі 2 (два) функціональних генератора (*Programming>Waveform >Analog Wfm>Waveform Generation>Basic Function Generator*). Під'єднайте виходи вузлів *Menu Ring* до входів генератора *signal type*. Зліва до генераторів підведіть провідники до контактів (зверху вниз): *frequency; amplitude; phase*.

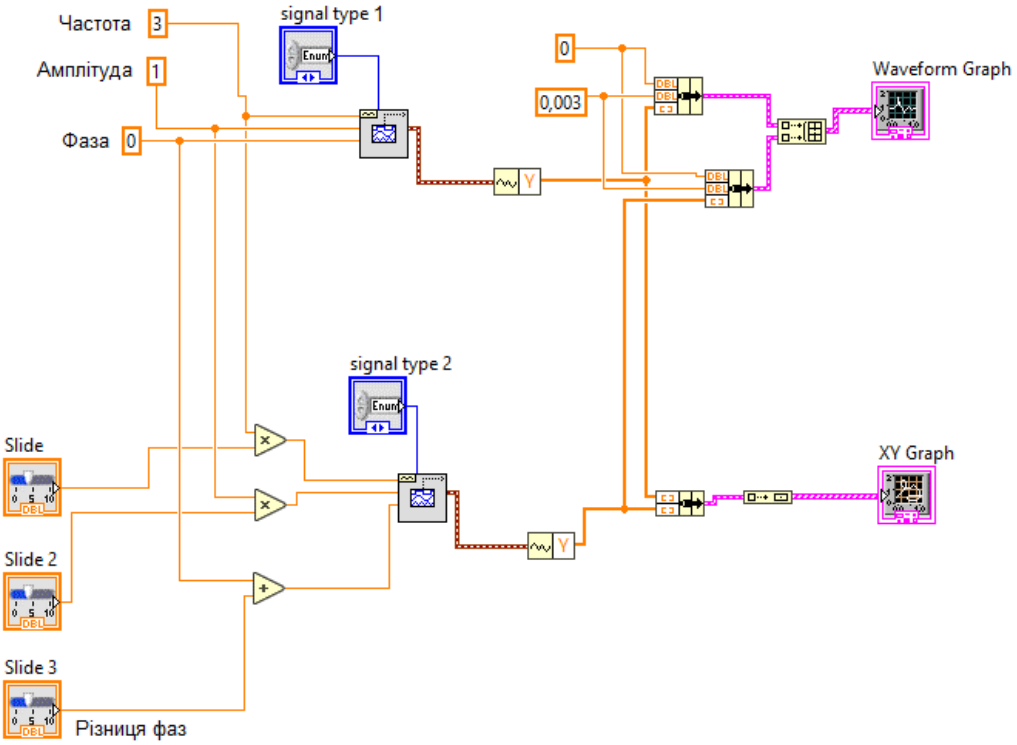
Правіше за генератори сигналів розмістіть два блоки, для збору потрібної інформації (*Programming>Waveform>Get Waveform Components*). Розмістіть три формувачі кластера, два для індикатора *Waveform Graph* і один для *XY Graph* (два верхніх розтягніть на 3 позиції, а нижній на 2).

До нижнього під'єднайте виходи двох вузлів *Get Waveform Components*. Для верхніх, створіть дві константи 0 (початок розгортки) і 0,03 (це крок по осі *X*). З'єднайте нижні входи цих формувачів до виходів вузлів *Get Waveform Components*.

Тепер сформуйте масиви даних для графічних індикаторів. Для цього розмістіть два вузли *Build Array* правіше за формувачі кластерів. Верхній розтягніть на два входи. З'єднайте виходи формувачів кластерів до входів вузлів *Build Array* так, як показано на схемі. Виходи вузлів подайте на графічні індикатори.



а)



б)

Рисунок 3.6 — ВП до прикладу 3.3 з використанням генераторів сигналів: а) передня панель ВП; б) блок-діаграма ВП

Схема готова. Виконайте її налаштування**:**

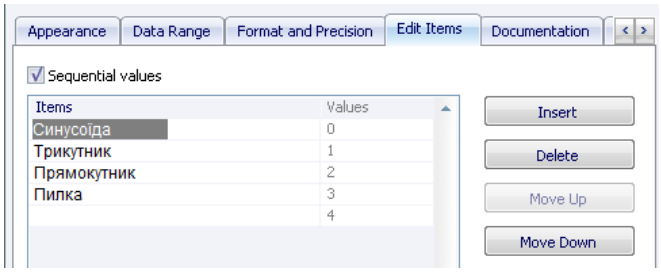


Рисунок 3.7 — Налаштування генератору сигналу

На передній панелі послідовно викличте властивості обох вузлів *Menu Ring* і введіть *Items* відповідно до рис. 3.7. (*Properties>Edit Items>Insert*, Пилка, Прямокутник, Трикутник, Синусоїда). Встановіть межі зміни для регуляторів. Виставте значення на регуляторах, наприклад, такі як на рисунку, і зробіть їх значеннями за замовчуванням. Для цього викличте контекстне меню, послідовно до кожного регулятора, і виберіть команду: *Data Operation>Make Current Value Default.*

Для графічних індикаторів перейменуйте вісі. Викличте властивості для індикатора *Waveform Graph* і на вкладці *Plots* встановіть кольори променів: для одного білий, а для іншого жовтий. На вкладці *Scale* для обох променів відзначте *Autoscale*. Елемент *Plot Legend* (справа вгорі над індикатором) розтягніть на 2 позиції і перейменуйте їх відповідно до рисунка передньої панелі. В другому індикаторі *XY Graph* у властивостях на вкладці *Scale* для обох сигналів встановіть режим *Autoscale*.

Запустіть віртуальний прилад та проаналізуйте його роботу в різних режимах.

Хід роботи

3.1) Виконати приклади для створення віртуальних приладів.

3.2) Отримати і ознайомитися з індивідуальним завданням.

3.3) Сформувати необхідні інструменти.

3.4) Налагодити програму і представити викладачеві для перевірки.

3.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

- звітність про виконання прикладів для створення віртуальних

приладів;

- варіант і зміст індивідуального завдання;
- скріншоти складових програми і результати її виконання;
- відповіді на запитання до лабораторно роботи.

3.6) Представити звіт на захист.

Варіанти завдань

Завдання №3.1

Сконструювати віртуальний прилад відображення синусоїдальної напруги за допомогою генератора синусоїдальних коливань. Використати генератор на панелі блок-схем *Functions>Analyses>Waveform Generation>Sine Waveform*. Для завдання частоти, амплітуди і початкової фази напруги і для спостереження отриманої кривої створити на лицьовій панелі три цифрові джерела (*Controls>Numeric>Digital Control*) і віртуальний осцилограф (*Controls> Graph>Waveform Graph*).

Застосувати термінали генератора, для чого натисніть ПКМ на його іконку і викличте *Visible Items>Terminals*, після чого знову натисніть ПКМ на іконці генератора і викличте *Help*, щоб визначити точки підключення джерел. Забезпечити можливість завдання значень частоти, амплітуди і фази напруги (у градусах).

Встановити відображення двох періодів коливань. Задаючи різні амплітуди, фази і частоти, побудувати декілька осцилограм синусоїдальної напруги.

Завдання №3.2

Зібрати прилад для моделювання двох синусоїдальних величин – напруги і струму із застосуванням синусоїдальних функцій і циклу *For Loop* (див. рис. 3.8). В тіло циклу внести іконки синусоїдальних функцій, одна з яких відображує напругу, а інша струм (*Functions*/*Numeric*/*Trigonometric*/*Sine*).

На вхід подати значення кутів в радіанах. Кількість обчислень за один період N=360. Поточний параметр циклу i відповідає кількості градусів і змінюється від 0 до 360.

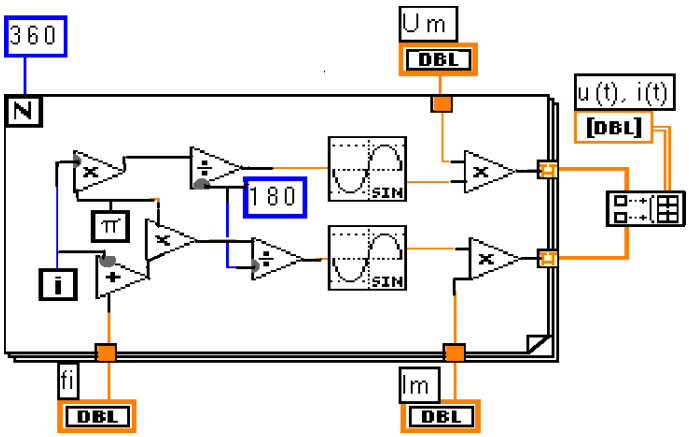


Рисунок 3.8 — Приклад блок-діаграми до ВП з завдання 3.2

Початкова фаза задається збільшенням потрібної кількості градусів до параметра i відповідної синусоїди. Значення вихідних величин синусоїдальних функцій зазнають множення відповідно на амплітудні значення напруги і струму та спостерігаються на віртуальному осцилографі.

Завдання №3.3

Створити віртуальний прилад − двопроменевий осцилограф, за допомогою *Waveform Graph*. Застосувати *Array Tools>Build Array* для об'єднання двох сигналів на вході осцилографа. Побудувати криві струму і напруги при різних фазових зсувах (позитивних і негативних). Двопроменевий осцилограф повинен мати загальну шкалу ординат, амплітудні значення вимірюваних величин зробити сумірними по абсолютній величині.

Завдання №3.4

Зібрати прилад для виміру напруги, струму і потужності із застосуванням формульного вузла і циклу *For Loop* (див. рис. 3.9). У формульний вузол (*Functions>Structures>Formula Node*) вписати формулу $u = U_m \sin \omega t$, де аргумент ωt в радіанах записується як $\omega t = k\pi/180$. Параметр циклу позначити буквою k, щоб відрізнити його від позначення струму.

Провести розрахунок впродовж двох періодів. Отримані значення напруги, струму і миттєвої потужності вивести на осцилографи *Waveform Graph*. Діючі значення синусоїдальної напруги і струму виміряти віртуальними приладами *RMS* (*Root Mean Square*), *Functions>Mathematics>Probability and Statistics>RMS*, до виходів яких підключити цифрові індикатори.

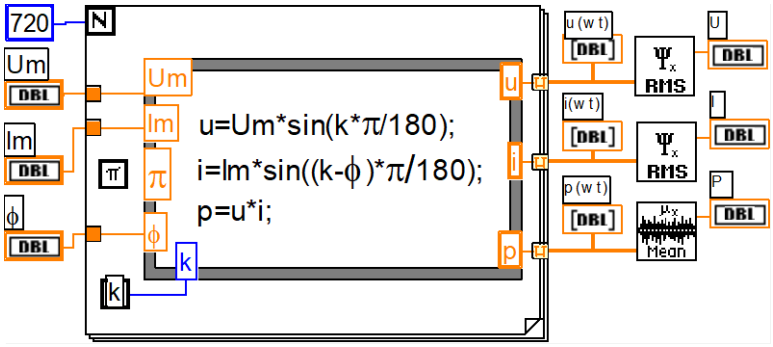


Рисунок 3.9 — Приклад блок-діаграми до ВП з завдання 3.4

Активну потужність виміряти віртуальним приладом *Mean*, що викликається аналогічно *Functions>Mathematics>Probability and Statistics/Mean*. Використати індикатори з іменами *I, U* і *P* для виведення розрахункових даних. Побудувати криві напруги, струму і миттєвої потужності при різних зсувах фаз між напругою і струмом.

Завдання №3.5

Створити віртуальний прилад, що генерує послідовність випадкових чисел та відображує на діаграмі: поточне число, середньоарифметичне значення, обчислене з використанням трьох останніх значень, і середньоквадратичне значення, обчислене з використанням трьох останніх значень. За допомогою кнопок управляти видимістю кожної кривої на графіці.

На лицьовій панелі ВП використати елемент *Horizontal Pointer Slide* (*Controls>Numeric*) з діапазоном значень від 100 до 1000 і видимим цифровим дисплеєм. Також використати *Waveform Chart* (*Controls - Graph*), три елементи *Push Button* (*Controls - Boolean*) і елемент *Stop Button* (*Controls - Boolean*).

На блок-діаграмі ВП (рис. 3.10) застосувати цикл *While*, термінал умови завершення якого з'єднати з кнопкою stop; елемент затримки *Wait Until Next ms Multiple*; регістр зсуву з трьома вхідними терміналами і генератор випадкових чисел; блок *Bundle* (*Functions - Clusters*); три структури *Case*; шість вузлів властивостей (*Property Node*) для графіка, використовуючи його контекстне меню (*Create - Property Node*, у вузлах властивості *Activeplot* і *Plot – Visible* перевести в режим запису), помістити кожен вузол властивостей в блоки *True* і *False* структур *Case*.

Створити по константі для кожної з властивостей, встановити в них необхідні значення і записати їх у властивості. Для вибору необхідної кривої на графіці використовувати індексацію 0, 1,2 ..., для установки видимості використовувати булеві значення.

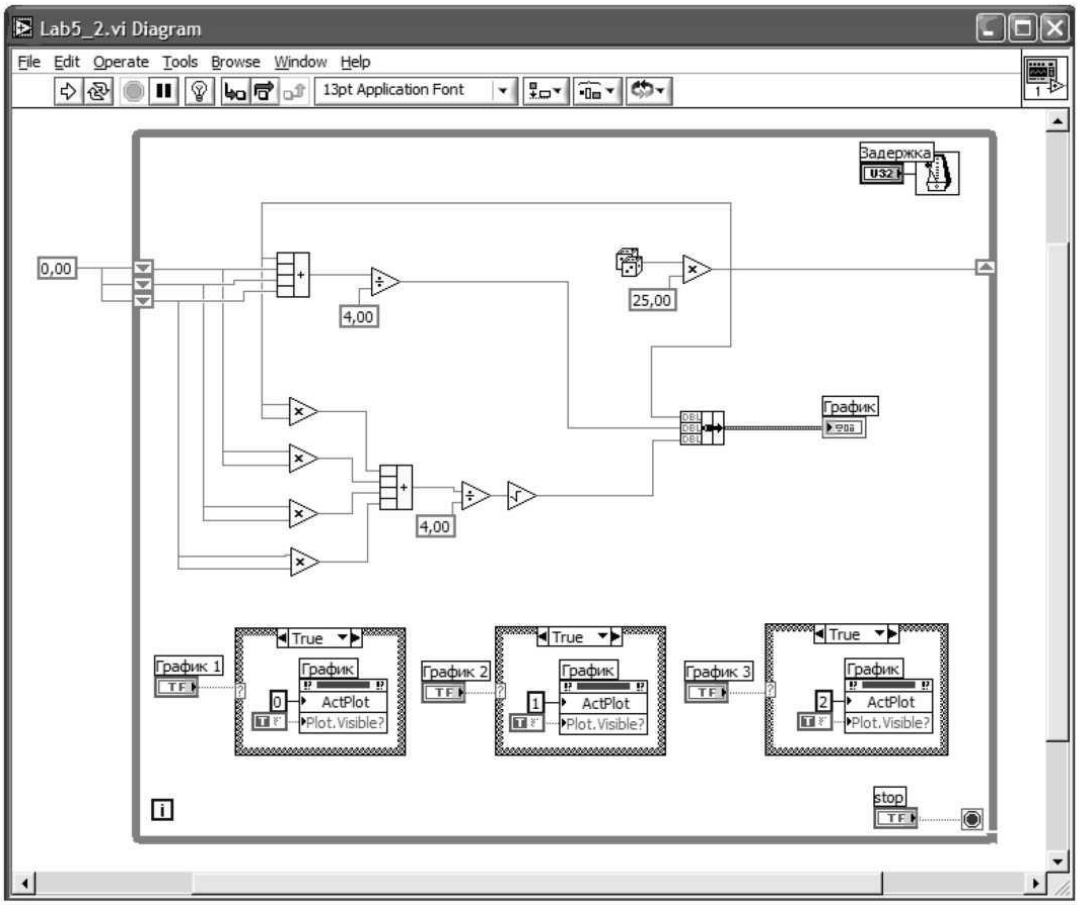


Рисунок 3.10 — Приклад блок-діаграми до ВП з завдання 3.5

За допомогою арифметичних операцій розрахувати середньоарифметичне і середньоквадратичне значення регістра, використовуючи три його останніх значення.

Контрольні запитання

3.1) Які засоби відображення графічної інформації є в *LabVIEW*?

3.2) Чим відрізняється *Waveform Chart* від *Waveform Graph*?

3.3) Особливості використанні *X-Y Graph*?

3.4) Для чого використовується *Mixed Signal Graph*?

3.5) Як використовуючи контекстні меню графіків можна змінювати форми представлення графічної інформації, колір, товщину ліній та ін.?

3.6) Який пункт в контекстному меню дозволяє здійснювати очищення вікна відображення інформації?

3.7) Якими способами можна моделювати синусоїди в середовищі *LabVIEW*?

3.8) Як викликається і підключається віртуальний осцилограф?

3.9) Як побудувати двопроменевий віртуальний осцилограф?

3.10) Як по осцилограмі визначається зсув фаз φ між напругою і струмом?

3.11) Як змінюється вигляд осцилограми миттєвої потужності із зміною кута?

3.12) Що таке діюче значення змінного струму та як його виміряти за допомогою віртуальних інструментів *LabVIEW*?

3.13) Що таке активна потужність та як її виміряти за допомогою віртуальних інструментів *LabVIEW*?

ЛАБОРАТОРНА РОБОТА № 4
МАТЕМАТИЧНІ РОЗРАХУНКИ У LABVIEW.
РОБОТА З МАСИВАМИ

Мета роботи: Навчитися працювати з масивами в *LabVIEW*.

Стислі теоретичні відомості

Масив (*array*) – це неперервний, пронумерований, необмежений набір однотипних даних. Кожен елемент масиву має набір індексів, відповідний розмірності масиву: одновимірний – 1 індекс, двовимірний, – 2 індекси і так далі.

Графічно масив виглядає як прямокутна область, через яку можна переглядати елементи масиву. Поряд з лівим верхнім кутом цієї області відображуються індекси. Значення цих індексів відповідають елементу масиву, показаному в лівому верхньому кутку. Одновимірний масив (вектор) – рядок або стовпець.

Двовимірний масив (матриця, таблиця) – таблиця з декількох рядків і декількох стовпців. Масиви великої розмірності на плоскості екрану монітора відображувати неможливо, тому вони виглядають як таблиці, що є зрізом по певних індексах. Масив може містити дані довільного типу. Наприклад, може бути масив тумблерів (дискретні регулятори), масив цілих чисел та масив кластерів.

Для зміни розмірності масиву можна використовувати спливаюче меню властивостей індексів масиву. У *LabVIEW* елементи масиву нумеруються по рядках від нуля. Таким чином елемент двовимірного масиву з індексами ([2; 4]) знаходиться в третьому рядку і п'ятому стовпці. Масив завжди неперервний набір даних, тобто без пропусків.

Всі елементи масиву мають однин і той же тип даних, причому в широкому сенсі. Це означає, що однакові типи даних, графічне відображення, кольори, розміри графічних образів кожного елементу. У палітрі функцій є всі необхідні засоби для роботи з масивами.

У *LabVIEW* реалізовано досить багато різних функцій для роботи з масивами у вікні редагування діаграм. Вони знаходяться на функціональній панелі (*Functions*) в розділі масивів (*Array*). Розглянемо деякі з них, найбільш важливі функції.

Array Size — повертає розмір масиву. Якщо вхідний масив *N*-мірний, то вихідний параметр (*size*) це одновимірний масив з *N* елементів, які вказують його розмірність. Наприклад, якщо вхідний 2*d* масив має розмір 4x6, то на виході буде сформований одновимірний масив з двох чисел, відповідно 4 і 6.

Initialize Array — створює масив розмірності dimension *size*, всі елементи якого набувають значення *element*. Для створення багатовимірного масиву необхідно "розтягнути" іконку, "потягнувши" за правий нижній кут вниз.

Build Array — об'єднує масиви або додає елементи до вже існуючого масиву. Можна змінити розмір піктограми цієї функції для збільшення кількості входів.

Array Subset — "вирізає" підмасив згідно із заданими значеннями стартового індексу (*index*) і довжини (*length*).

Index Array — виділення елементу масиву (доступ до елементу масиву). Використовуючи цю функцію можна виділяти не лише елементи масиву, але і бажані рядки та стовпці масивів.

У *LabVIEW* для роботи з масивами реалізовані такі арифметичні функції як складання, множення, ділення та інші. Ці функції поліморфні, тому вхідні дані можуть бути різного типу: скалярними або масивами. Наприклад, можна підсумовувати скаляр з масивом або скласти два масиви разом.

Для створення масиву елементів управління і індикації даних необхідно вибрати шаблон масиву з палітри *Controls>Array* & *Claster* і помістити його на лицьову панель (рис. 4.1).

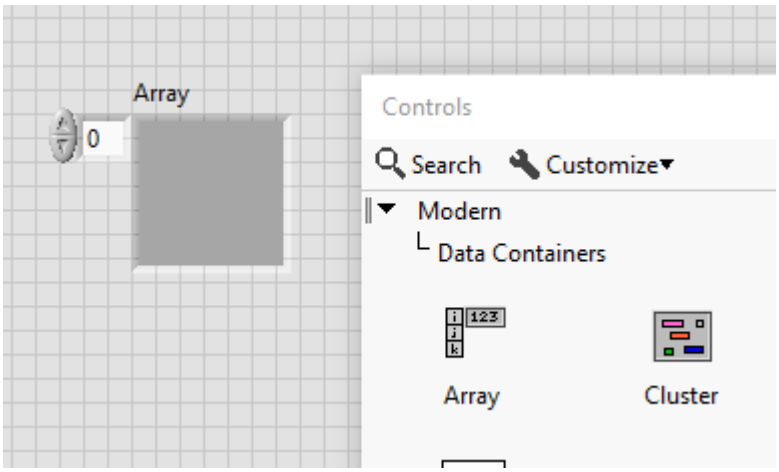
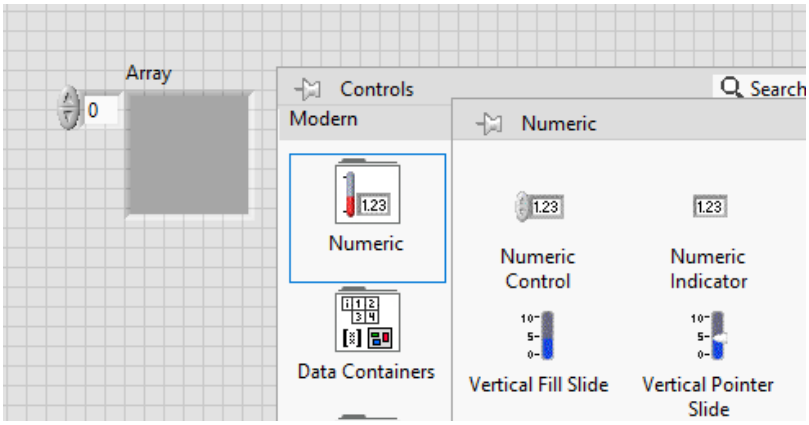
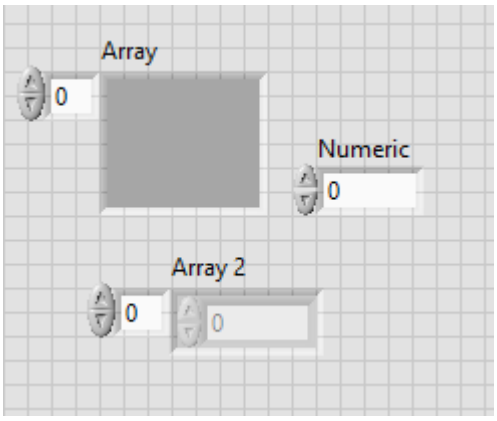


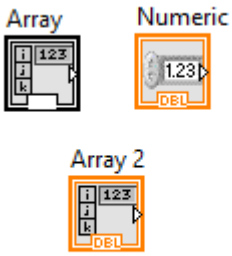
Рисунок 4.1 — Створення масиву елементів з палітри *Controls>Array* & *Claster*

Потім в шаблон помістити елемент управління або індикації (рис 4.2).



а)





б) в)

Рисунок 4.2 — Створення масиву елементів управління і індикації:
а) вибір елементу управління чи індикатору; б) передня панель масивів;
в) вигляд на блок-діаграмі

Відразу після створення масиву видно лише один елемент. Для того, щоб побачити декілька елементів необхідно розтягнути шаблон. Аналогічним чином створюється масив констант на блок – діаграмі.

Автоіндексація масивів. Цикли з фіксованим числом ітерацій можуть автоматично індексувати масиви на їх границі шляхом додавання одного нового елементу в кожному повторенні циклу.

Для створення двовимірних масивів необхідно використовувати два цикли *For*, один усередині іншого (рис. 4.3). Зовнішній цикл створює елементи масиву в рядку, а внутрішній цикл створює елементи масиву в стовпці.

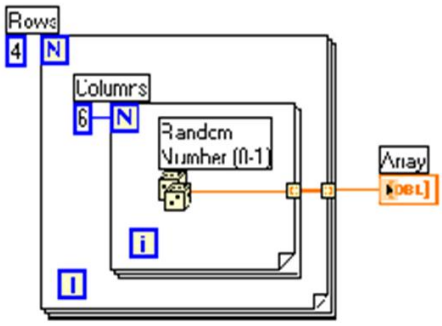
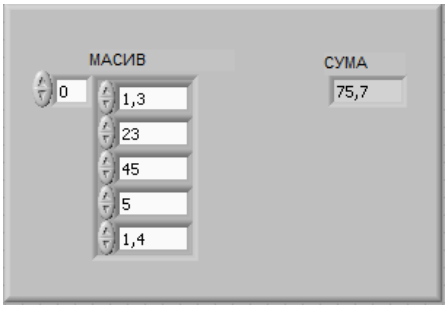


Рисунок 4.3 — Блок-діаграма створення масиву

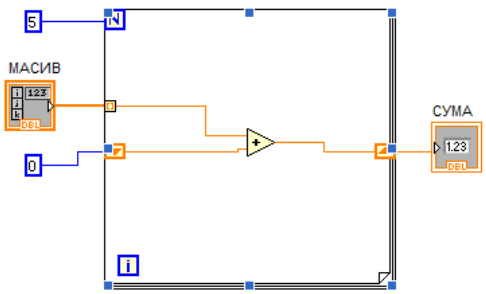
Регістри зсуву (*shift registers*) застосовуються спільно із структурою цикл і є особливим типом змінної, використовуваної для передачі величини з однієї ітерації циклу в наступну. Регістр зсуву створюється натисканням правої кнопки миші по правій або лівій границі циклу (*Add shift register*).

Регістр зсуву складається з пари терміналів, розташованих напроти один одного. У правому терміналі зберігаються дані, отримані при завершенні ітерації. Ці дані зміщуються в кінці ітерації і з'являються в лівому терміналі на початку наступної ітерації.

Зразок розрахунку суми елементів масиву показано на рис.4.4.



а)



б)

Рисунок 4.4 — ВП розрахунку суми елементів масиву: а) передня панель; б) блок-діаграма

Зразок знаходження максимального елементу масиву показано на рисунку 4.5.

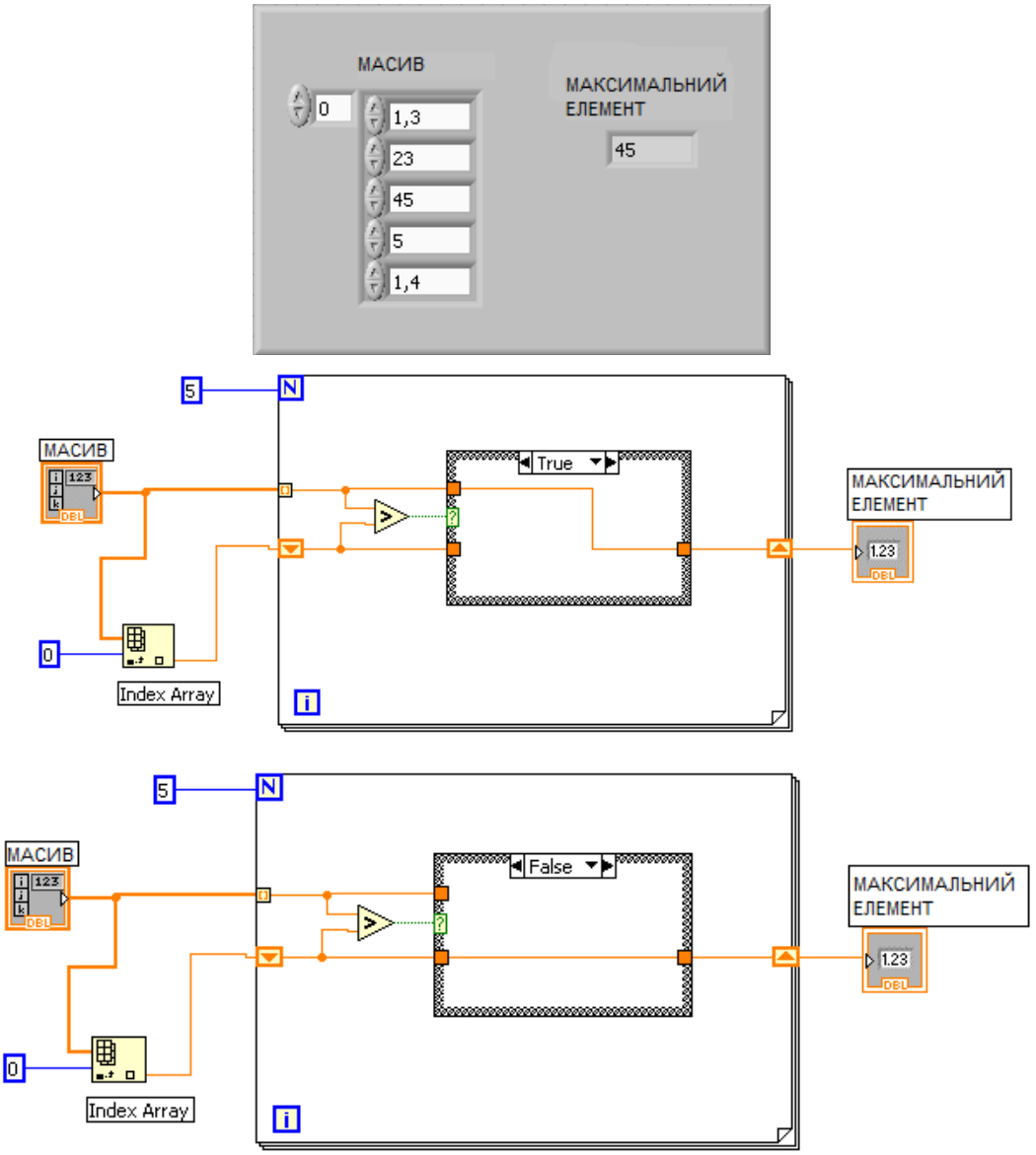


Рисунок 4.5 — Приклад ВП знаходження максимального елементу масиву: а) передня панель; б) блок-діаграма

На рисунках 4.6 та 4.7 наведені блок-діаграми прикладів застосування функції *Array Max & Min* та функції *Add Array Elements*, відповідно.

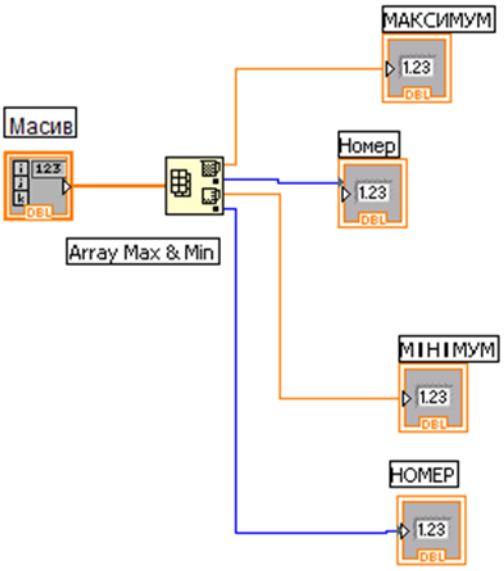


Рисунок 4.6 — Блок-діаграма ВП використання функції *Array Max & Min*

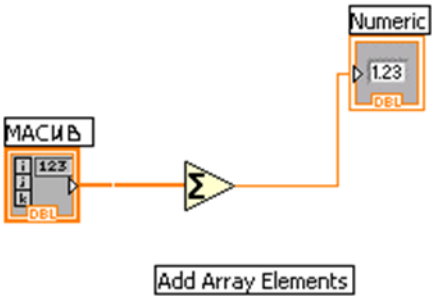
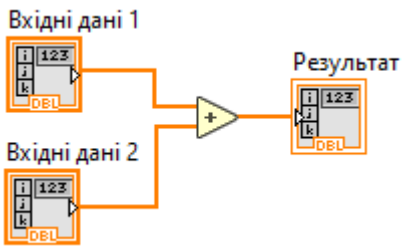


Рисунок 4.7 — Блок-діаграма ВП використання функції *Add Array Elements*

Зразок додавання двох одновимірних масивів представлено на рис.4.8.

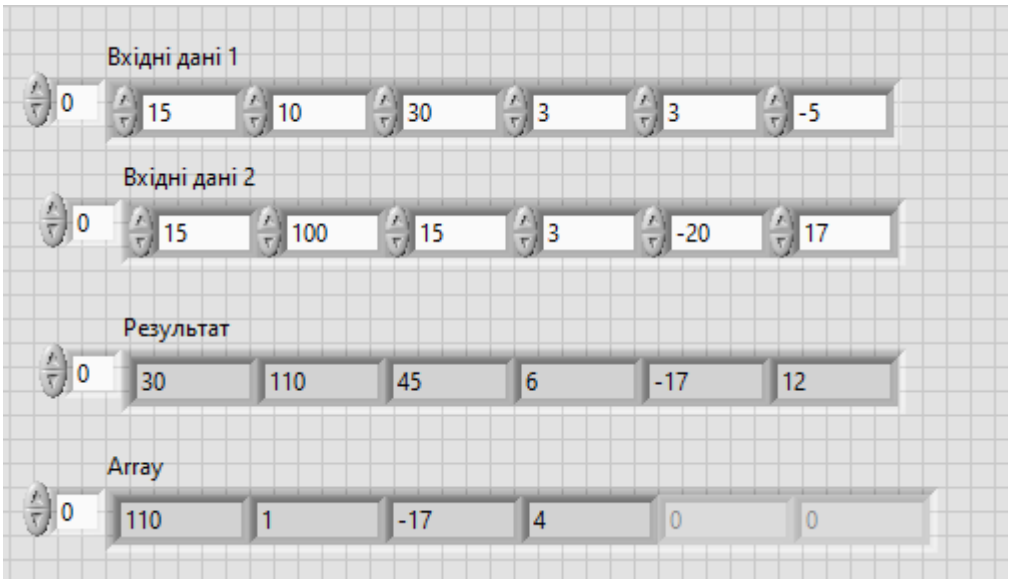
а)



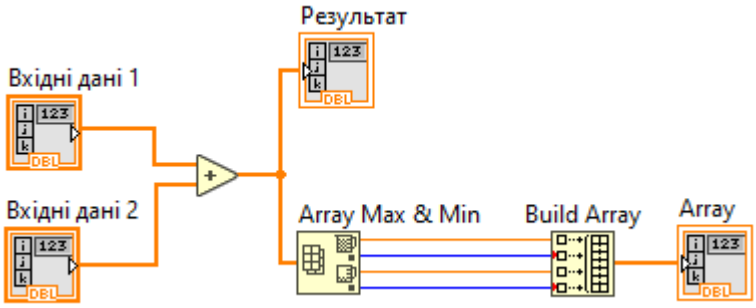
б)

Рисунок 4.8 — Додавання двох одновимірних масивів:
а) передня панель; б) блок-діаграма

Зразок додавання двох одновимірних масивів з використанням функції *Build Array* (рис. 4.9). Використовується функція *Array Max & Min*, де виводиться значення максимального елементу та його порядковий номер, а потім значення мінімального елементу та його порядковий номер, тому бачимо в таблиці *Array* лише чотири числа.



а)



б)

Рисунок 4.9 — Додавання двох одновимірних масивів з використанням функції *Build Array*: а) передня панель; б) блок-діаграма

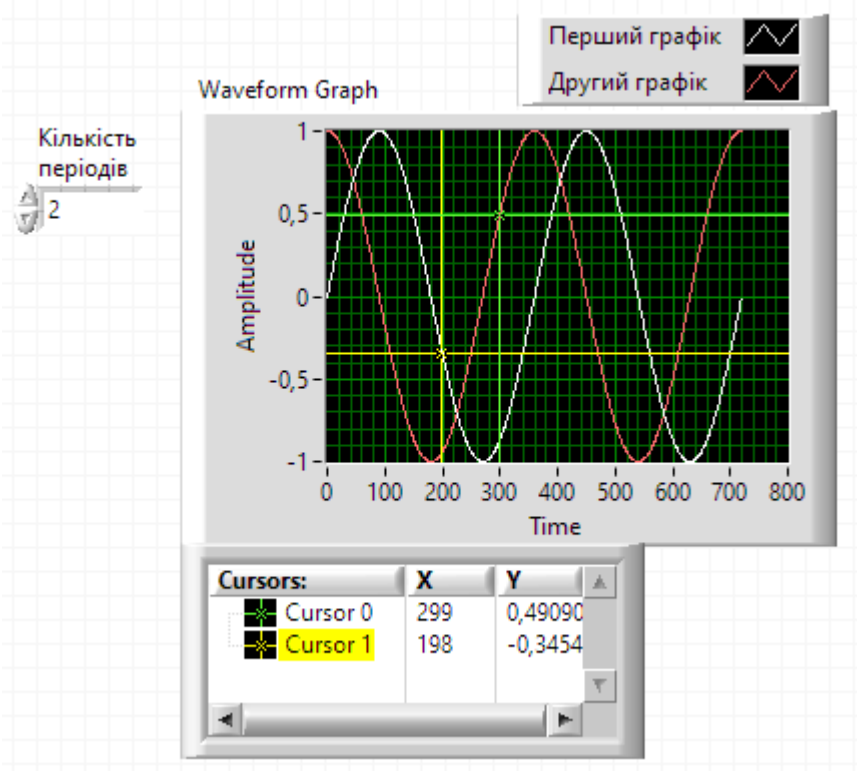
Пам’ятаємо, що по замовчуванню, нумерація починається з нульового елементу.

Приклад 4.1

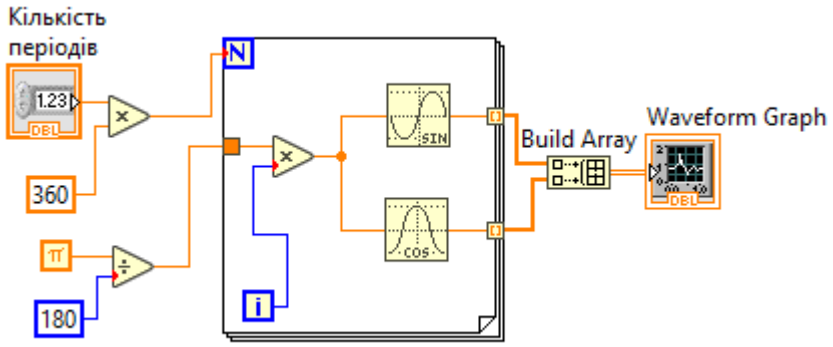
Побудуйте віртуальний прилад, який показуватиме на екрані дві функції *sin* і *cos*. Прилад повинен дозволяти змінювати число періодів, що відображуються на екрані, і забезпечувати вимірювання значення цих функцій. Для цього використайте графічний індикатор *Waveform Graph*, засіб для формування циклу *For Loop* і засіб для об'єднання масивів даних в кластер *Build Array*.

Створіть новий віртуальний прилад і помістіть індикатор на передню панель (*Control* > *Graph* > *Waveform Graph*). Додайте до нього контекстне меню. В розділі *Visible Item* меню використайте пункти: *Label*, *Plot Legend*,

Cursor Legend, *X – Scale*, *Y – Scale*. Відзначте ці пункти галочкою. Розмістіть елементи на передній панелі так, як показано на рис. 4.10.



а)



б)

Рисунок 4.10 — До прикладу 4.1: а) передня панель; б) блок-діаграма

Налаштування вузла. У контекстному меню до графічного індикатора викличте останній пункт *Properties*. Скоректуйте вміст вкладок.

• Вкладка ***Appearance***. У рамці зліва вгорі введіть «Екран». У розділі *Plot Shown* введіть 2 (на екрані дві кривих).

• Вкладка ***Format and Precision*** в рамці зліва вгорі: Ордината (*X-axis*), Значення функції (Y-axis); параметр digit 6.

• Вкладка ***Plots*** в рамці зліва вгорі

- для графіка 0 *Name*: Перший графік 0. *Color* – білий

- Перейдіть у верхній рамці на 1 і введіть в рамку *Name*: Другий графік 1. *Color* – червоний.

• Вкладка ***Scales*** в рамці зліва вгорі: (*X-axis*), введіть в рамку *Name* слово Ордината (*Y-axis*) у рамку *Name* слово Значення функції. Відзначте параметр *Autoscale*.

• Вкладка ***Cursors.*** Параметр *Show cursor* має бути відмічений

- Для параметра *Cursor* 0, розміщеного в лівому верхньому кутку рамки, відредагуйте розділ *Allow Dragging*: перша рамка *Single-plot*, друга рамка Перший графік 0. Параметр *Cursor color* − колір білий.
- Для параметра *Cursor* 1, розміщеного в лівому верхньому кутку рамки відредагуйте розділ *Allow Dragging*: перша рамка *Single-plot*, друга рамка Другий графік 1. Параметр *Cursor color* − колір червоний.

Після налаштування завантажте в блок діаграму цикл *For Loop* (*Function* > *Structure* > *For Loop*). Вихід вузла множення подайте на вхід *N* циклу. Нижня частина формуватиме значення в 1 градус, виражене в радіанах. Вихід дільника бажано ще не з'єднувати.

Розмістіть в тілі циклу вузли розрахунку синуса і косинуса (*Function* > *Mathematics* > *Elementary* > *Trigonometric* > …), перед ними поставте вузол множення і його вихід з'єднайте до входів цих вузлів. На входи вузла множення подайте: ***i*** – номер реалізації і вихід дільника, який розташований за межами циклу.

Завдяки цьому отримаєте розрахунок функцій *sin* і *cos* через 1 градус протягом числа періодів, визначених регулятором «Число періодів».

Щоб подати вихідні дані на вхід графічного індикатора, їх потрібно об'єднати в єдиний потік, що можна здійснити, сформувавши масив.

Розмірність масиву визначається числом відображуваних кривих, а число елементів в рядку масиву – числом точок на одному графіку. Для побудови масиву на блок діаграмі скористайтеся функцією *Build Array* (*Function* > *Array* > *Build Array*). По цій команді на екран буде виведено

зображення для одновимірного масиву. Виберіть цей елемент і розтягніть його так, щоб у нього з'явилися 2 входи.

Верхній вхід відповідає стовпцю 0 (перша крива на індикаторі), а нижній – стовпцю 1 (друга крива на індикаторі). Подайте на верхній вхід вихід блоку sin, який розташований усередині циклу, а на нижній – вихід блоку cos, який розташовується там же. Вихід масиву подайте на вхід графічного індикатора Екран.

Включіть проміжний віртуальний прилад і перевірте його роботу (рекомендовано зберегти віртуальний прилад).

Отримайте таблицю значень функцій синус і косинус, для цього скористайтеся масивами. На рис. 4.11 представлений модернізований прилад, який не лише виводить графік функцій, але і їх значення.

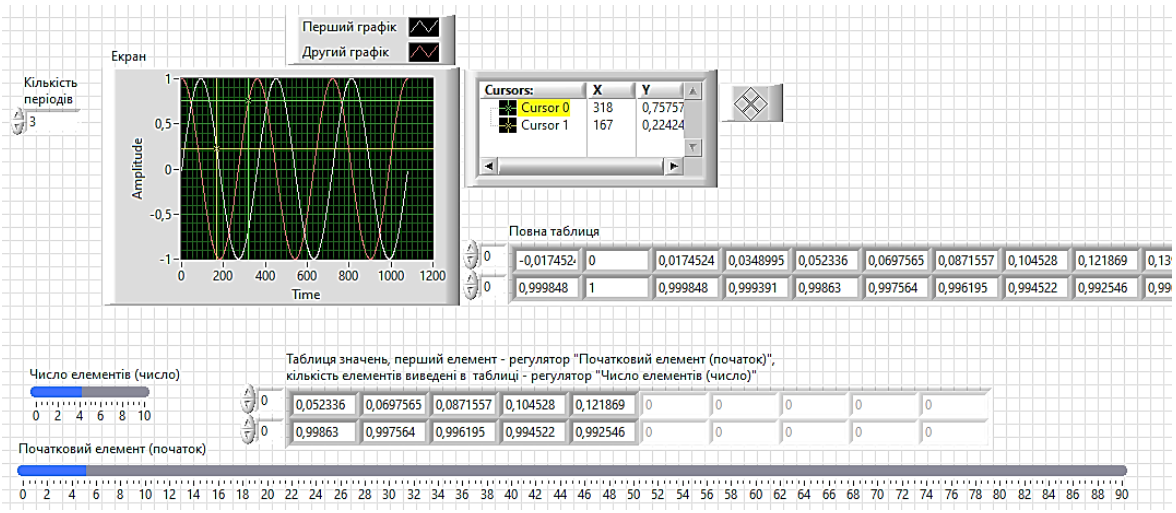
Створіть два масиви, один з міткою (*lable*) «Повна таблиця» для виведення всіх значень функцій через 1 градус, а інший «Таблиця значень» від «початок», число елементів «число» для виводу від 0 до 10 значень функцій в інтервалі від 0 до 90 градусів через градус, причому початкове значення вибирається за допомогою регулятора.

Для цього викличте контекстне меню до лицьової панелі і за допомогою команди: *Control>Array, Matrix > Array* розмістіть дві заготівки масиву на передній панелі. Для кожного з них встановіть числовий тип даних, завантаживши в обидва масиви число (*Control > Numeric > Numeric Control*). За допомогою контекстного меню визначте для цих чисел формат *SGL* (*Properties > Data Range > Representation > SGL*).

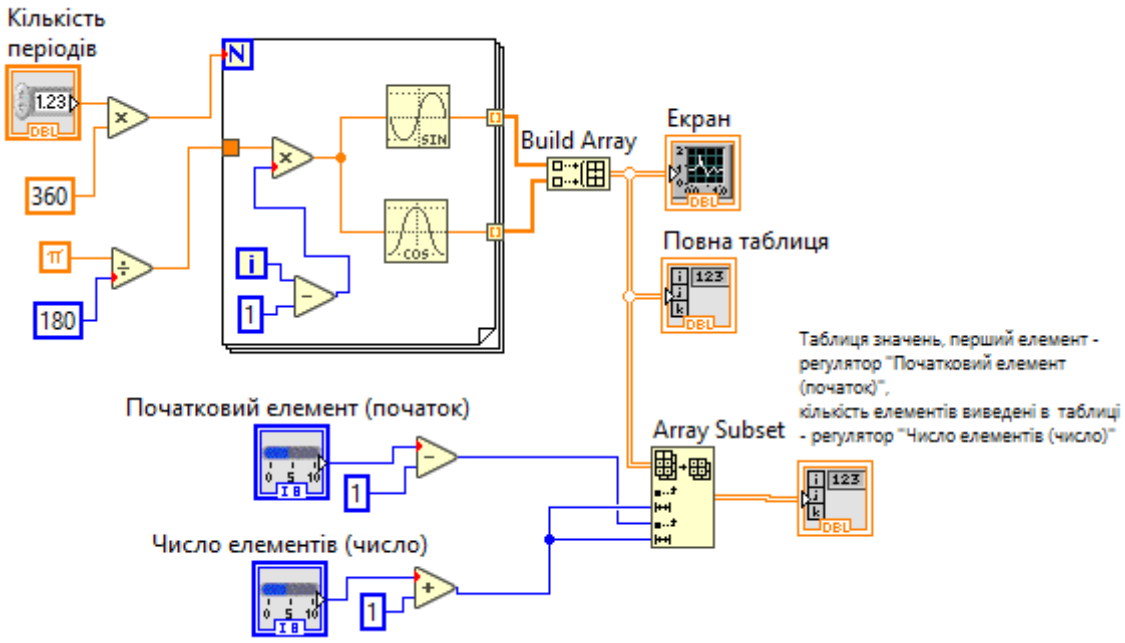
Розмістіть ці масиви так само, як показано на рис. 4.11, і зробіть видимими їх мітки (контекстне меню до масиву > *Visible Item > Lable*). Додайте два регулятори з мітками «Початковий елемент (початок)» і «Число елементів (число)».

Розмістіть їх на передній панелі (див. рис. 4.11). Встановіть видимість значень регуляторів (контекстне меню до регулятора > *Visible Item > Digital*

Display). Нижче за другий масив введіть текст «Кут першого елементу в градусах» і перетягніть числове значення першого індикатора під цей напис.



а)



б)

Рисунок 4.11 — Модернізований ВП до прикладу 4.1, який не лише виводить графік функцій, але і їх значення:
а) передня панель; б) блок-діаграма

Перейдіть до блок-діаграми. Розмістіть масив «Повна таблиця» під графічним індикатором і з'єднайте його вхід. Нижче розташуйте вузол для

вибору даних з масиву (*Control* > *Array, Matrix* > *Array Subset*). З'єднайте його вхід «*array*» до шини даних (його значок повинен змінитися).

Розмістіть регулятори «Початковий елемент (початок)» і «Число елементів (число)». Від значення першого з них відніміть, а до значення другого додайте 1, і з'єднайте результат з входами елементу *Array Subset.* Щоб почати виведення елементів з нульового значення в циклі потрібно зменшити поточний номер на 1.

Перевірте правильність роботи блоку, наприклад, використовуючи значення для регуляторів, що приведені на рис. 4.11.

Приклад 4.2

Створити віртуальний прилад, який створюватиме двовимірний масив (три рядки і сто стовпців) випадкових чисел і записуватиме транспоновані дані у файл таблиці символів. Файл повинен містити заголовки для кожного стовпця. Потім відображувати три результуючих вектора на одному графіку.

Для цього створіть новий ВП. На лицьовій частині ВП (рис. 4.12) помістіть елемент управління *File Path Control* (*Controls* > *String & Path*), графік *Waveform Graph* (*Controls* > *Graph*) та кнопку. На блок-діаграмі ВП створіть нескінчений цикл за умови. Всередині циклу створіть структуру *Case* і з'єднаєте термінал кнопки з терміналом умови структури.

Всередині структури *Case* створіть структуру *Sequence* з трьома кадрами. У першому кадрі структури *Sequence* (рис. 14.13) помістіть елементи *Write to Text File* (*Functions - File* I/O), *Concatenate Strings* (*Functions - String*), створіть рядок заголовків шляхом склеювання підрядків і строкових констант символу табуляції і кінця рядка. Запишіть результуючий рядок у файл, задавши як ім'я файлу значення, вибране користувачем в елементі управління *Path.*

Створіть локальну змінну усередині кадру (з контекстного меню за натисканням правою кнопкою миші на рамці структури) і запишіть в неї значення імені файлу.

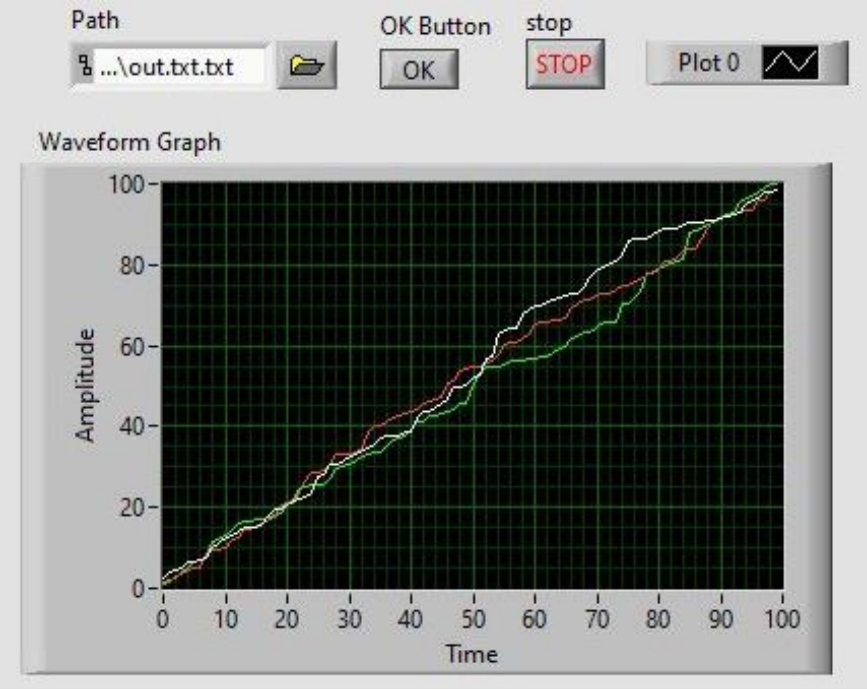


Рисунок 4.12 — Передня панель ВП до прикладу 4.2

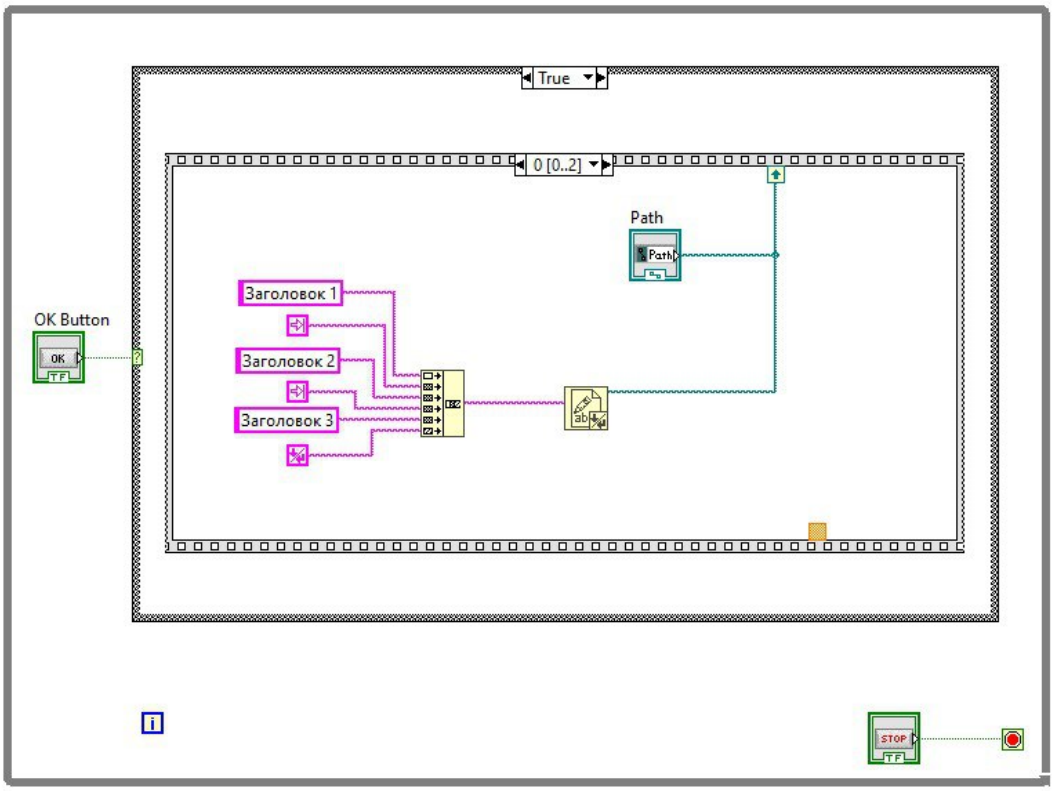


Рисунок 4.13 — Блок-діаграма першого кадру структури *Sequence* до прикладу 4.2

У другому кадрі структури *Sequence* (рис. 4.14) розмістіть два цикли *For Loop* і згенеруйте з їх допомогою двовимірний масив, що складається з трьох рядків по сто стовпців.

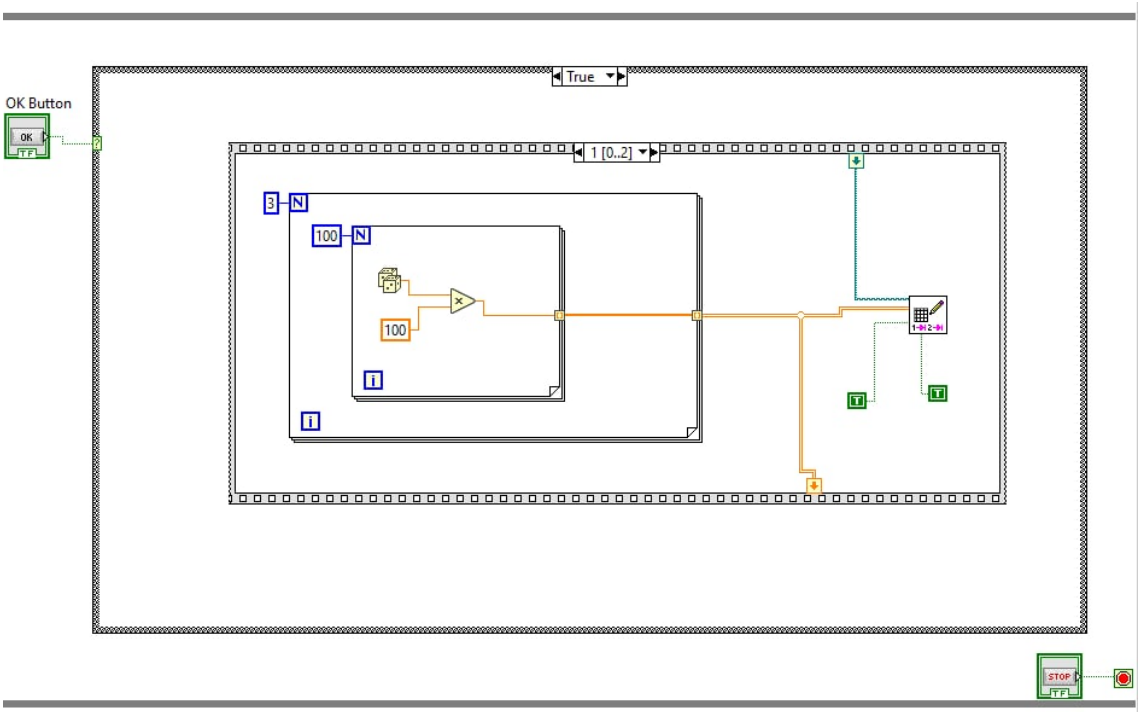


Рисунок 4.14 — Блок-діаграма другого кадру структури *Sequence* до прикладу 4.2

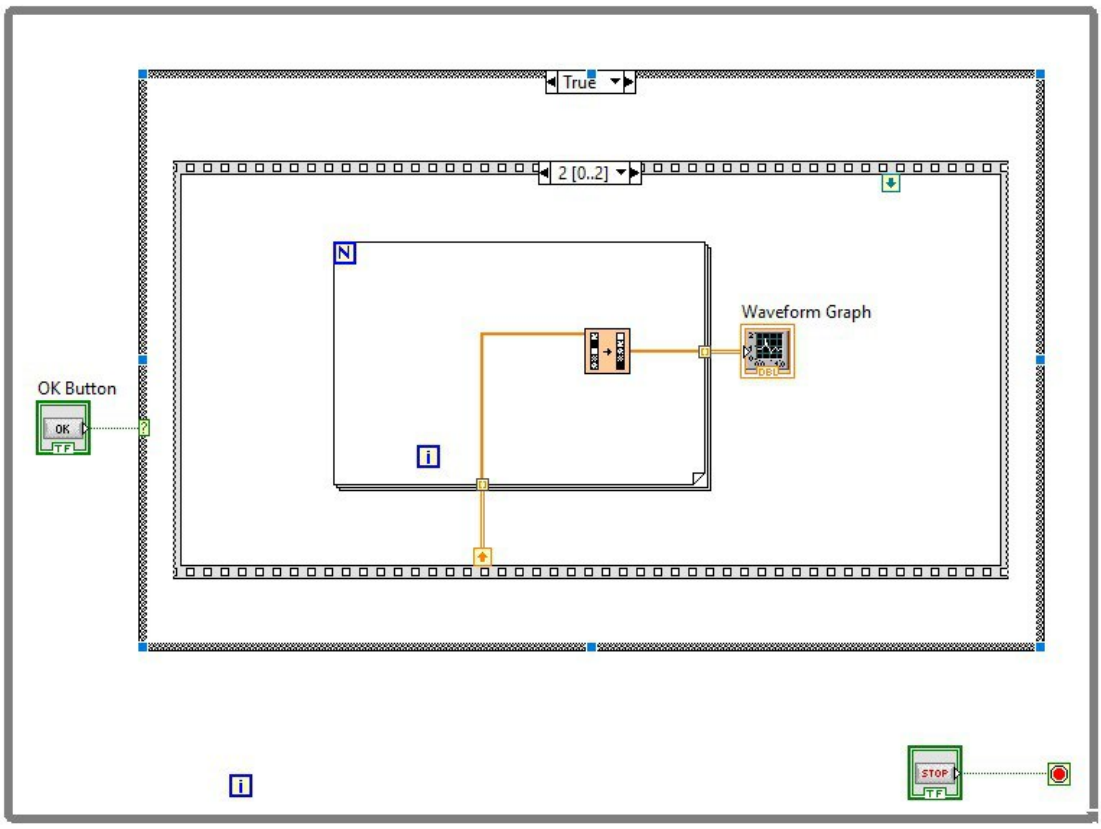


Рисунок 4.15 — Блок-діаграма третього кадру структури *Sequence* до прикладу 4.2

Запишіть результуючий двовимірний масив у файл табличного формату *Write Delimited Spreadseet* (*Functions - File* I/O), встановіть властивість *transpose* в *true* і властивість *append to file* в *true*. Створіть іншу локальну змінну на рамці структури *Sequence* і запишіть в неї результуючий двовимірний масив. У третьому кадрі структури *Sequence* (рис. 4.15) відповідно до рисунка повинно відбуватися сортування кожного одновимірного масиву та відображення результату на графіку. Запустіть прилад на виконання. Відкрийте збережений текстовий документ, куди записувалися дані під час роботи віртуального пристрою, оцініть значення та порівняйте з виведеними на графік.

Хід роботи

4.1) Виконати приклади для створення віртуальних приладів.

4.2) Отримати і ознайомитися з індивідуальним завданням.

4.3) Сформувати необхідні інструменти.

4.4) Налагодити програму і представити викладачеві для перевірки.

4.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

- звітність про виконання прикладів для створення віртуальних приладів;
- варіант і зміст індивідуального завдання;
- скріншоти складових програми і результати її виконання;
- відповіді на запитання до лабораторно роботи.

4.6) Представити звіт на захист.

Варіанти завдань

Завдання №4.1. Побудуйте віртуальний прилад, який показуватиме на екрані зашумлену функцію *sin* (максимальна амплітуда шуму вдвічі менша за амплітуду функції). Прилад повинен дозволяти змінювати число періодів, що відображуються на екрані, і забезпечувати вимірювання миттєвих значень зашумленого сигналу. Для цього використайте графічний індикатор

Waveform Chart, засіб для формування циклу *For Loop* і засіб для об'єднання масивів даних в кластер. Забезпечте можливість створення таблиці миттєвих значень функції, щоб потім знайти суму елементів масиву та середнє значення.

Завдання №4.2. Побудуйте віртуальний прилад, який показуватиме на екрані зашумлену функцію *cos* (максимальна амплітуда шуму вдвічі менша за амплітуду функції). Прилад повинен дозволяти змінювати число періодів, що відображуються на екрані, і забезпечувати вимірювання миттєвих значень зашумленого сигналу. Для цього використайте графічний індикатор *Waveform Chart*, засіб для формування циклу *For Loop* і засіб для об'єднання масивів даних в кластер. Забезпечте можливість створення таблиці миттєвих значень функції, щоб потім знайти максимальний елемент масиву.

Завдання №4.3. Побудуйте віртуальний прилад, який показуватиме на екрані зашумлену функцію *sin* (максимальна амплітуда шуму вдвічі менша за амплітуду функції). Прилад повинен дозволяти змінювати число періодів, що відображуються на екрані, і забезпечувати вимірювання миттєвих значень зашумленого сигналу. Для цього використайте графічний індикатор *Waveform Chart*, засіб для формування циклу *For Loop* і засіб для об'єднання масивів даних в кластер. Забезпечте можливість створення таблиці миттєвих значень функції, щоб потім знайти мінімальний елемент масиву.

Завдання №4.4. Побудуйте віртуальний прилад, який показуватиме на екрані зашумлену функцію *cos* (максимальна амплітуда шуму вдвічі менша за амплітуду функції). Прилад повинен дозволяти змінювати число періодів, що відображуються на екрані, і забезпечувати вимірювання миттєвих значень зашумленого сигналу. Для цього використайте графічний індикатор *Waveform Chart*, засіб для формування циклу *For Loop* і засіб для об'єднання масивів даних в кластер. Забезпечте можливість створення таблиці миттєвих значень функції, щоб потім обробити масив з використанням функції *Array Max & Min*.

Завдання №4.5. Побудуйте віртуальний прилад, який показуватиме на екрані зашумлену функцію *sin* (максимальна амплітуда шуму вдвічі менша за амплітуду функції). Прилад повинен дозволяти змінювати число періодів, що відображуються на екрані, і забезпечувати вимірювання миттєвих значень зашумленого сигналу. Для цього використайте графічний індикатор *Waveform Chart*, засіб для формування циклу *For Loop* і засіб для об'єднання масивів даних в кластер. Забезпечте можливість створення таблиці миттєвих значень функції, щоб потім обробити масив з використанням функції *Add Array Elements*.

Контрольні запитання

4.1) Що розуміється під терміном «Масив»?

4.2) Що не може бути елементами масиву у середовищі *LabVIEW*?

4.3) Дати опис *Controls Array* та *Matrix*&*Cluster*?

4.4) Як здійснюється індексація елементів масивів?

4.5) Основні способи створення масивів у *LabVIEW*?

4.6) Назвіть основні операції над масивами у *LabVIEW*?

4.7) Розкрийте основні функції роботи з масивами?

ЛАБОРАТОРНА РОБОТА № 5
ФУНКЦІЇ РОБОТИ З ДАНИМИ В LABVIEW

Мета роботи: Вивчення функцій *LabVIEW* для роботи з даними у часовій і частотній областях.

Стислі теоретичні відомості

В середовищі *LabVIEW* є широкий набір функціональних можливостей для налагодження складних програм, тестування реальних систем вимірювання і регулювання та розгорнутого аналізу отримуваних даних.

Функції генерації сигналів і шумів використовуються для формування детермінованих і випадкових сигналів із заданим набором параметрів (рис. 5.1).

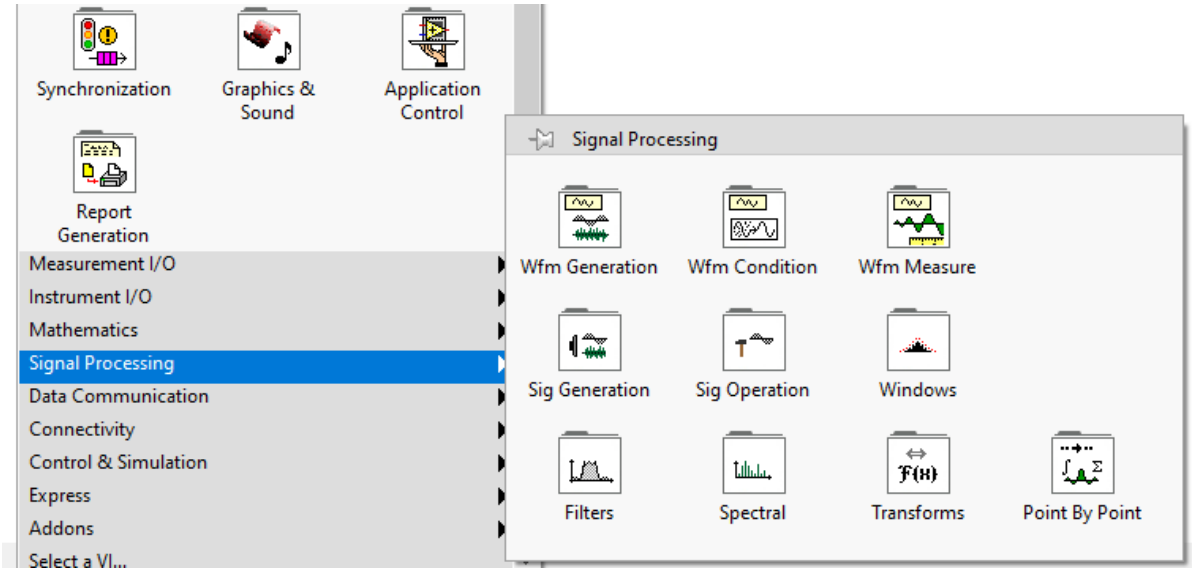


Рисунок 5.1 — Палітра функцій *Signal Processing*

Перші два прилади у верхньому ряді представляють багатофункціональні програмно регульовані генератори сигналів з широким набором контрольованих параметрів. Прилади, розміщені далі призначені для генерації найбільш широко вживаних детермінованих періодичних сигналів, для генерації шумів з різними законами амплітудного і спектрального розподілу.

Серед них:

- генератор із заданою тривалістю сигналів;

- гармонічні коливання і шум;

- відрізки синусоїдального, імпульсного, пилкоподібного, *sin*(*x*)/*y*, прямокутного і частотно-модульованого сигналів;

- синусоїдальні, трикутні, прямокутні, пилкоподібні і довільні коливання будь-якої тривалості;

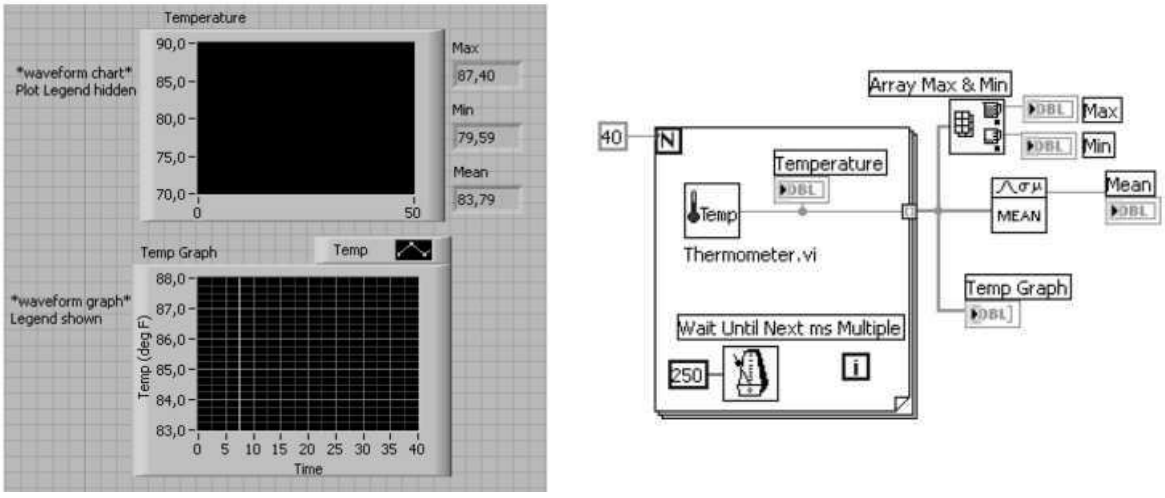
- рівномірний, Гауссовий, періодичний випадковий шуми і двійкова послідовність максимальної довжини;

- гамма-шум; Пуассонівський, біноміальний шуми; шум Бернуллі.

Приклад 5.1

Створимо віртуальний прилад (ВП), який вимірює температуру через кожні 0,25 с протягом 10 с. Під час отримання даних ВП демонструє результати вимірювань в реальному часі на розгортці осцилограми. Після завершення збору даних ВП виводить дані на графіку і обчислює мінімальне, максимальне і середнє значення температури.

Відкрийте нову лицьову панель і створіть віртуальний прилад, як показано на рис. 5.2. Якщо у вас зберігся віртуальний прилад *Thermometer*, який ви створили на першій лабораторній роботі, використайте його для цієї роботи. У іншому випадку створіть цей прилад заново.



а) б)

Рисунок 5.2 — ВП до прикладу 5.1: а) передня панель; б) блок-діаграма

Змініть масштаб графіка для отримання діапазону від 70,0 до 90,0. Переконаєтеся, що режим автоматичного налаштування шкал включений для обох осей графіка. Використовуючи інструмент введення тексту, надрукуйте слово «Темп» на панелі редагування графіка. Натисніть правою кнопкою миші (або інструментом управління) по зразку графіка «Темп» на панелі редагування і змініть тип точки (опція Тип точки) до маленьких квадратів (або будь-який інший). Задайте променям інший, більш виразний колір.

Розгортка «Температура» відображує температуру по мірі отримання даних. Після завершення збору інформації про температуру ВП виводить дані на Графіку температури. Цифрові елементи відображення Середнє, *Max* і *Min* покажуть відповідно середнє, максимальне і мінімальне значення температури.

Побудуйте блок-діаграму, як показано на рис. 5.2. Обов'язково зверніться до вікна допомоги для відображення вводів і виводів цих функцій і визначте потрібні вводи провідників, інакше матимете неприємність помилково підключитися до непотрібного терміналу.

Цикл з фіксованим числом ітерацій виконується 40 разів. Функція «Затримка» до наступного кратного інтервалу у мс спонукає кожну ітерацію виконуватися приблизно 0,25 с. Віртуальний прилад зберігає результати вимірювань температури в масиві, створеному на границі циклу з фіксованим числом ітерацій, використовуючи автоіндексування.

Після завершення циклу масив переходить до різних вузлів даних. Функція Масиву *Max* & *Min* повертає максимальне і мінімальне значення температури, функція *Mean* − середнє значення. Поверніться до лицьової панелі і запустіть віртуальний прилад.

За допомогою панелі редагування шкали (зробіть її видимою (рис. 5.3), використовуючи опцію Видимі елементи > Панель редагування шкали з контекстного меню графіка) змініть точність значень шкали *Y* так, щоб графік відображував числа з трьома знаками після коми.

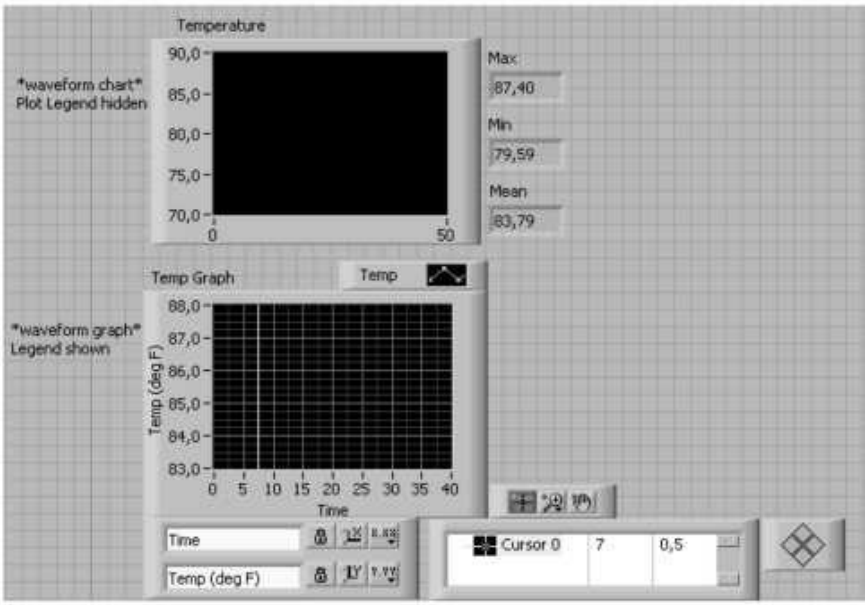


Рисунок 5.3 — Передня панель після використання функції Панелі редагування шкали

Використовуючи палітру елементів управління графіком, збільшіть його, натиснувши по кнопці *Zoom* і вибравши потрібний режим. Викличте контекстне меню графіка і відзначте опцію Видимі елементи > Панель редагування курсора. Спочатку курсор буде сірого кольору, натисніть по кнопці «Вкдючить курсор» у верхньому рядку, щоб активізувати перший курсор.

Використовуйте інструмент управління для переміщення курсора по графіку; зверніть увагу на те, як змінюються значення *X* і *Y* на дисплеї курсора. Ці значення допоможуть вам визначити координати будь-якої точки на графіці. За допомогою кнопок управління курсором переміщуйте курсор у різних напрямках.

Приклад 5.2

Створимо простий віртуальний прилад «Спектральний аналізатор прямокутного імпульсу». Основними функціональними вузлами цього віртуального приладу є генератор прямокутного імпульсу і обчислювач спектру потужності.

Генератор прямокутного імпульсу *Pulse Pattern* розміщений в палітрі *Functions* > *Analysis* > *Signal Generation*. Цей генератор формує на своєму

виході «*Pulse Pattern*» одновимірний масив відліків прямокутного імпульсу із заданою затримкою, шириною і амплітудою.

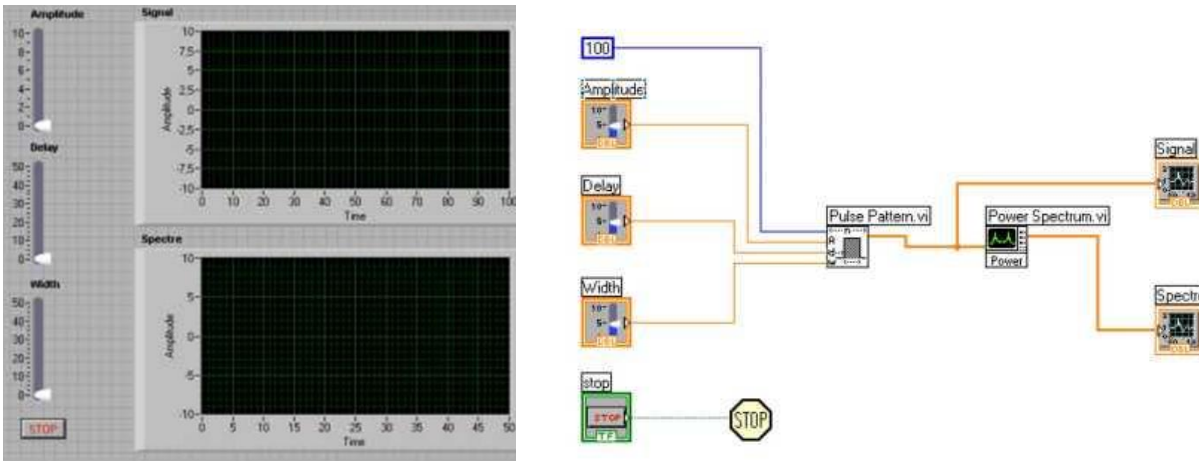
При цьому кількість відліків сигналу задається цілим числом на вході «*samples*», тривалість і затримка імпульсу (в кількості відліків) задаються цілими числами на входах «*width*» і «*delay*» відповідно, а амплітуда (в умовних одиницях) задається реальним десятковим числом на вході «*amplitude*».

Обчислювач спектру потужності *Power Spectrum* розміщений в палітрі *Functions* > *Analysis* > *Digital Signal Processing*. Для лицьової панелі виберіть два осцилографи *Waveform Graph*. Для першого в мітці, що з'явилася, введіть назву «Сигнал», а для другого − «Спектр». У об'єктному меню другого *Waveform Graph* зніміть виділення з опції *X Scale* на *Auto Scale X*. За допомогою міточного інструменту введіть кінцеве значення горизонтальної шкали 50.

З підпалітри *Numeric* палітри *Controls* виберіть три вертикальних повзункових регулятора *Vertical Pointer Slide* для регуляторів «Амплітуда», «Тривалість» і «Затримка». Розмістіть їх на лицьовій панелі зліва від індикаторів зверху вниз і введіть їх назви в мітки. За допомогою міточного інструменту змініть кінцеві значення регулювання на шкалі регуляторів «Тривалість» і «Затримка» на 50.

З підпалітри *Boolean* палітри *Controls* виберіть кнопку «СТОП» (*Rectangular Stop Button*) і розмістіть її на лицьовій панелі знизу від регуляторів. Лицьова панель повинна виглядати так, як показано на рис. 5.4.

Виберіть числову константу *Numeric Constant* (*Functions* > *Numeric*) і розмістіть її на структурній схемі вище за термінали органів управління. За допомогою міточного інструменту введіть в константу значення 100. Далі виберіть генератор прямокутного імпульсу *Pulse Pattern* (*Functions* > *Analysis* > *Signal Generation*) і розмістіть його на структурній схемі праворуч від терміналів органів управління.



а) б)

Рисунок 5.4 — ВП до прикладу 5.2: а) передня панель; б) блок-діаграма

Використайте обчислювач спектру потужності *Power Spectrum* (*Analysis* > *Digital Signal Processing*) і розмістіть його на структурній схемі між генератором прямокутного імпульсу і терміналами індикаторів. Також розмістіть функцію «Стоп» (*Functions* > *Advanced* > *Stop*) на структурній схемі поряд з терміналом кнопки «*Stop*».

З'єднаєте між собою термінали органів управління, функцій, констант та індикаторів так, як показано на рис. 5.4. Запустіть віртуальний прилад і проаналізуйте його роботу.

Приклад 5.3

Створимо генератор сигналів заданої тривалості з числом вибірок 50, частотою 10 Гц, амплітудою рівною 2, постійним зсувом, рівним 1 і початковим значенням фази рівним $\pi/2$. Подамо вихідний сигнал на однопроменевий осцилограф, збережемо в *Excel*, побудуємо графік і опишемо властивості сигналу.

Почніть з «Генератора сигналів із заданою тривалістю» (*Function* > *All Function* > *Analyzed* > *Signal Processing* > *Signal Generation*).

Примітка! Для отримання довідки по якому-небудь віртуальному приладу активізуйте його зображення за допомогою лівої кнопки миші і виберіть пункт меню *Help* (рис. 5.5).

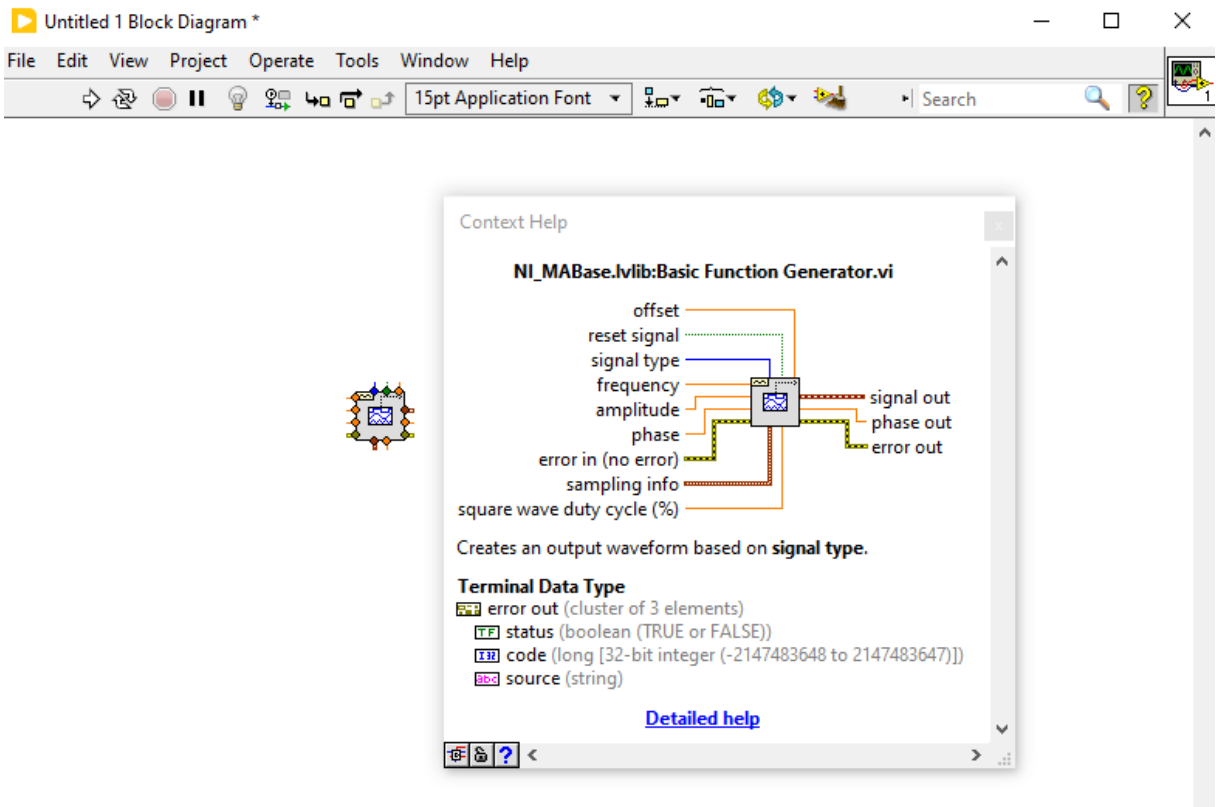


Рисунок 5.5 — Довідка *Context Help*

Розглянутий віртуальний прилад має дев'ять входів і три виходи. Входи: скинути фазу, тривалість, тип сигналу, число вибірок, частота, амплітуда, постійна складова, вхід фази, заповнення циклу прямокутного коливання (%). Виходи: сигнал, частота вибірок, вихід фази (рис. 5.6).

ВП генерує сигнал (*signal*), форма якого задається на вході signal type. Вхід *reset phase* визначає початкову фазу вихідного сигналу. За умовчуванням на вході встановлений стан ІСТИНА. При цьому початкова фаза сигналу встановлюється відповідно до значення на вході *phase in*. Якщо на вході *reset phase* встановлений стан ХИБНІСТЬ, то початкова фаза встановлюється рівною значенню фази на *phase out* при останньому виконанні цього ВП.

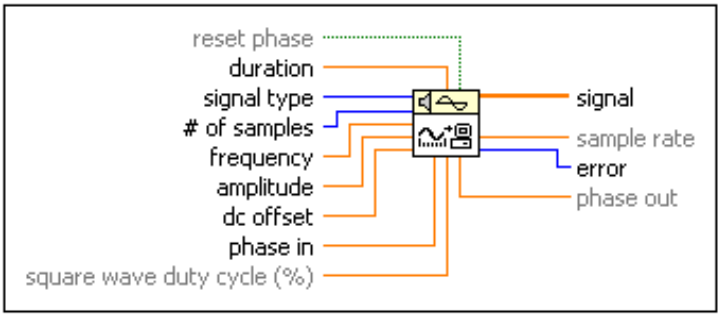


Рисунок 5.6 — Позначення всіх входів/виходів *Signal Generator VIs*

Вхід *duration* задає час в секундах, рівний тривалості вихідного генерованого сигналу. За умовчуванням значення тривалості дорівнює 1,0.

Вхід *signal type* задає наступні типи генерованого сигналу: 0 − синусоїдальний, 1 − косинусоїдальний, 2 − трикутний, 3 − прямокутний, 4 − пилкоподібний, 5 − лінійно наростаючий, 6 − лінійно спадаючий.

Вхід *# of samples* задає число вибірок вихідного сигналу. За умовчуванням це значення дорівнює 100.

Вхід *frequency* визначає частоту вихідного сигналу в герцах. За умовчуванням значення частоти дорівнює 10. При завданні частоти необхідно враховувати вимогу виконання критерію Найквіста.

Вхід *amplitude* задає амплітуду вихідного сигналу. За умовчуванням значення амплітуди дорівнює 1,0.

Вхід *dc offset* задає постійний зсув або значення постійної складової вихідного сигналу. За умовчуванням значення постійної складової дорівнює 0.

Вхід *phase* визначає початкову фазу (у градусах) вихідного сигналу при установці *reset phase* в стан ІСТИНА. За умовчуванням значення на *phase in* дорівнює 0.

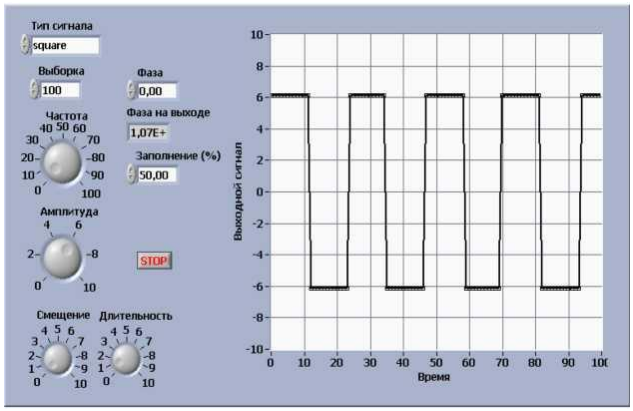
Вхід *square wave duty cycle* визначає час (у % від періоду), протягом якого прямокутний сигнал має високий рівень. ВП використовує цей параметр лише для прямокутного сигналу. За умовчуванням значення на вході дорівнює 50 %.

Вихід *signal* є згенерованим масивом вибірок сигналу.

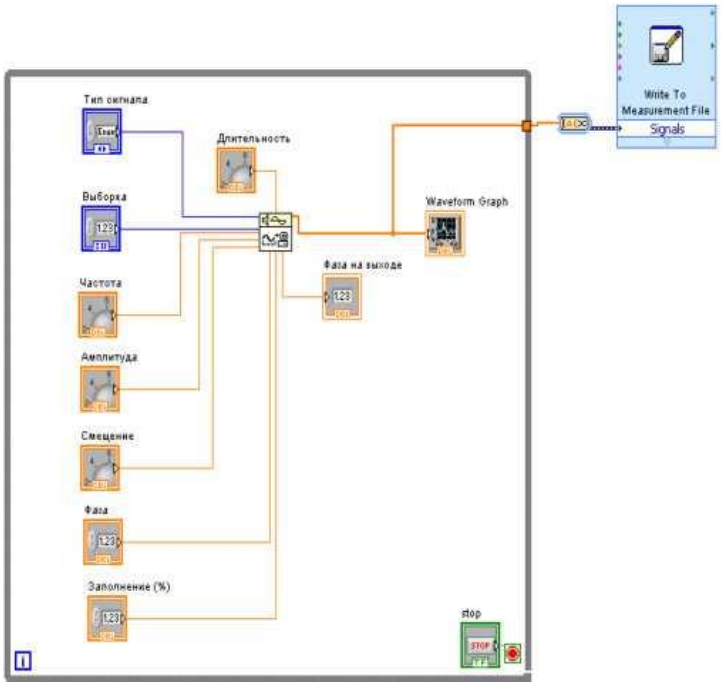
Вихід *sample rate* відображує частоту дискретизації вихідного сигналу. Частота вибірок дорівнює відношенню числа вибірок до тривалості.

Вихід *phase out* вказує значення фази (у градусах) останньої вибірки вихідного сигналу.

Подайте вихідний сигнал на однопроменевий осцилограф. Запустіть програму і проаналізуйте властивості сигналу. Визначте зсув (рис. 5.7).



а)



б)

Рисунок 5.7 — ВП генератору сигналів: а) передня панель;
б) блок-діаграма

Збережіть отримані дані в *Excel*. Процедура передачі даних в *Microsoft Excel* полягає в наступному. Для передачі в *Excel* часових характеристик сигналу, що відображуються компонентом *Waveform Graph*, при

працюючому віртуальному приладі натисніть правою кнопкою миші на полі індикатора *Waveform Graph* і потім у випадному меню виберіть опцію *Export*.

У наступному вікні виберіть опцію *Export Data To Excel*. В результаті в створений аркуш *Microsoft Excel* будуть передані дані, представлені в двох колонках.

Побудуйте в *Microsoft Excel* графік і опишіть властивості сигналу.

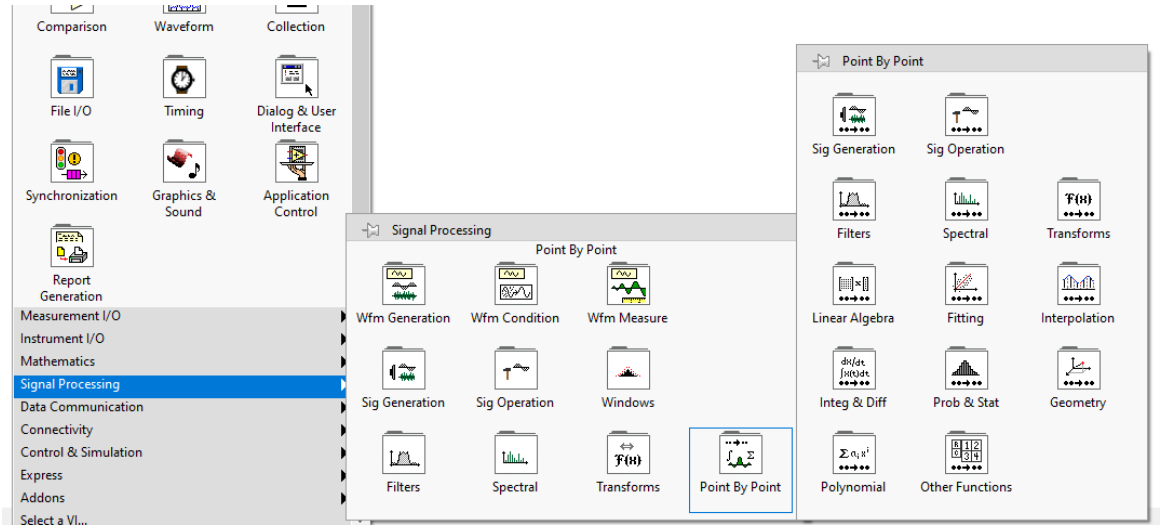
Приклад 5.4

Використаємо генератор, створений в прикладі 5.3, для моделювання вхідних сигналів системи регулювання. Створимо систему і програму її тестування. Додамо до вхідного сигналу випадковий Гауссовий шум з амплітудою рівною 1, запустимо програму і проаналізуємо зміну спектру вихідного сигналу.

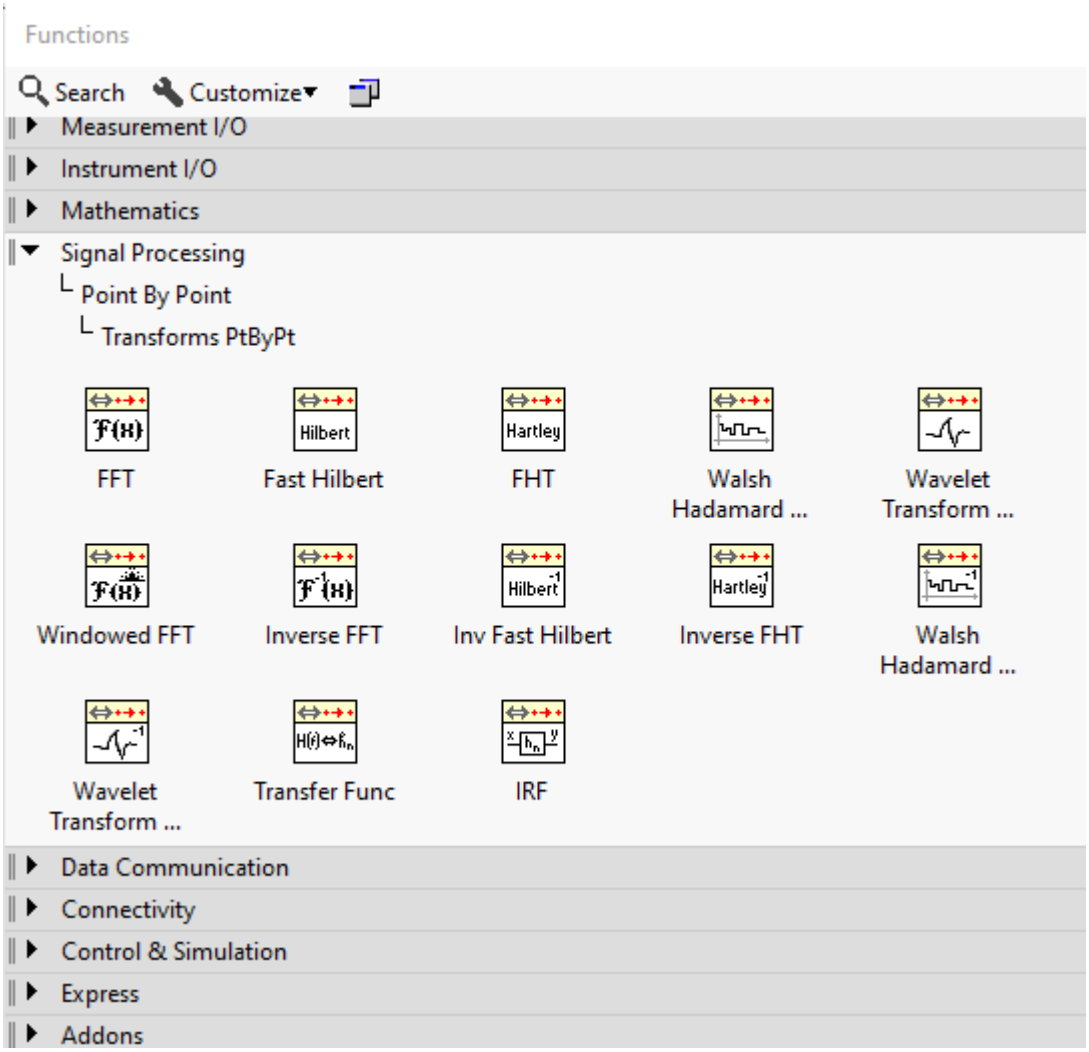
Створіть систему регулювання, що формує керуючий сигнал на відключення устаткування при досягненні значення згенерованої функції 0,75 частини її максимуму.

Для чого виберіть Палітру функцій обробки сигналів в частотній області (*Function > All Function > Signal Processing > Point By Point > Transforms >Transfer Function*), яка містить набори функцій, що дозволяють виконати пряме і зворотне перетворення Фур'є, Гільберта, Хартлі і Уолша-Адамара, а також пряме і зворотне вейвлет-перетворення (рис. 5.8).

Зверніть увагу! На базі функцій перетворення Фур'є розроблений ряд високорівневих приладів для оцінки взаємного спектру потужності, імпульсної і частотної передавальної характеристик кіл, частотної функції когерентності і вимірювач гармонічних спотворень сигналу, що розміщені на цій палітрі.



а)



б)

Рисунок 5.8 — Палітра функцій обробки сигналів: а) в розгорнутому виді;б) через меню пошуку функцій *Search*

Transfer function (передавальна функція) виконує розрахунок однобічної передавальної функції, також відомої як частотна передавальна функція, на основі аналізу заданих в часовій області тестувального сигналу (*Stimulus Signal*) і вихідного сигналу тестованого об'єкту (*Response Signal*) на вході і виході тестованого електричного кола (рис. 5.9).

Додайте до схеми генератора Гауссовий шум із стандартними параметрами. Для цього використайте *Gaussian White Noise* (Білий Гауссовий шум), який генерує псевдовипадкову послідовність з розподілом Гаусса (нормальний розподіл) з параметрами: вибірка (*samples*), стандартне відхилення (*standart deviation*), початкове значення (*seed*).

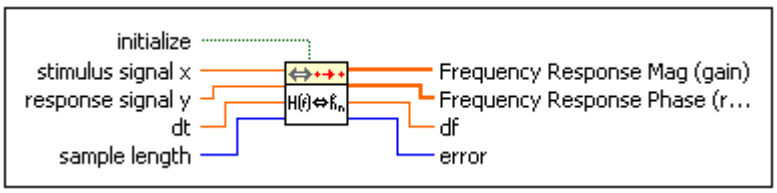


Рисунок 5.9 — Позначення передавальної функції *Transfer function*

За умовчуванням значення стандартного відхилення дорівнює 1,0. Створіть програму його тестування (рис 5.10, 5.11).

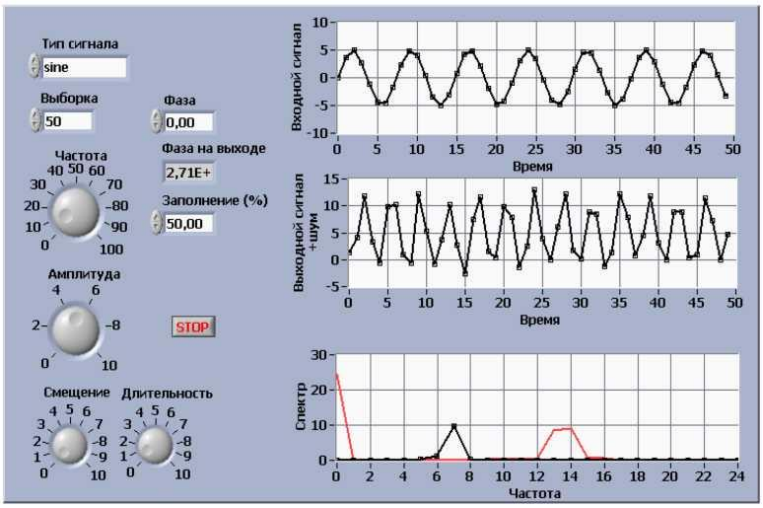


Рисунок 5.10 — Передня панель до прикладу 5.4

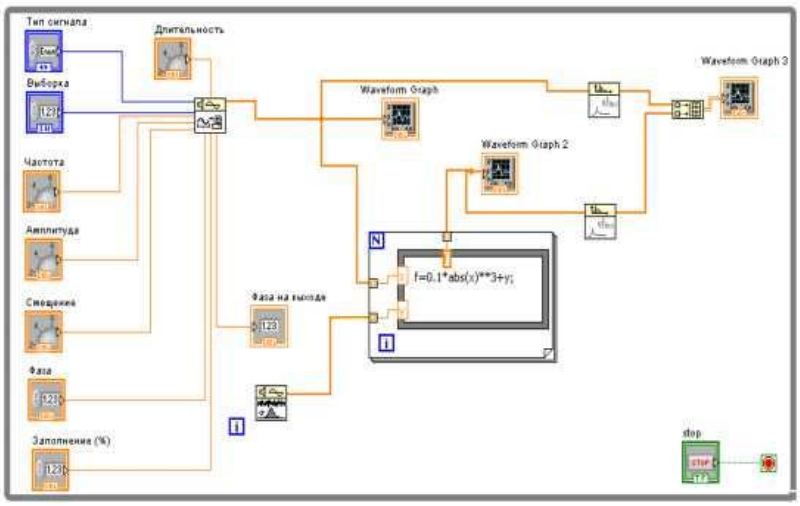


Рисунок 5.11 — Блок-діаграма до прикладу 5.4

Додайте до сигналу генератора випадковий Гауссовий шум з амплітудою рівною 1. Подайте результуючий сигнал на вхід аналізатора спектру.

Запустіть програму і проаналізуйте реакцію системи регулювання за зміною спектру вихідного сигналу.

Хід роботи

5.1) Виконати приклади для створення віртуальних приладів.

5.2) Отримати і ознайомитися з індивідуальним завданням.

5.3) Сформувати необхідні інструменти.

5.4) Налагодити програму і представити викладачеві для перевірки.

5.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

- звітність про виконання прикладів для створення віртуальних приладів;
- варіант і зміст індивідуального завдання;
- скріншоти складових програми і результати її виконання;
- відповіді на запитання до лабораторно роботи.

5.6) Представити звіт на захист.

Варіанти завдань

Завдання №5.1. Створіть віртуальний прилад (ВП), який вимірює тиск через кожні 0,5 с протягом 20 с. Під час здобуття даних ВП демонструє результати вимірів в реальному часі на розгортці осцилограми. Після завершення збору даних ВП імпортує результати в *MS Excel*. Побудуйте в *MS Excel* відповідні графіки.

Завдання №5.2. Створіть віртуальний прилад «Спектральний аналізатор прямокутного імпульсу». Основними функціональними вузлами цього віртуального приладу є генератор прямокутного імпульсу і обчислювач спектру потужності. Вимір проводиться через кожні 1,5 с протягом 15 с. Як результат прилад виводить середнє значення спектру потужності.

Завдання №5.3. Створіть генератор сигналів заданої тривалості з числом вибірок 150, частотою 50 Гц, і випадковою амплітудою, що не перевищує значення, рівне 1, постійним зсувом, рівним 1 і початковим значенням фази, рівним $\pi/4$. Прилад повинен виводити вихідний сигнал на однопроменевий осцилограф і зберігати дані в *Excel*. Побудуйте в *MS Excel* відповідні графіки.

Завдання №5.4. Створіть генератор для моделювання вхідних сигналів системи регулювання з можливістю додавання до вхідного сигналу випадкового Гауссового шуму з амплітудою рівною 1. Для сигналу число вибірок 80, частота 20 Гц, амплітуда дорівнює 2, постійний зсув рівний 0,5 і початкове значення фази рівне $\pi/6$. Прилад повинен виводити вихідний сигнал на однопроменевий осцилограф і зберігати дані в *Excel*.

Контрольні запитання:

5.1) Для яких цілей в *LabVIEW* використовуються функції генерації шумів і сигналів?

5.2) Які властивості тестованого об'єкту характеризує передавальна функція?

5.3) Як за допомогою контекстної довідки визначити обов'язкові, рекомендовані та опціональні входи і виходи генератора сигналу із заданою тривалістю?

5.4) За допомогою яких елементів пакету *LabVIEW* можна досліджувати спектр вихідного сигналу?

5.5) Поясніть за структурною схемою вашого віртуального приладу призначення його вузлів, функцій, органів управління та індикаторів, порядок роботи віртуального приладу.

5.6) Якою функцією щільності вірогідності описується Гауссовий шум?

5.7) Охарактеризуйте генератор прямокутного імпульсу (*Functions* > *Signal Processing* > *Wfm Generation (Waveform Generation*)) > *Square Waveform*?

5.8) Охарактеризуйте обчислювач спектру потужності (*Functions* > *Signal Processing* > *Wfm Measurements (Waveform Measurements*)) > *Spectral*)?

ЛАБОРАТОРНА РОБОТА № 6
ЗАСОБИ РОБОТИ ЗІ СТРОКАМИ В LABVIEW

Мета роботи: набути навики роботи зі строковими даними в *LabVIEW*.

Стислі теоретичні відомості

Подібно до всіх мов програмування високого рівня, в *LabVIEW* також реалізована робота зі строками. Строки в *LabVIEW* − це ще один тип даних, для роботи з якими існують свої функції, індикатори і елементи управління.

Строки − це послідовність відображуваних і не відображуваних символів таблиці *ASCII*. Строки забезпечують незалежний від платформи формат обміну даними (рис.6.1). Деякі з найбільш поширених строкових додатків включають:

- створення простих текстових повідомлень;
- передачу числових даних в прилади у вигляді рядків символів і перетворення строк в числові дані;
- збереження числових даних на диск, з процедурою перетворення числових даних з двійкового формату у формат символів *ASCII* строк перед записом у файл;
- діалогові вікна інструкцій і підказок.

Строкові змінні в *LabVIEW* мають те ж значення, що і в мовах програмування високого рівня.

На лицьовій панелі приладів строки можуть бути представлені у вигляді таблиць, полів введення тексту і міток. Для розміщення строкового елементу на лицьовій панелі необхідно викликати палітру елементів *Modern* і вибрати в ній групу «*String* & *Path*» (рис. 6.1).

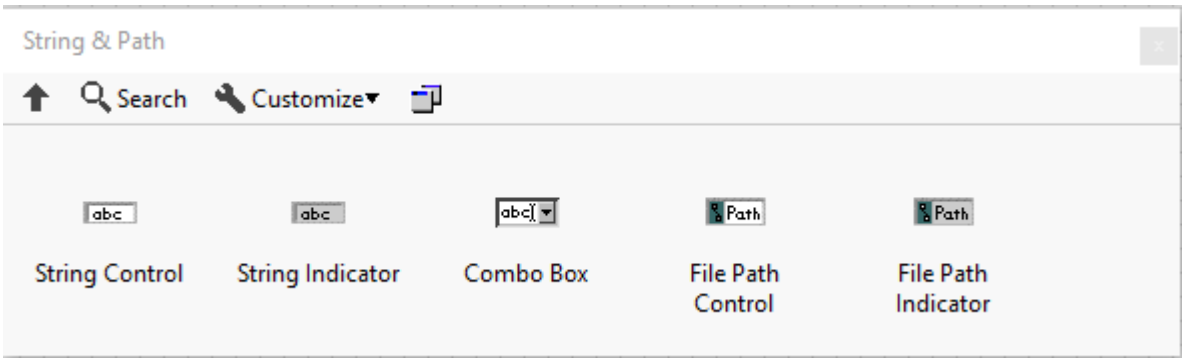


Рисунок 6.1 — Розміщення строкового елементу на лицьовій панелі

Строка, розміщена на лицьовій панелі, може мати декілька типів відображення строкових даних (рис. 6.2), режим стандартного відображення, режим відображення із зворотним слешем для виведення керуючих кодів, режим прихованого відображення тексту і режим відображення у вигляді шістнадцятирічних *ASCII*-кодів символів.

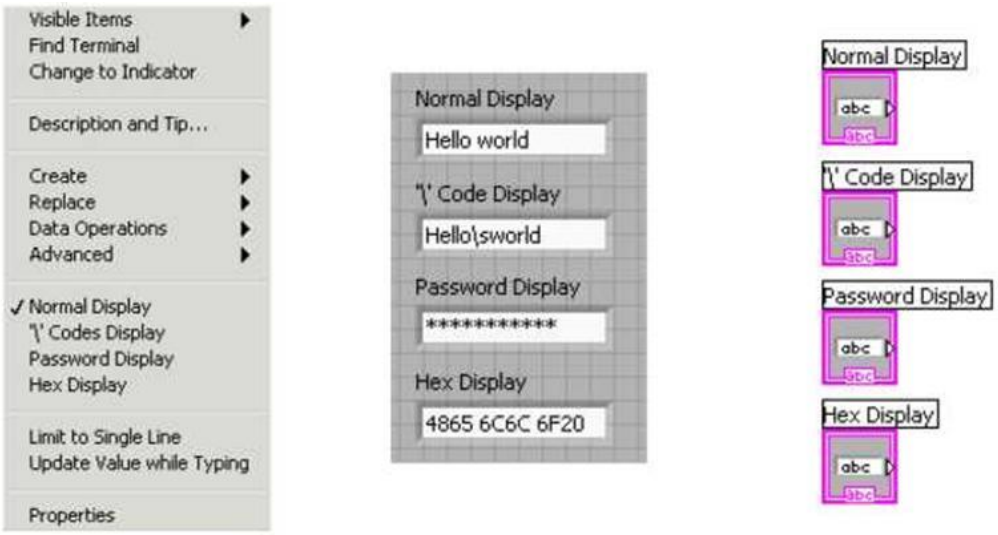


Рисунок 6.2 — Типи відображення строкових даних

Для зміни типу відображення символів потрібно правим натисканням миші викликати контекстне меню строкового елементу і вибрати необхідний тип відображення − «*Normal Display*», «*'V Code Display*», «*Password Display*», «*Hex Display*».

На панелі діаграм строкові елементи будуть показані у вигляді звичайних терміналів, так само, як і інші елементи.

Строки, подібно до інших елементів-змінних в *LabVIEW* можуть бути елементами управління і елементами відображення даних, елементами-джерелами і елементами-приймачами.

У палітрі функцій для виконання операцій над строковими змінними відведена група функцій *String* (рис. 6.3).

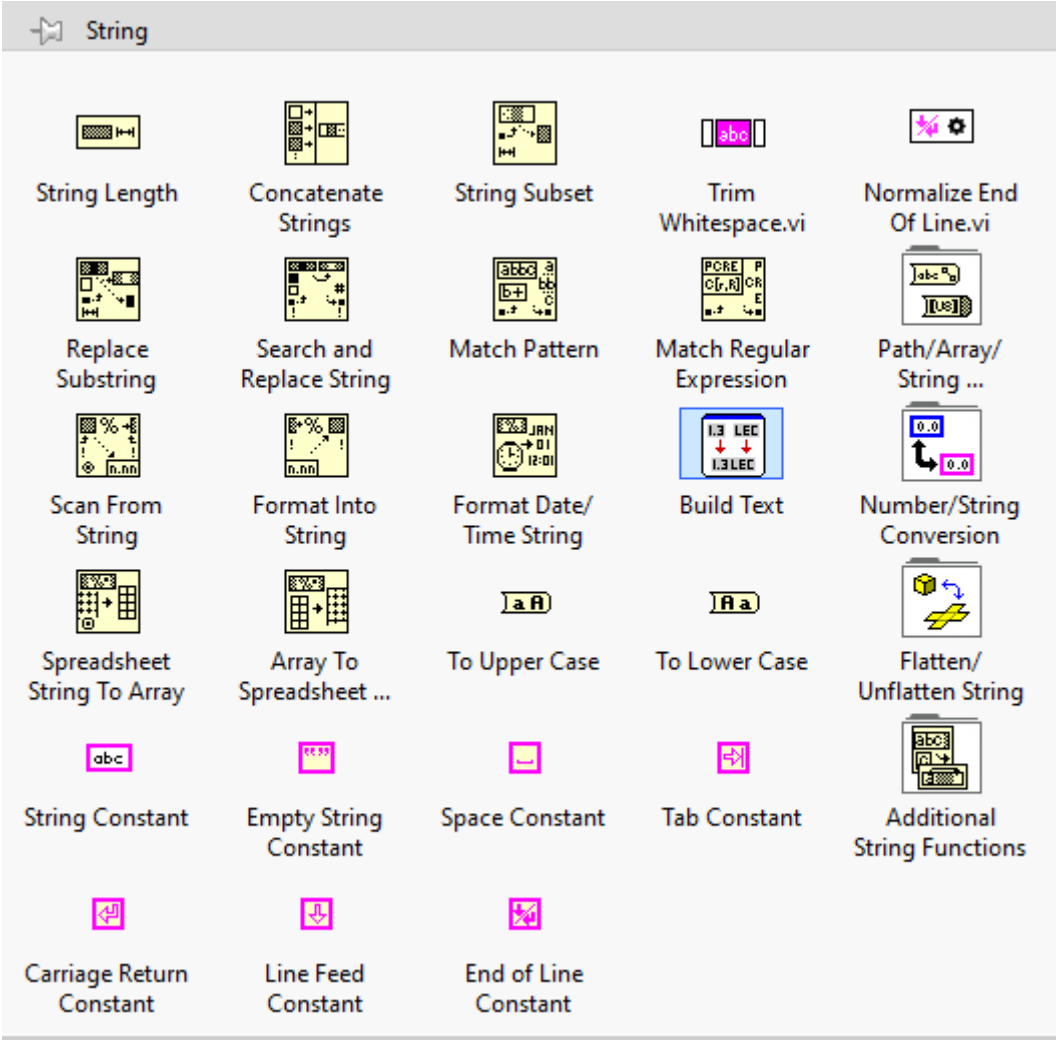


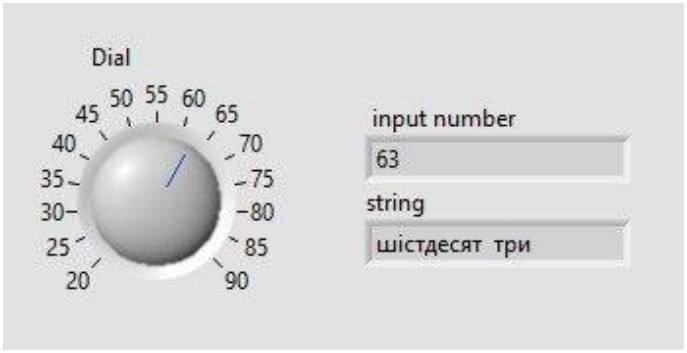
Рисунок 6.3 — Палітра функцій *String*

Правила з'єднання строкових елементів між собою нічим не відрізняються від правил з'єднання числових і логічних елементів. Лінії з'єднання між строковими елементами мають рожевий колір. Строкові елементи по аналогії з чисельними і логічними можуть складати масиви рядків і входити до складу кластерів.

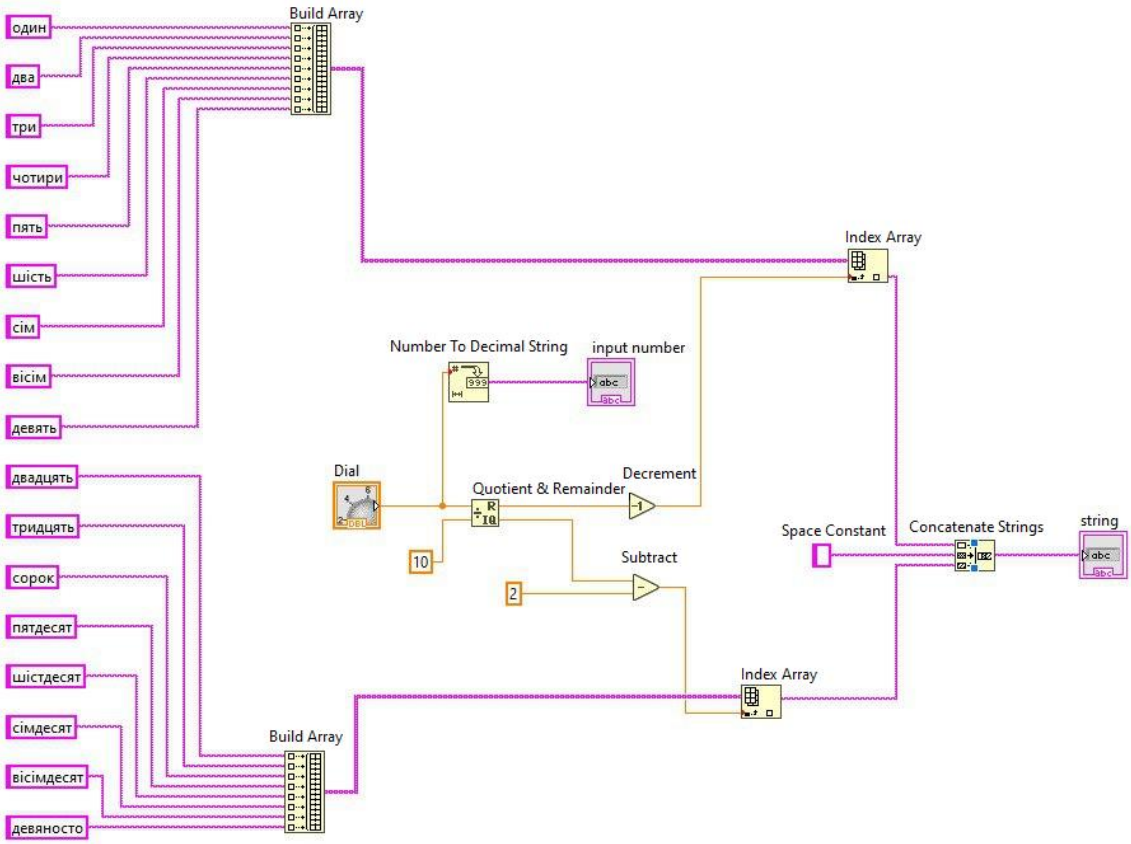
Приклад 6.1

Розробимо віртуальний прилад, який перетворить числове представлення числа в строкове, записане словами (для простоти побудови візьмемо діапазон чисел від 20 до 90). Реалізація ВП приведена на рис. 6.4.

Використовуючи отриманий досвід у попередніх лабораторних роботах, самостійно створіть і оформіть віртуальний прилад, представлений на рис. 6.4.



а)



б)

Рисунок 6.4 — ВП до прикладу 6.1: а) передня панель; б) блок-діаграма

При роботі зі строками виникає необхідність використовувати функції файлового введення/виводу.

Функції файлового введення/виводу виконують файлові операції запису і зчитування даних. Функції файлового введення/виводу розташовані в палітрі функцій «*File* I/O» (рис. 6.5) і призначені для:

- відкриття і закриття файлу даних;
- зчитування і запису даних із/у файл(а);
- зчитування і запису даних із/у файл(а) у вигляді таблиці символів;
- переміщення і перейменування файлів і каталогів;
- зміни характеристик файлу;
- створення, зміни і зчитування файлів конфігурації.

Палітру функцій роботи з файлами можна розділити на три частини: функції високого рівня, функції низького рівня і підпалітру розширених можливостей.

Функції файлового введення/виводу високого рівня розташовані у верхньому рядку палітри «*File* I/O» і призначені для виконання основних операцій по введенню/виводу даних.

Функції файлового введення/виводу низького рівня розташовані в середньому рядку палітри «*File* I/O». Вони використовуються для створення нового або звернення до раніше створеного файлу, запису і зчитування даних і закриття файлу.

Функції низького рівня роботи з файлами підтримують всі операції, необхідні при роботі з файлами.

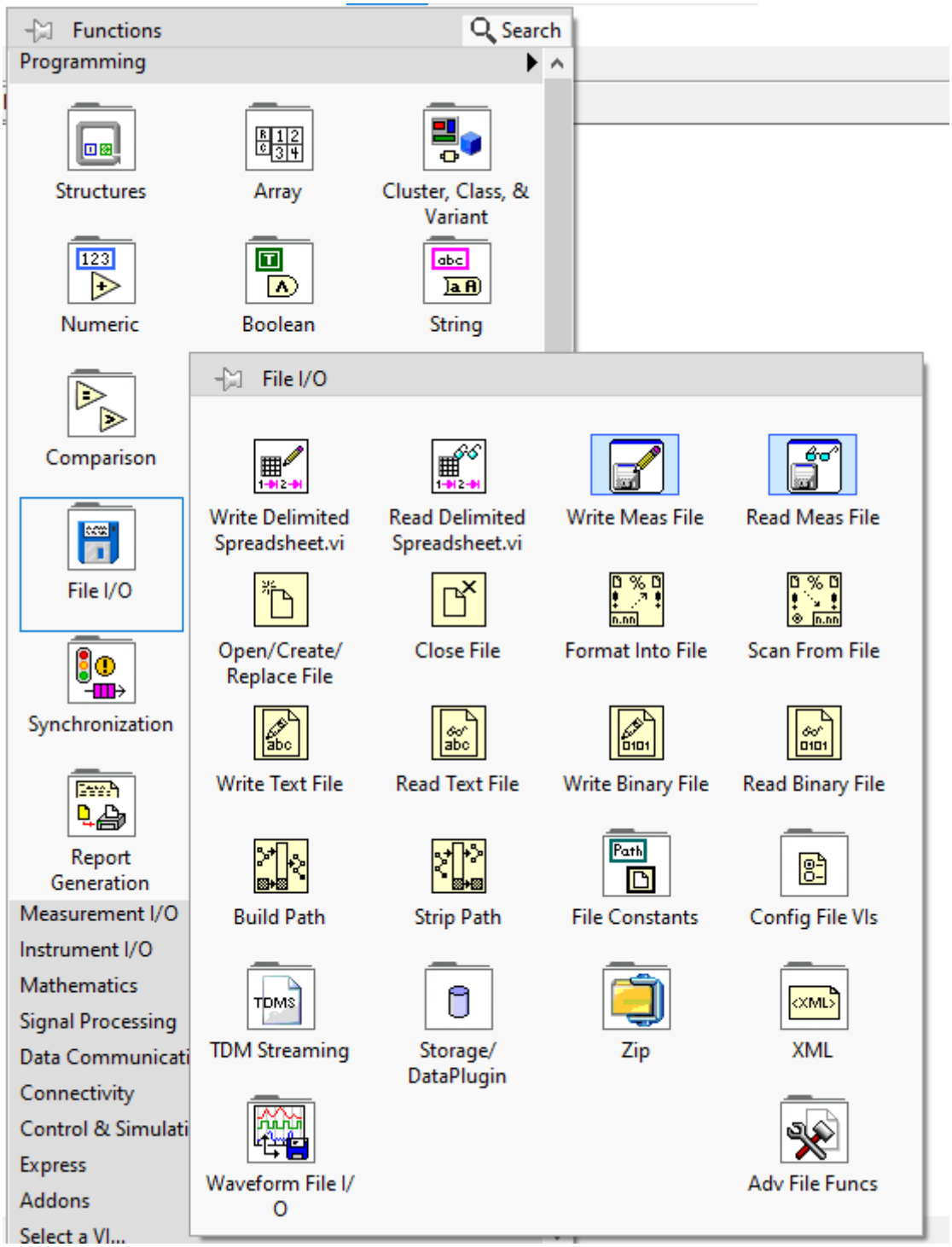


Рисунок 6.5 — Палітра *File* I/O

Приклад 6.2

Для здійснення стандартної процедури введення/виводу даних в/із файл(а) необхідно виконати наступну послідовність дій:

1. Створити або відкрити файл. Вказати місце розташування існуючого файлу або шлях для створення нового файлу за допомогою діалогового вікна *LabVIEW*. Після відкриття файл *LabVIEW* створює

посилання на нього.

2. Виконати операції зчитування або запису даних в/із файл(а).

3. Закрити файл.

4. Обробити помилки.

На рис. 6.6 приведений приклад простого віртуального приладу що виконує запис текстової строки у файл.

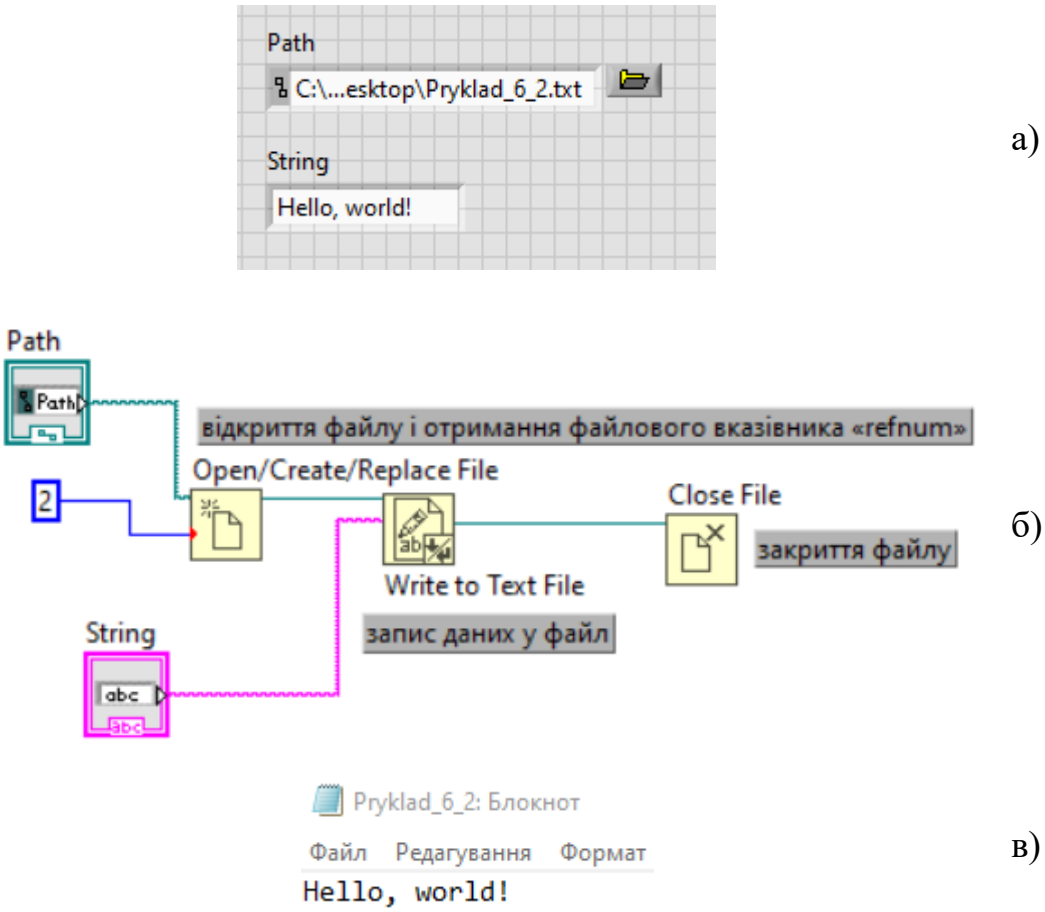


Рисунок 6.6 — ВП із запису текстової строки у файл:

а) передня панель; б) блок-діаграма; в) результат запису у файл

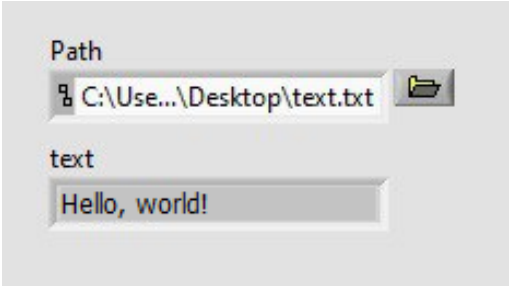
На передній панелі приладів розташовано два елементи: строковий елемент, що містить текст, який буде записаний у файл, і елемент «*File Path Control*» з палітри «*String&path*», який містить шлях та ім'я файлу, з яким взаємодіятиме програма.

На функціональній схемі за допомогою вузла «*Open/Create/Replace File*» відбувається відкриття файлу і отримання файлового вказівника «*refnum*». На вхід вузла подається два сигнали: шлях та ім'я файлу, і число,

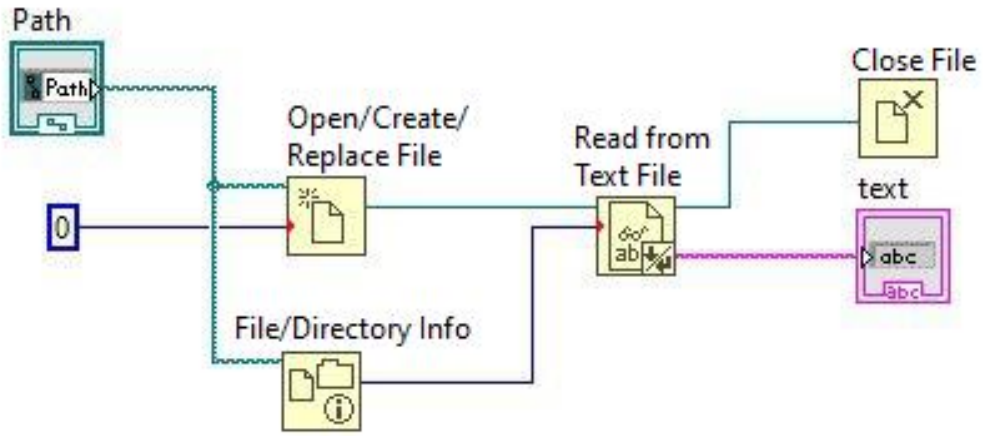
що відповідає функції, яка виконуватиметься над вказаним файлом (у нашому випадку ми вказуємо число 2 − створити новий файл або перезаписати існуючий).

Далі за допомогою вузла «*Write file*» виконується запис даних у файл і закриття файлу за допомогою вузла «*Close file*». Після виконання цієї програми на диску «*h*:\» отримаємо файл «*test.txt*» з текстом «Ця строка запишеться у файл».

На рис. 6.7 приведений приклад програми, яка дозволяє прочитувати дані з файлу.



а)



б)

Рисунок 6.7 — ВП по зчитуванню даних з файлу:

а) передня панель; б) блок-діаграма

Для зчитування інформації з файлу використовується вузол «*Read File*», на який подається файловий вказівник «*refnum*» і кількість символів (байт), яку потрібно підрахувати у файлі. Розмір файлу визначається за допомогою вузла «*File/Directory Info*».

Приклад 6.3

Створимо програму, яка реалізує конкатенацію (злиття) рядків і освоєння технології зміни властивостей елементів управління та індикаторів.

Для реалізації конкатенації (злиття) рядків створіть новий додаток. Встановіть на панель інтерфейсу два поля введення строки (строк): *Controls* > *String* & *Path* > *String Control*. Винесіть на панель текстовий індикатор для виведення результату: *Controls* > *String* & *Path* > String Indicator. Перейменуйте всі встановлені на панелі елементи, як показано на рис. 6.8. У вікні редагування діаграм встановіть піктограму функції складання рядків: *Functions* > *String* > *Concatenate Strings*. З'єднайте відповідні виводи, як показано на діаграмі.

а) б)

Рисунок 6.8 — ВП до прикладу 6.3: а) передня панель; б) блок-діаграма

Виконайте тестування програми. Заповніть першу і другу строку введення довільною послідовністю символів, і запустіть програму на виконання.

У контекстному меню до рядка є опція *Password Display*. Ця опція перемикає поле введення в режим, призначений для введення пароля. При цьому замість символів, що вводяться, поле заповнюватиметься зірочками.

Приклад 6.4

Створимо програму, яка при правильному або неправильному введенні пароля інформує про це користувача шляхом зміни кольору "овальної лампочки" індикатора (рис. 6.9).

Для чого перейдіть у вікно редагування діаграм і встановіть знак порівняння на перевірку на рівність введеного пароля і константи, з якою він порівнюється: *Functions* > *Comparison* > *Equal*?.

З'єднайте строку введення та індикатор з відповідними виводами. Для створення константи, з якою порівнюватиметься рядок вводу, підведіть вказівник миші у вигляді котушки до другого виводу функції порівняння і натисніть праву клавішу миші. Створіть константу пароля, наприклад, "*mypass*". Перевірте програму на працездатність.

а) б)

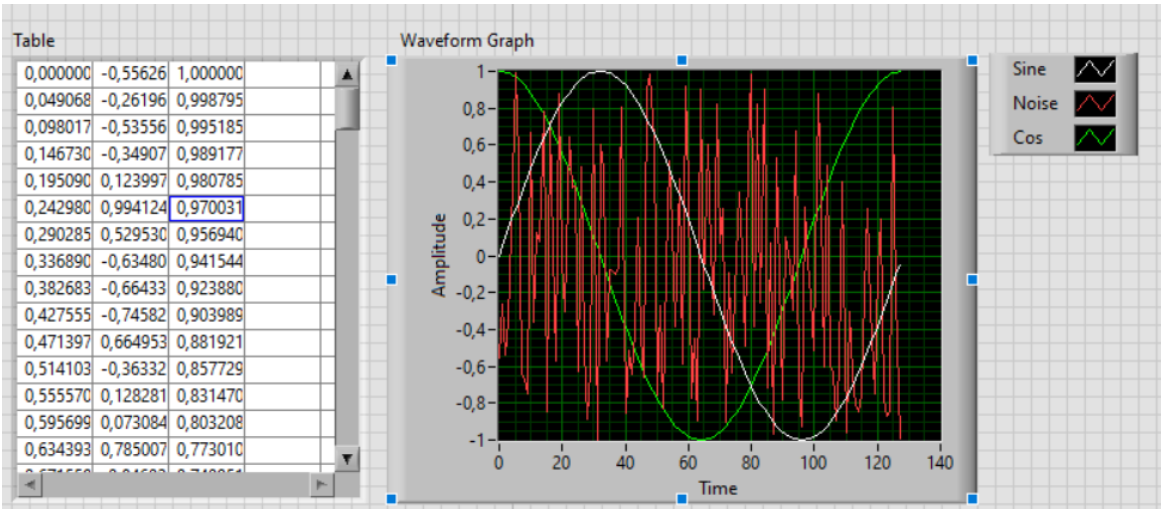
Рисунок 6.9 — ВП до прикладу 6.4: а) передня панель; б) блок-діаграма

Перейдіть на панель інтерфейсу. Введіть вірний пароль і запустіть на виконання, а потім невірний. Проаналізуйте отриманий результат. Спробуйте модифікувати цю програму.

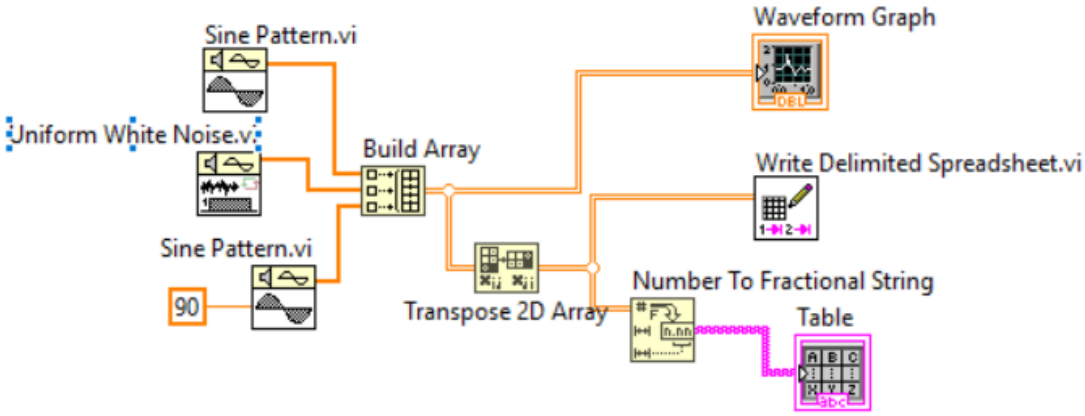
Приклад 6.5

Створимо програму, яка генеруватиме 2-х мірний масив з 128 рядків і 3-х стовпців. Перший стовпець міститиме дані синусоїдальної хвилі, другий − шумову хвилю, а третій − косинусоїдальну хвилю.

Окрім цього, результат формування хвиль відображуватиметься у вигляді графіків на одній площині і в табличному вигляді. Лицьова панель та блок-схема ВП представлені на рис. 6.10.



а)



б)

Рисунок 6.10 — ВП до прикладу 6.5: а) передня панель; б) блок-діаграма

Більш детально розглянемо вилучення даних з текстових файлів з таблицями та їх обробку в середовищі *LabVIEW*.

В процесі експерименту, часто доводиться застосовувати операції порівняння вимірюваної фізичної величини з табличними значеннями, отриманими теоретично або емпірично (наприклад, калібрувальні таблиці даних датчиків).

Допустимо, маємо напівпровідниковий резистор, температурна залежність опору якого описується експонентою, при зменшенні температури опір зростає. Такого роду резистори зазвичай використовують як термометри під час низькотемпературних досліджень.

Для того, щоб реалізувати термометр на напівпровідниковому резисторі необхідно його відкалібрувати. При цьому потрібно використати дані еталонного термометра, калібрування якого відоме, або чисельного вирішення аналітичних виразів, що описують залежність опору від температури для конкретного резистора.

Записавши масив даних у вигляді двох стовпців *R*[Ом] і *T*[К] в файл *calibration.txt*, потрібно вирішити завдання автоматичного порівняння значення в таблиці з вимірюваним опором і виконати зіставлення з відповідною температурою.

Для цього в *LabVIEW* існує набір спеціальних терміналів роботи з *DATA* файлами. Використовуючи ці термінали, можна вилучити потрібне значення з одного стовпця, а потім привласнити йому відповідне значення іншого стовпця. У нашому випадку краще всього для роботи з дійсними числами підходить термінал *Read Delimited Spreadsheet* (рис. 6.11)

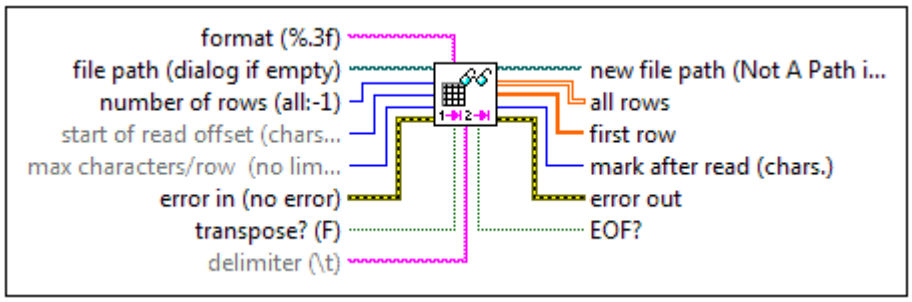


Рисунок 6.11 — Термінал *Read Delimited Spreadsheet*

Зразки використання цього терміналу представлено на рис. 6.12 та 6.13.

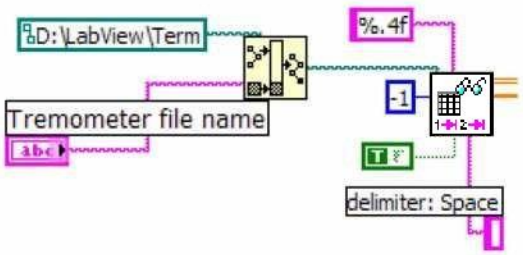


Рисунок 12 — Зразок 1 використання терміналу *Read Delimited Spreadsheet*

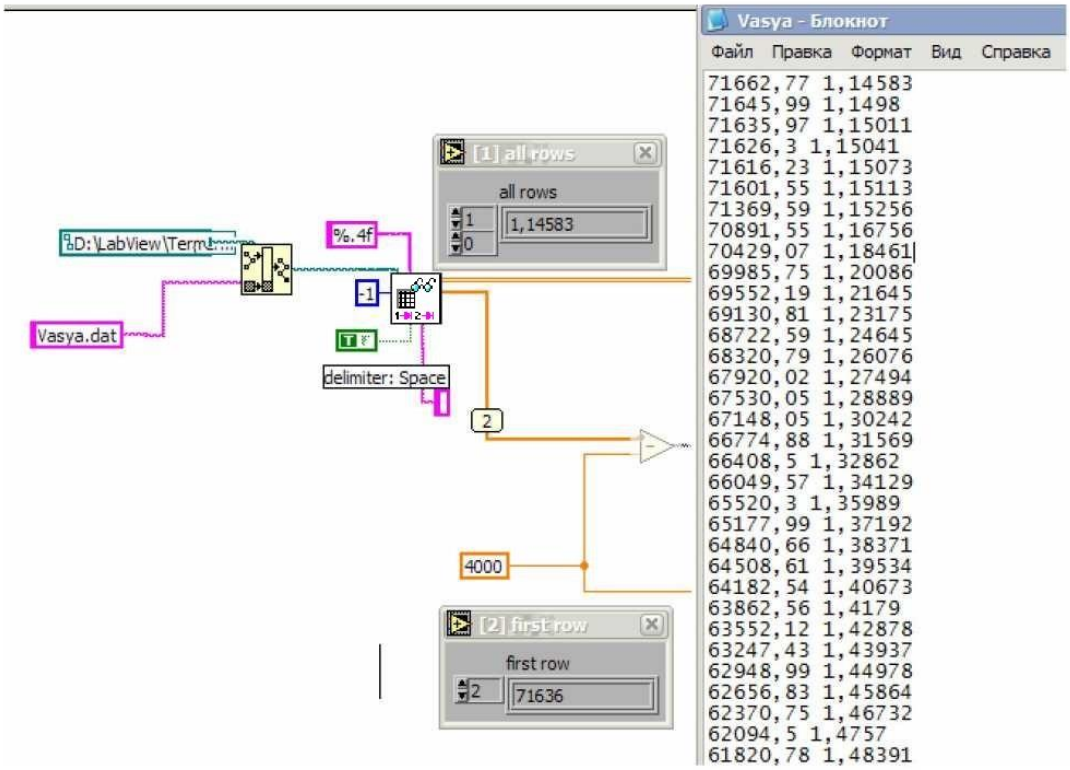


Рисунок 13 — Зразок 2 використання терміналу *Read Delimited Spreadsheet*

При цьому (повернемося до прикладу 6.5) матимемо два масиви 2*d* і 1*d* (рис. 6.14), що складаються з двох і одного стовпців відповідно. В процесі роботи віднімається реальне виміряне значення (4000 Ом) по всьому масиві. Це необхідно, щоб знайти координати елементу з мінімальним по модулю значенням в стовпці та відповідну йому температуру. Для виділення мінімального або максимального елементів масиву зручно використати оператор *Array Max* & *Min*, а для заміщення елементів можна використати *Replace Array Subset* (рис. 6.14).

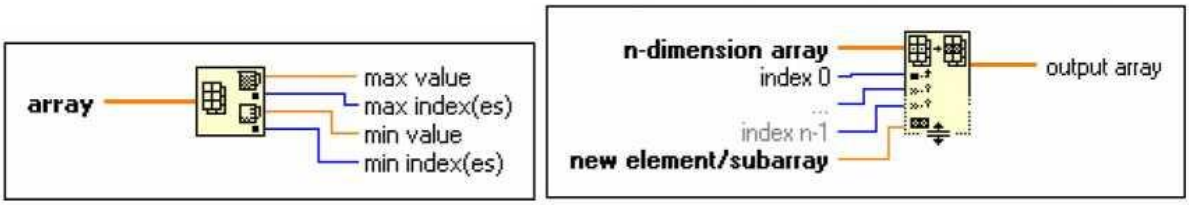


Рисунок 6.14 — Використання операторів *Array*:
а) *Array Max* & *Min*; б) *Replace Array Subset*

Вибір елементу з повного масиву виконує термінал *Index Array* (рис. 6.15).

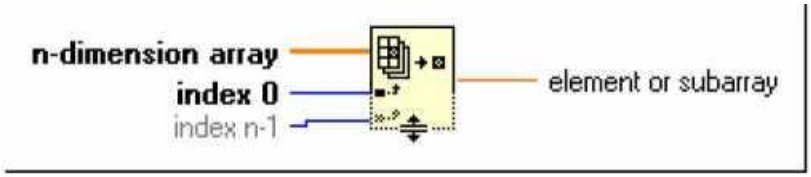


Рисунок 6.15 — Термінал *Index Array*

Також, доцільно використати усереднення по двох найближчих значеннях (у випадку якщо значення не відповідає оціночному числу), при цьому необхідно виконати зсув масиву і повторити пошук мінімуму.

Хід роботи

6.1) Виконати приклади для створення віртуальних приладів.

6.2) Отримати і ознайомитися з індивідуальним завданням.

6.3) Сформувати необхідні інструменти.

6.4) Налагодити програму і представити викладачеві для перевірки.

6.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

- звітність про виконання прикладів для створення віртуальних приладів;
- варіант і зміст індивідуального завдання;
- скріншоти складових програми і результати її виконання;
- відповіді на запитання до лабораторно роботи.

6.6) Представити звіт на захист.

Варіанти завдань

Завдання №6.1. Розробіть ВП, який виконуватиме такі дії:

- згенерує значення опору, що працює як термопара, в діапазоні 500±100 Ом;
- згенерує покази температури в діапазоні 300±10 К (незалежно від значень опору);
- переведе отримані значення з числового формату в строковий;
- запише послідовно масив даних у вигляді двох стовпців *R*[Ом] і

T[К] в файл *.*txt*;

- вилучить дані з текстового файлу;
- зіставить значенням елементів масиву показань опорів значення елементів масиву показань температури (перегрупування масиву показань температури);
- виведе максимальне значення температури та відповідне йому значення опору.

Завдання №6.2. Розробіть ВП, який виконуватиме такі дії:

- згенерує значення напруги, що визначається обертами двигуна, в діапазоні 9±3 В;
- згенерує покази швидкості обертів двигуна в діапазоні 500±200 об/хв (незалежно від значень напруги);
- переведе отримані значення з числового формату в строковий;
- запише послідовно масив даних у вигляді двох стовпців *V*[В] і *W*[об/хв] в файл *.*txt*;
- вилучить дані з текстового файлу;
- зіставить значенням елементів масиву показань напруг значення елементів масиву показань швидкості обертів (перегрупування масиву показань швидкості обертів);
- виведе середнє значення швидкості обертів двигуна та відповідне йому значення напруги

Завдання №6.3. Розробіть ВП, який виконуватиме такі дії:

- згенерує значення тиску рідини на дно ємності, що сигналізує про об’єм рідини, в діапазоні 3±2 Бар;
- згенерує покази рівня рідини в ємності 5±2 м (незалежно від значень тиску);
- переведе отримані значення з числового формату в строковий;
- запише послідовно масив даних у вигляді двох стовпців *P*[Бар] і *L*[м] в файл *.*txt*;

- вилучить дані з текстового файлу;
- зіставить значенням елементів масиву показань тиску значення елементів масиву показань рівнів рідини (перегрупування масиву показань рівнів рідини);
- виведе максимальне значення абсолютного відхилення рівня рідини та відповідне йому значення абсолютного відхилення тиску на дно ємності.

Завдання №6.4. Розробіть ВП, який виконуватиме такі дії:

- згенерує значення опору, що працює як фотоелемент, в діапазоні 400±200 Ом;
- згенерує покази сили світла в діапазоні 1200±300 Лм (незалежно від значень опору);
- переведе отримані значення з числового формату в строковий;
- запише послідовно масив даних у вигляді двох стовпців *R*[Ом] і *L*[Лм] в файл **.txt*;
- вилучить дані з текстового файлу;
- зіставить значенням елементів масиву показань опорів значення елементів масиву показань сили світла (перегрупування масиву показань сили світла);
- виведе максимальне значення відносного відхилення сили світла та відповідне йому значення відносного відхилення опору.

Завдання №6.5. Розробіть ВП, який виконуватиме такі дії:

- згенерує значення сили стуму в діапазоні 4±2 А;
- згенерує покази швидкості обертів двигуна в діапазоні 300±100 об/хв (незалежно від значень опору);
- переведе отримані значення з числового формату в строковий;
- запише послідовно масив даних у вигляді двох стовпців *I*[А] і *W*[об/хв] в файл **.txt*;
- вилучить дані з текстового файлу;
- зіставить значенням елементів масиву показань швидкості обертів

двигуна значення елементів масиву показань сили струму (перегрупування масиву показань швидкості обертів двигуна);

- виведе на графіку залежність швидкості обертів двигуна від сили струму живлення.

Завдання №6.6. Розробіть ВП, який виконуватиме такі дії:

- згенерує значення опору, що працює як термопара, в діапазоні 300±100 кОм;
- згенерує покази тиску в котлі в діапазоні 5±3 Бар (незалежно від значень опору);
- переведе отримані значення з числового формату в строковий;
- запише послідовно масив даних у вигляді двох стовпців *R*[Ом] і *P*[Бар] в файл **.txt*;
- вилучить дані з текстового файлу;
- зіставить значенням елементів масиву показань опорів значення елементів масиву показань тиску (перегрупування масиву показань тиску);
- виведе на графіку залежність тиску від опору.

Контрольні запитання

6.1) Які типи даних в *LabVIEW* пов'язані з роботою із строковими змінними?

6.2) Основні функції роботи зі строками?

6.3) Послідовність дій при записі даних у файл?

6.4) Послідовність дій при зчитуванні даних з файлу?

6.5) Які операції з файлами можна виконувати в *LabVIEW*?

6.6) Чи можна за допомогою засобів *LabVIEW* створити текстовий редактор на зразок редактора «Блокнот» з *Windows*?

6.7) Який тип даних в *LabVIEW* відображається рожевим кольором?

ЛАБОРАТОРНА РОБОТА № 7
РОБОТА З ФУНКЦІЯМИ ФАЙЛОВОГО ВВЕДЕННЯ/ВИВОДУ В СЕРЕДОВИЩІ LABVIEW

Мета роботи: Навчитися працювати з файлами, розраховувати значення функції на заданому інтервалі та програмувати роботу віртуального приладу (ВП) в декількох режимах.

Стислі теоретичні відомості

Приклад 7.1

Розглянемо запис даних у файл формату електронної таблиці.

Створіть блок-діаграму, показану на рис. 7.1.

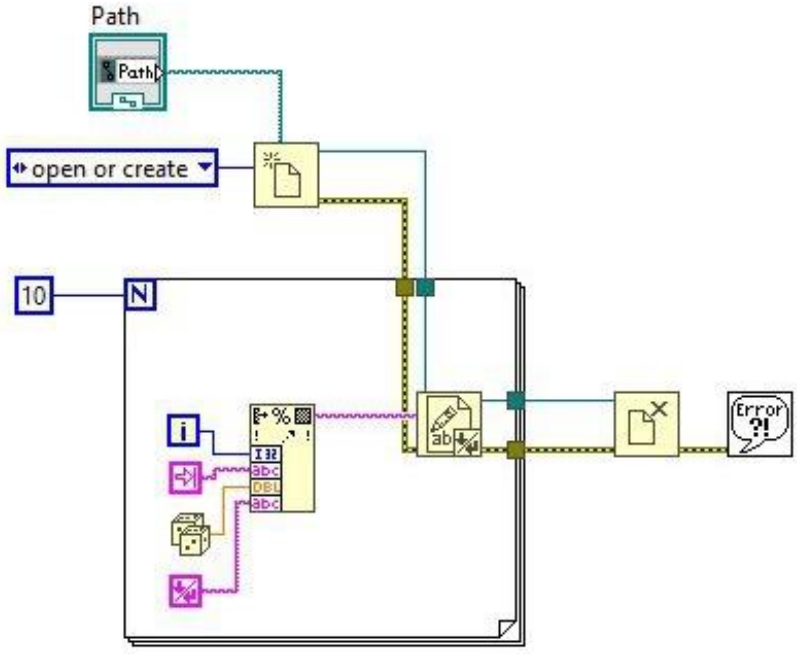


Рисунок 7.1 — Блок-діаграма ВП до прикладу 7.1

Виберіть цикл *For*, розташований в палітрі *Functions* > *Structures*. Задайте число ітерацій, рівне десяти. Виберіть функцію *Write Text File*, розташовану в палітрі *Functions* > *File* I/O і помістіть її всередину циклу *For*. Ця функція записує об'єднаний рядок у файл.

Помістіть всередину циклу *For* функцію *Format Into String*, розташовану в палітрі *Functions* > *String*, і збільште число входів до

чотирьох. Вихід функції *resulting string* з'єднайте з входом *data* функції *Write Text File*.

На перший вхід подайте значення лічильника ітерацій циклу. На другий − константу *Tab constant*, розташовану в палітрі *Functions* > *String*. На третій − значення генератора випадкових чисел і на четвертий − константу *End of Line constant*, розташовану в палітрі *Functions* > *String*.

Помістіть на блок-діаграму підпрограму ВП *Open*/*Create*/*Replace File VI*, розташовану в палітрі *Functions* > *File* I/O. Цей ВП виводить на екран діалогове вікно для створення файлу.

Створіть і з'єднайте з входом *function* константу *open or create* і строкову константу *Enter filename* з входом *prompt*. У палітрі *Functions* > *File* I/O виберіть функцію *Close File*. Ця функція закриє файл.

Підпрограми ВП і функції низького рівня містять інформацію про помилки. Для їх обробки використовуються підпрограми обробки помилок, такі як *Simple Error Handler VI* (ВП Простий обробник помилок), розташований в палітрі *Functions* > *Time* & *Dialog*.

Поля введення error in і виведення error out інформації про помилки використовуються в кожному ВП для обміну інформацією про помилки між ВП.

Перейдіть на лицьову панель і запустіть ВП. З'явиться діалогове вікно *Enter Filename*. У діалоговому вікні введіть ім'я файлу *Random num.txt* і натисніть на кнопку *Save* або *OK*. Відкрийте файл *Random num.txt* одним з текстових редакторів *Windows*, наприклад *Notepad* або *Wordpad*.

Закрийте текстовий редактор і поверніться в *LabVIEW*.

Приклад 7.2

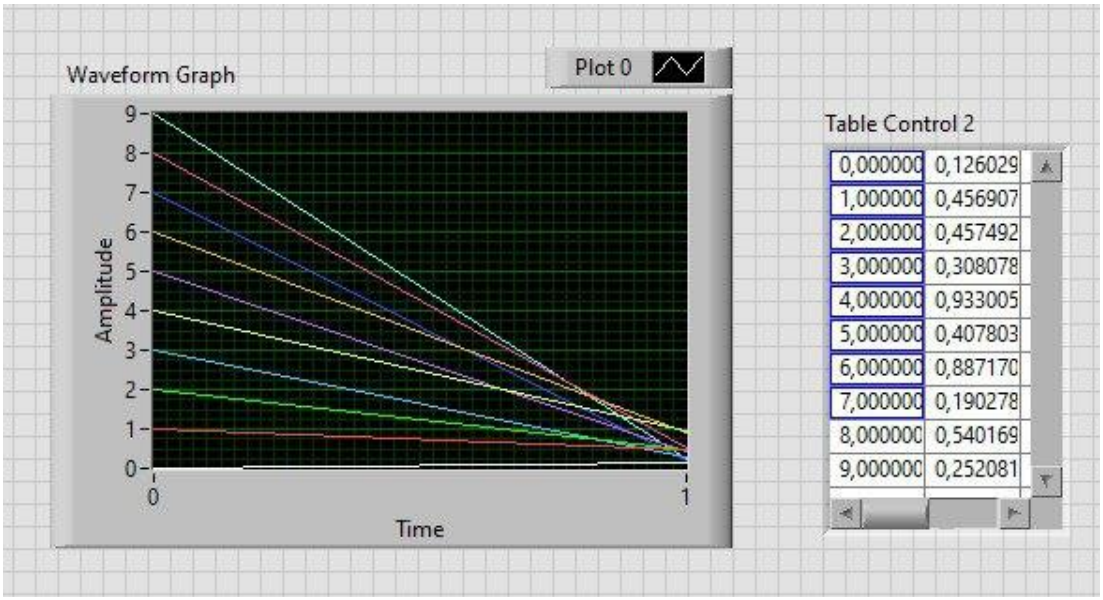
Розглянемо зчитування даних з файлу і представлення їх у вигляді графіків і таблиці.

Створіть новий додаток. Встановіть на передню панель таблицю *Table*: *Controls* > *List* & *Table* > *Table*, потім перетворіть в індикатор. При натисненні правої клавіші миші на об'єкті виберіть пункт *Change to*

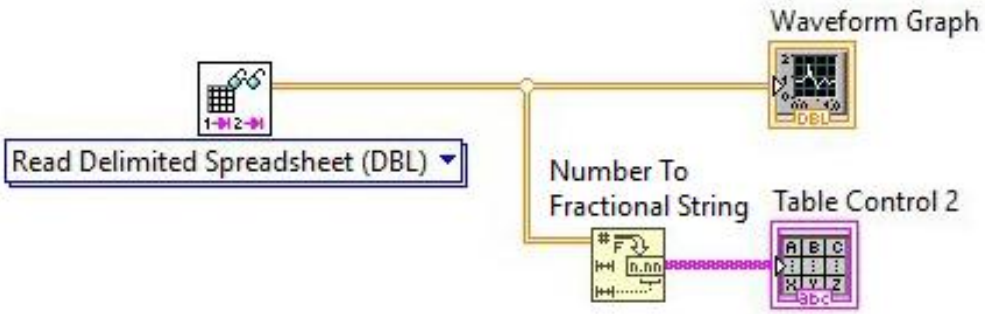
Indicator. Встановіть елемент *Waveform Graph* (передня панель): *Control* > *Graph*. У вікні редагування діаграм встановіть наступні компоненти:

- *Number To Fractional String* (блок діаграма): (*Functions* > *String* > *String / Number Conversation*).
- *Read Delimited Spreadsheet.vi* (блок діаграма): (*Functions* > *File* I/O).

Блок діаграма і передня панель програми зчитування представлені на рис. 7.2.



а)



б)

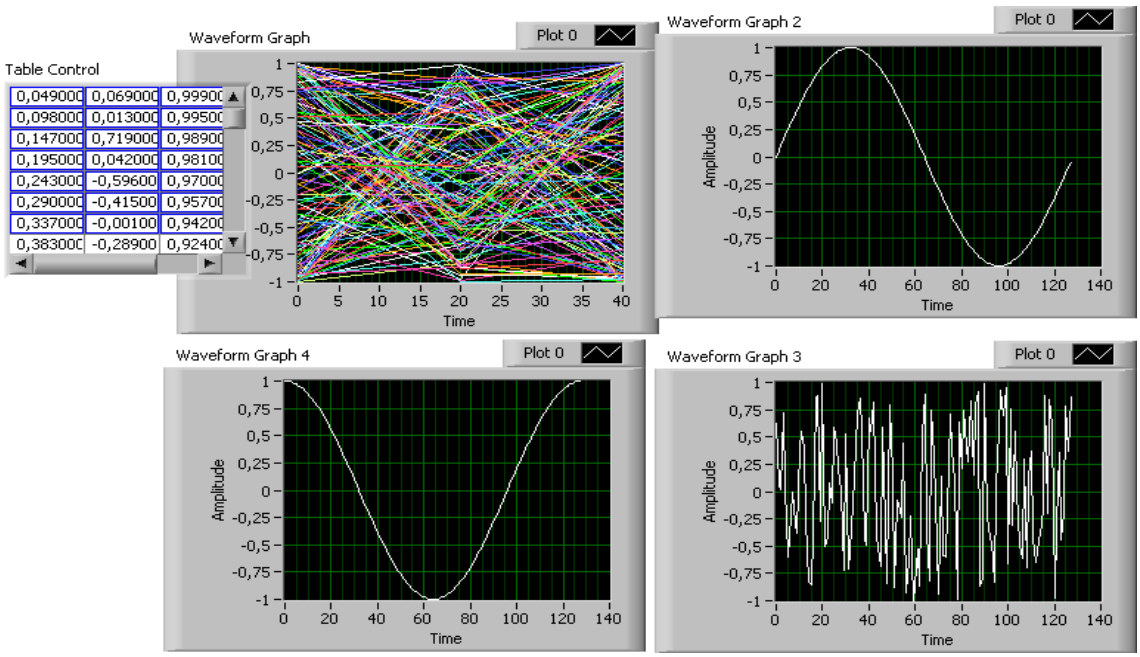
Рисунок 7.2 — ВП програми зчитування до прикладу 7.2:

а) передня панель; б) блок-діаграма

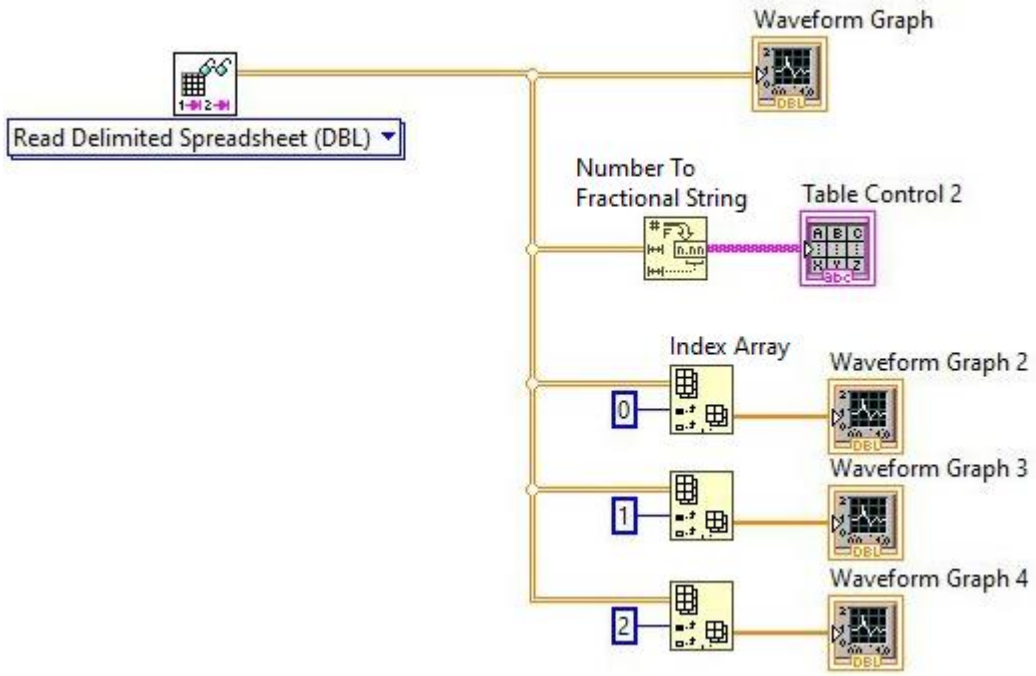
Виконайте необхідні з'єднання і запустіть програму. Після запуску, у з'явившомуся вікні виберіть файл даних (створений в попередньому прикладі).

Приклад 7.3

Модифікуємо створену програму. Блок діаграма і передня панель програми зчитування представлені на рис. 7.3.



а)



б)

Рисунок 7.3 — ВП до прикладу 7.3: а) передня панель; б) блок-діаграма

Розділимо двовимірний масив на три одновимірних і побудуємо графіки кожного сигналу окремо. Для цього скористаємося вже відомою

функцією роботи з масивами: *Functions* > *Array* > *Index Array*. З її допомогою виділимо стовпці масиву.

Додайте три *Waveform Graph*. У вікно редагування діаграм встановіть три *Index Array*, з'єднайте *Waveform Graph* з вихідним масивом і створіть константи (0, 1, 2), що визначають номер стовпця масиву, а значить і тип залежності.

Після запуску, у вікні, що з'явилося, необхідно вибрати файл даних (створений раніше).

Приклад 7.4

Створимо віртуальний прилад (ВП), що дозволяє розрахувати значення функції $f(x)$ на заданому інтервалі із заданим кроком і відображувати значення функції на графіку. ВП повинен підтримувати роботу в двох режимах.

У першому режимі має бути передбачена можливість будувати графік функції на заданому інтервалі і зберігати його у файл у форматі *LabVIEW Waveform*. У другому режимі необхідно організувати зчитування раніше записаних даних з файлу, відображення їх на графіку і конвертацію даних в звичайний текстовий файл.

Створіть новий ВП як на блок-схемі (рис. 7.4). Помістіть структуру *While Loop* (*Functions* > *Structures* > *While Loop*) і розтягніть її на весь екран. Створіть нескінчений цикл. Для цього помістіть булеву константу *True* (*Functions* > *Boolean* > *True Constant*) і з'єднайте її з умовою виходу з циклу.

Помістіть структуру *Case* (*Functions* > *Structures* > *Case*) усередині *While Loop* і розтягніть її так, щоб вона займала значну її частину.

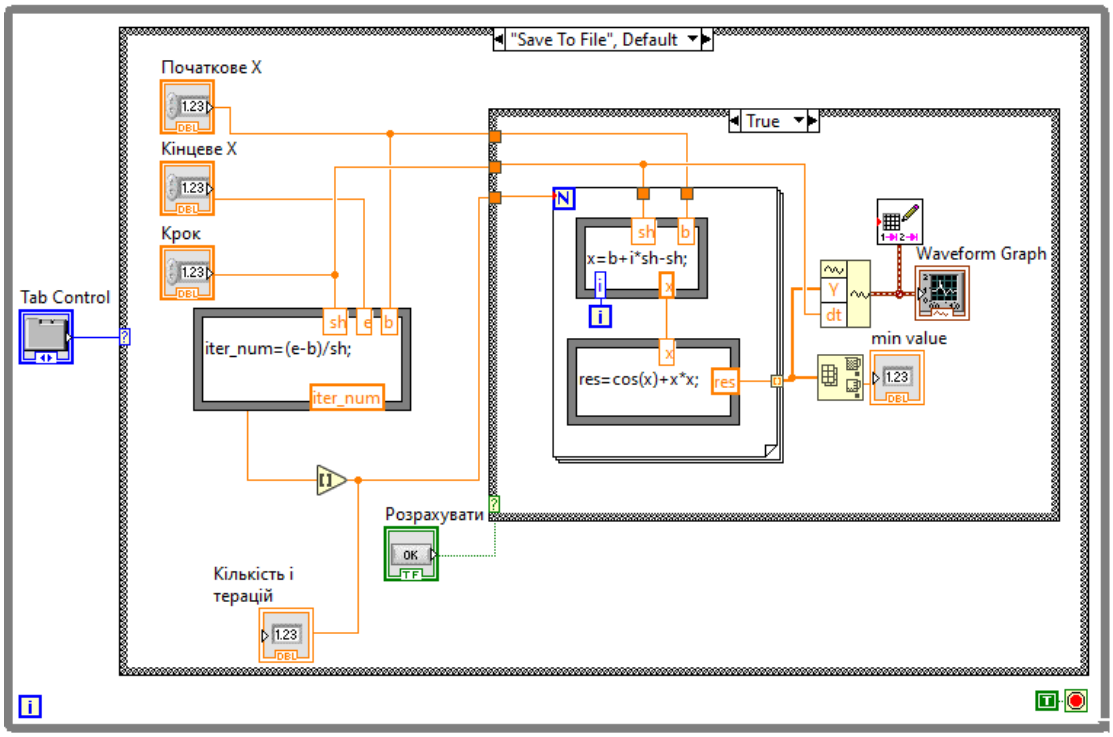


Рисунок 7.4 — Блок-діаграма до прикладу 7.4

На лицьовій частині ВП (рис. 7.5) помістіть елемент *Tab Control* (*Controls* > *Array & Cluster* > *Tab Control*), розтягніть його на весь екран і назвіть «Режими роботи». Натисніть правою клавішею миші на закладці вгорі та в контекстному меню виберіть пункт *Add page after*. Назвіть першу закладку «*Save to file*», а другу – «*Load from file*».

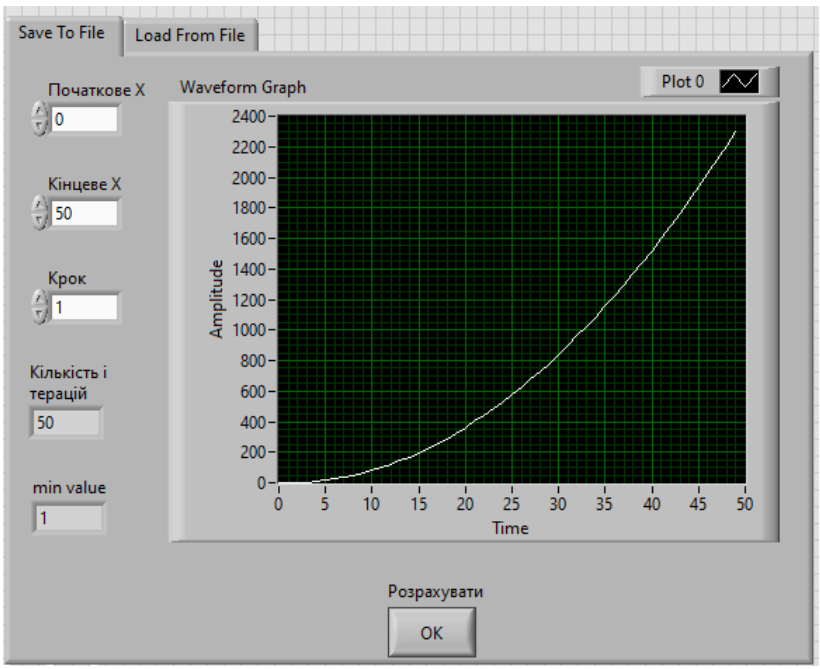


Рисунок 7.5 — Передня панель до прикладу 7.4

На блок-схемі ВП (рис. 7.4) з'єднайте елемент *Tab Control* з точкою умови структури *Case*. Елемент *Tab Control* має бути усередині циклу. У структурі *Case* повинні з'явитися дві закладки «*Save to file*» і «*Load from file*», відповідні закладкам *Tab Control*.

На лицьовій частині ВП (рис. 7.5): створіть інтерфейс для першого режиму роботи. Для цього виберіть першу закладку (*Save to file*). Помістіть три елементи *Digital Control* (*Controls > Numeric > Digital Control*), назвіть їх відповідно: «Початкове значення *x*», «Кінцеве значення *x*» і «Крок». Натисніть правою клавішею миші на кожному елементі і в контекстному меню виберіть представлення у вигляді числа *DBL* (дійсне з плаваючою точкою подвійної точності).

Помістіть елемент *Digital Indicator* (*Controls > Numeric > Digital Indicator*) і назвіть його «Кількість ітерацій». Виберіть представлення у вигляді 32-розрядного цілого числа (*I*132).

Помістіть елемент *Waveform Graph* (*Controls > Graph > Waveform Graph*). Використовуючи контекстне меню встановіть автомасштабування по осях *XY*. Помістіть елемент *OK Button* (*Controls > Boolean > OK Button*) і привласніть йому назву «Розрахувати».

На блок схемі ВП (рис. 7.4) перевірте, щоб структура *Case* була встановлена на закладці «*Save to file*». Переконайтеся, що піктограми всіх поміщених в попередньому пункті елементів управління та індикації знаходяться усередині структури *Case*. Помістіть елемент *Round Te Nearest* (*Functions > Numeric > Round To Nearest*), що дозволяє округлювати числа до найближчого цілого.

Використовуючи значення елементів керування «Початкове значення *x*», «Кінцеве значення *x*» і «Крок», розрахуйте необхідну кількість ітерацій. Для цього слід скористатися структурою *Formula Node* (*Functions > Structures > Formula Node*), що дозволяє використати текстові формули, як у всіх поширених мовах програмування.

Створіть на рамці структури три точки входу і одну точку виходу. Для цього скористайтеся контекстним меню. Назвіть їх відповідно «*b*», «*e*», «*sh*» і «*iter_num*». З'єднайте входи з відповідними елементами *Digital Control*, а вихід з елементом *Round To Nearest*, як показано на рис. 7.4.

Усередині структури *Formula Node* напишіть текстом формулу з використанням створених змінних (*iter_num*=(*e*-*b*)/*sh*;). Всі формули повинні закінчуватися точкою з комою. З'єднайте вихід елементу *Round To Nearest* з елементом «Кількість ітерацій» (*Digital Control*). Помістіть усередині структури *Case* ще одну таку ж структуру.

З'єднайте елемент *OK Button* («Розрахувати») з точкою умови структури. Встановіть закладку *True*. Помістіть усередині створеної структури *Case* структуру *For Loop* (*Functions* > *Structures* > *For Loop*). Задайте кількість ітерацій. Для цього з'єднайте піктограму *N* структури *For Loop* з виходом елементу *Round To Nearest*, як показано на рис. 7.4. У структуру *For Loop* додайте елемент *Formula Node.*

Аналогічно створіть формулу для розрахунку значень *x* (*x=b+i*sh-sh*;). Змінні *b, sh, i* − входи, а *x* − вихід. З'єднайте змінні, як показано на рис. 7.4. На блок-схемі ВП (рис. 7.4) додайте в цикл *For Loop* ще одну структуру *Formula Node* і реалізуйте функцію *f*(*x*)=*cos*(*x*)+*x*x*.

На блок-схемі ВП (рис. 7.4) створіть точку зовнішнього з'єднання в циклі *For Loop*. Для цього з'єднаєте вихід реалізованої функції з рамкою циклу.

Якщо піктограма Waveform Graph знаходиться за рамкою внутрішньої структури Case, перемістіть її так, щоб вона була усередині. Помістіть туди ж елементи *Build Waveform* (*Functions* > *Waveform* > *Build Waveform*) і *Write Waveforms to File* (Functions > Waveform > Waveform File I/O > *Write to File*). З'єднайте ці два елементи і *Waveform Graph*, як показано на рис. 4.

Використовуючи елемент *Array Max* & *Min* (*Functions* > *Array* > *Array Max* & *Min*), відобразіть за допомогою елементу *Digital Indicator* мінімальне значення вектора (рис. 7.4).

Перш ніж переходити до наступного розділу, переконаєтеся, що закладка *False* внутрішньої структури *Case* порожня.

Опишіть другий режим роботи ВП. На лицьовій частині ВП (рис. 7.6) перейдіть на закладку «*Load from file*». Помістіть елемент *Waveform Graph* та два елементи *OK Button* («Конвертувати в текстовий файл» і «Показати графік»).

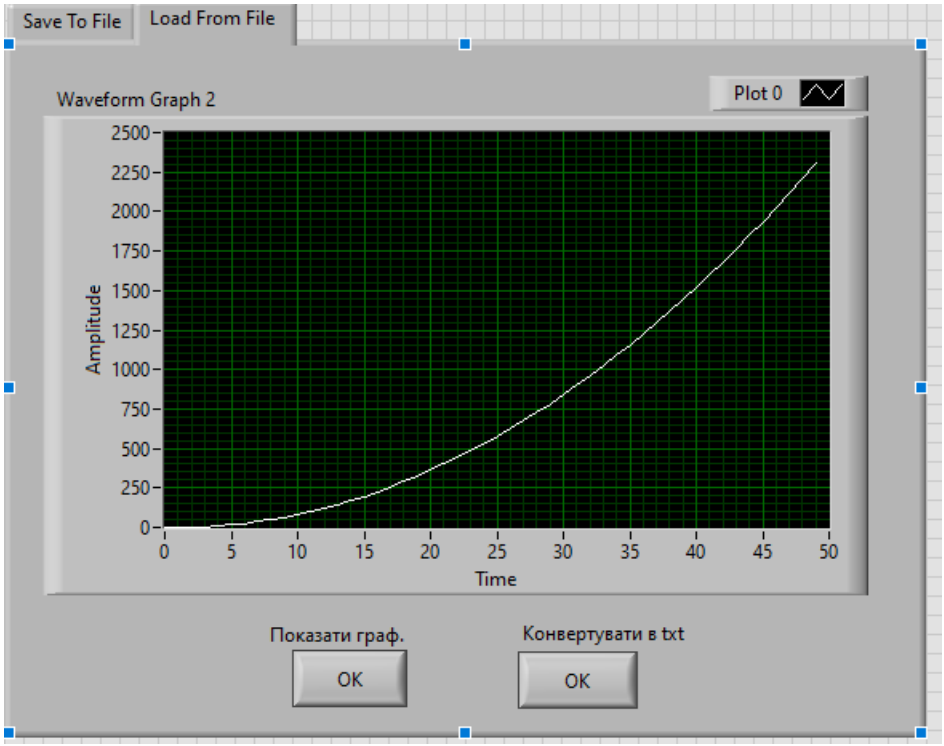


Рисунок 7.6 — Передня панель другої заладки «*Load from file*» ВП до прикладу 7.4

У блок схемі ВП (рис. 7.7) перейдіть на закладку «*Load from file*» зовнішньої структури *Case*. При необхідності перемістіть додані раніше піктограми елементів на цю закладку (вони можуть бути автоматично на неї поміщені). Помістіть ще одну структуру *Case* усередині першої.

З'єднайте кнопку «Показати графік» з точкою умови цієї структури. У внутрішню структуру *Case* помістіть елемент *Read Waveforms From File* (*Functions* > *Waveform* > *Waveform File* I/O > *Read Waveforms From File*). Перемістіть піктограму графіка *Waveform Graph* в цю ж структуру. З'єднайте їх, як показано на рис. 7.7. Аналогічно додайте ще одну структуру *Case* і з'єднайте її з кнопкою «Конвертувати в текстовий файл» (рис. 7.7).

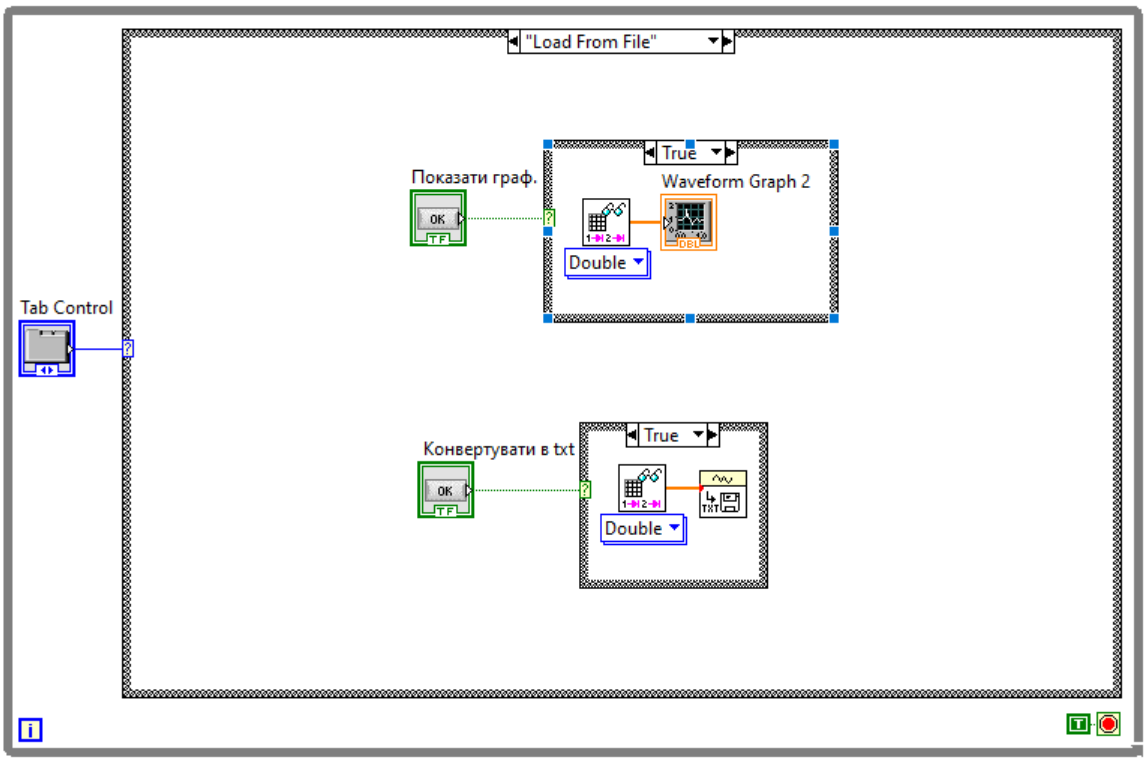


Рисунок 7.7 — Передня панель другої заладки «*Load from file*» ВП до прикладу 7.4

Додайте в структуру *Case* елементи *Read Waveform From File* і *Export Waveforms to Spreadsheet File* (*Functions* > *Waveform* > *Waveform* I/O) З'єднайте їх як показано на рис. 7.7. Закладки *False* обох структур *Case* мають бути порожні.

Перевірте працездатність ВП: наявність файлів, правильність відображення графіків, правильність розрахунків.

Хід роботи

7.1) Виконати приклади для створення віртуальних приладів.

7.2) Отримати і ознайомитися з індивідуальним завданням.

7.3) Сформувати необхідні інструменти.

7.4) Налагодити програму і представити викладачеві для перевірки.

7.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

- звітність про виконання прикладів для створення віртуальних приладів;

- варіант і зміст індивідуального завдання;
- скріншоти складових програми і результати її виконання;
- відповіді на запитання до лабораторно роботи.

7.6) Представити звіт на захист.

Варіанти завдань

Спільна частина для всіх завдань: Створити віртуальний прилад (ВП), що дозволяє експериментальні дані на заданому інтервалі із заданим кроком відображувати на графіці. ВП повинен підтримувати роботу в двох режимах.

У першому режимі має бути передбачена можливість будувати графік функції на заданому інтервалі і зберігати його у файл у форматі *LabVIEW Waveform*. У другому режимі необхідно організувати зчитування раніше записаних даних з файлу, відображення їх на графіці і конвертацію даних в звичайний текстовий файл.

Завдання №7.1. В якості експериментальних даних:

• згенеруйте значення опору, що працює як термопара, в діапазоні 500±100 Ом;

• згенеруйте покази температури в діапазоні 300±10 К (незалежно від значень опору);

• зіставте значенням елементів масиву показань опорів значення елементів масиву показань температури (перегрупування масиву показань температури).

Завдання №7.2. В якості експериментальних даних:

• згенеруйте значення напруги, що визначається обертами двигуна, в діапазоні 9±3 В;

• згенеруйте покази швидкості обертів двигуна в діапазоні 500±200 об/хв (незалежно від значень напруги);

• зіставте значенням елементів масиву показань напруг значення елементів масиву показань швидкості обертів (перегрупування масиву показань швидкості обертів).

Завдання №7.3. В якості експериментальних даних:

• згенеруйте значення тиску рідини на дно ємності, що сигналізує про об’єм рідини, в діапазоні 3±2 Бар;

• згенеруйте покази рівня рідини в ємності 5±2 м (незалежно від значень тиску);

• зіставте значенням елементів масиву показань тиску значення елементів масиву показань рівнів рідини (перегрупування масиву показань рівнів рідини);

Завдання №7.4. В якості експериментальних даних:

• згенеруйте значення опору, що працює як фотоелемент, в діапазоні 400±200 Ом;

• згенеруйте покази сили світла в діапазоні 1200±300 Лм (незалежно від значень опору);

• зіставте значенням елементів масиву показань опорів значення елементів масиву показань сили світла (перегрупування масиву показань сили світла);

Завдання №7.5. В якості експериментальних даних:

• згенеруйте значення сили стуму в діапазоні 4±2 А;

• згенеруйте покази швидкості обертів двигуна в діапазоні 300±100 об/хв (незалежно від значень опору);

• зіставте значенням елементів масиву показань швидкості обертів двигуна значення елементів масиву показань сили струму (перегрупування масиву показань швидкості обертів двигуна);

Завдання №7.6. В якості експериментальних даних:

• згенеруйте значення опору, що працює як термопара, в діапазоні 300±100 кОм;

• згенеруйте покази тиску в котлі в діапазоні 5±3 Бар (незалежно від значень опору);

• зіставте значенням елементів масиву показань опорів значення елементів масиву показань тиску (перегрупування масиву показань тиску).

Контрольні запитання

7.1) Особливості зчитування файлів через віртуальні прилади?

7.2) Робота з текстовими таблицями у середовищі *LabVIEW*?

7.3) Яким чином здійснюється перехід між закладками ВП?

7.4) Як застосовуються структури при роботі з файлами?

7.5) Призначення елементів роботи з файлами високого рівня в *LabVIEW*?

7.6) Елементи роботи з файлами низького рівня в *LabVIEW*?

7.7) Як здійснюється побудова графіків за даними з файлу?

7.8) Які засоби для налагодження роботи з табличними даними є в *LabVIEW*?

ЛАБОРАТОРНА РОБОТА № 8
МОДЕЛЮВАННЯ РОБОТИ ЦИФРОВИХ ПРИСТРОЇВ В СЕРЕДОВИЩІ LABVIEW

Мета роботи: моделювання роботи цифрових пристроїв в середовищі *LabVIEW*: АЦП, шифратор, дешифратор, мультиплексом та інші.

Стислі теоретичні відомості

Побудуємо модель аналого-цифрового перетворювача (АЦП) в середовищі LabVIEW

Приклад 8.1

Створіть новий документ віртуального приладу. З палітри *Controls* > *Numeric* виберіть три цифрових керівних елементи (*Digital Control*) і розмістіть їх на лицьовій панелі: Випадкова похибка АЦП, Розрядність АЦП, Крок квантування. З підменю *Array* > *Matrix* & *Cluster* палітри *Controls* виберіть два елементи *Array*. Задайте тип масиву: у першому випадку керуючий цифровий елемент (*Numeric Control*), а в другому цифровий індикатор (*Numeric Indicator)* (рис. 8.1).

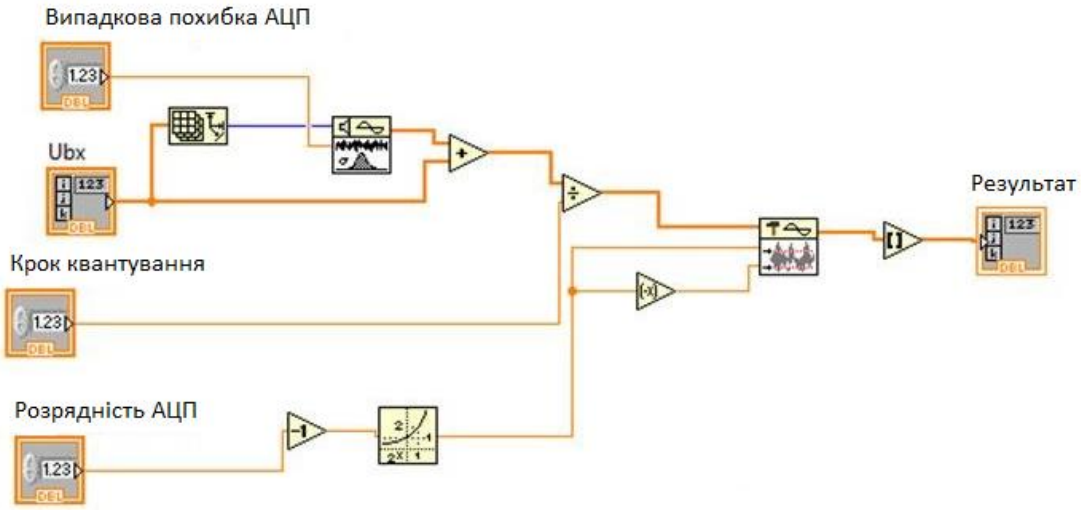
Перейдіть у вікно структурної схеми *Diagram*. Виберіть з палітри *Function* > *Signal Processing* > *Signal Generation* генератор нормального білого шуму і розмістіть на структурній схемі.

З палітри *Function* > *Signal Processing* > *Signal Operation* візьміть обмежувач амплітуди (*Y*[*i*]=*Clip*{*X*[*i*].*vi*). З палітри *Function* > *Programming* > *Numeric* виберіть такі елементи: суматор (*Add*), дільник (*Divide*), декремент (*Decrement*), інвертор (*Negate*) та *Round to Nearest*.

З палітри *Functions* > *Mathematics* > *Elementary* & *Special Functions* > *Exponental Functions* виберіть *Power of* 2 (елемент дозволяє зводити 2 в ступінь). З палітри *Functions* > *Programming* > *Array* оберіть елемент *Array Size*.

З'єднайте елементи так, як показано на структурній схемі рис 8.1.

а)



б)

Рисунок 8.1 — ВП до прикладу 8.1: а) передня панель; б) блок-діаграма

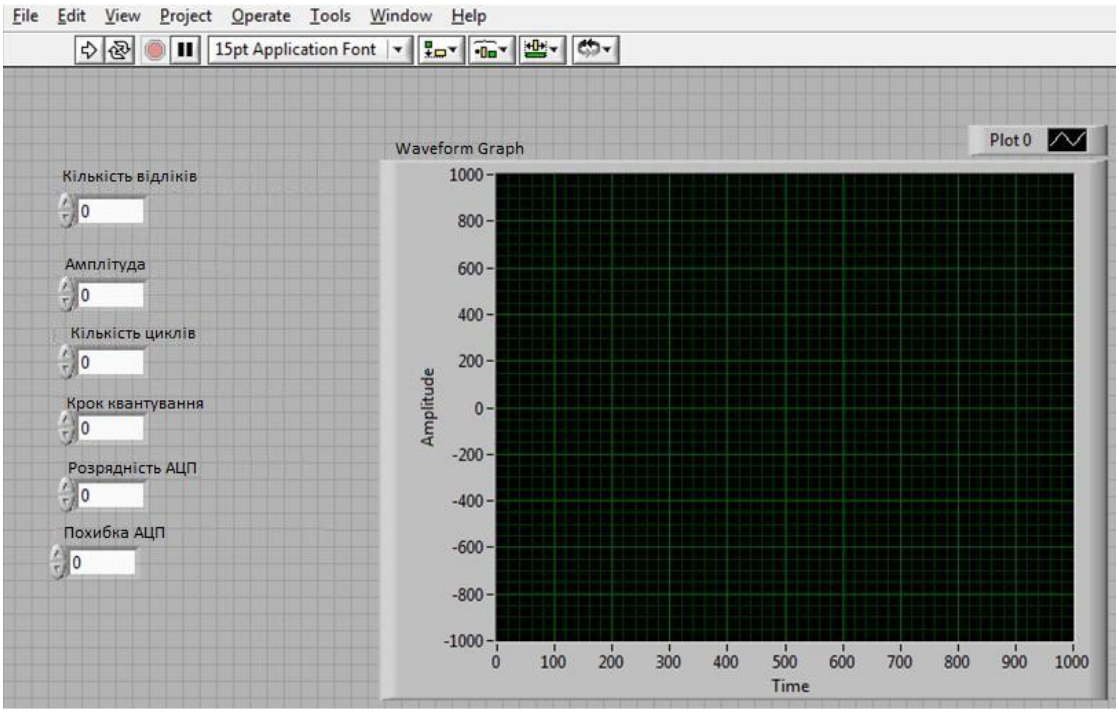
Перейдіть на вікно лицьової панелі, правою кнопкою миші натисніть по іконці віртуального приладу. У вікні меню, що з'явилося, виберіть пункт Показати термінали. Отримані термінали з'єднайте з елементами лицьової панелі.

Збережіть та протестуйте програму. Зробіть висновки.

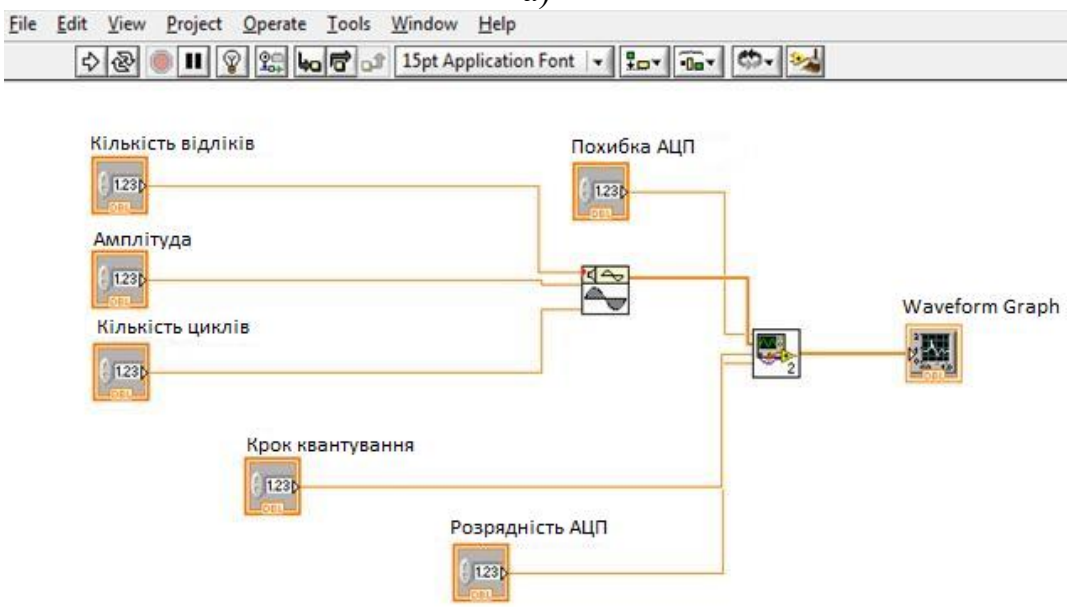
Приклад 8.2

Відкрийте новий бланк віртуального приладу. З палітри *Control* > *Numeric* по черзі виберіть 6 керуючих елементів: Кількість відліків, Амплітуда, Кількість циклів, Крок квантування, Розрядність АЦП, Похибка

АЦП. Встановіть *Waveform Graph* та інші необхідні елементи подібно до першого прикладу. Реалізація ВП представлена на рис. 8.2.



а)



б)

Риунок 8.2 — ВП до прикладу 8.2: а) передня панель; б) блок-діаграма

Задайте початкові умови: Кількість відліків = 1000; Похибка АЦП = 0 В; Амплітуда = 1 В; Кількість циклів = 1; Розрядність АЦП = 12; Крок квантування = 0,01 В (рис. 8.3). Переконайтеся у працездатності програми.

Встановіть перші тестові дані: Кількість відліків = 1000; Похибка АЦП = 0 В; Амплітуда = 1 В; Кількість циклів = 1; Розрядність АЦП = 8; Крок квантування = 0,0001 В (рис. 8.4). Зробіть висновки.

Встановіть другі тестові дані: Кількість відліків = 1000; Похибка АЦП = 0 В; Амплітуда = 1 В; Кількість циклів = 1; Розрядність АЦП = 12; Крок квантування = 0,1 В (рис. 8.5).

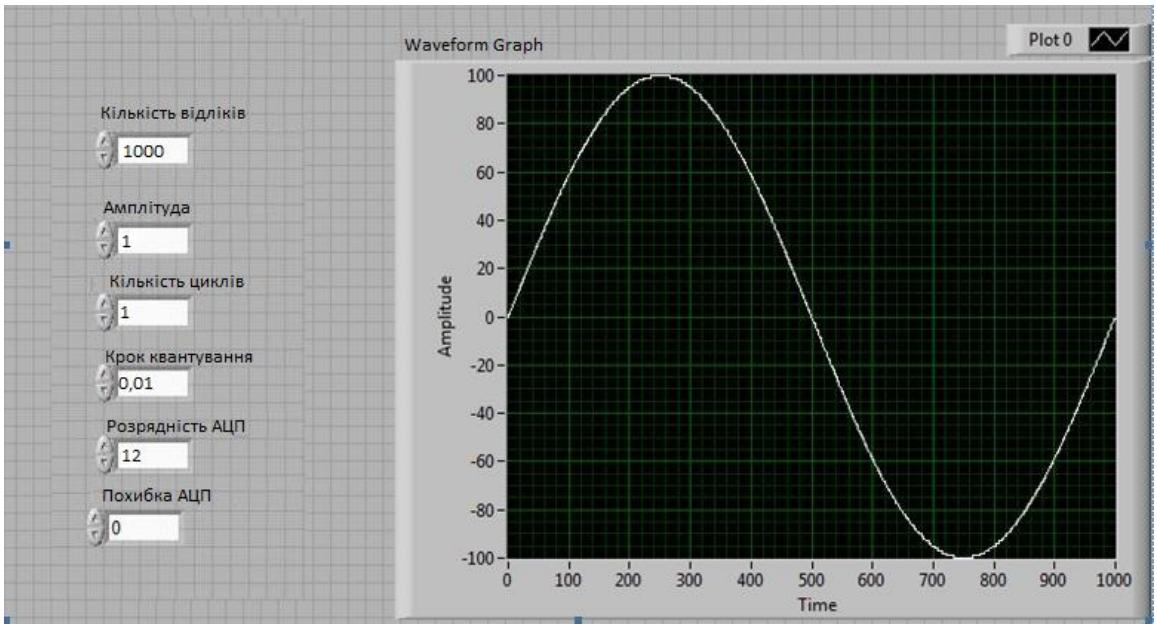


Рисунок 8.3 — Встановлення початкових умов ВП з прикладу 8.2, передня панель

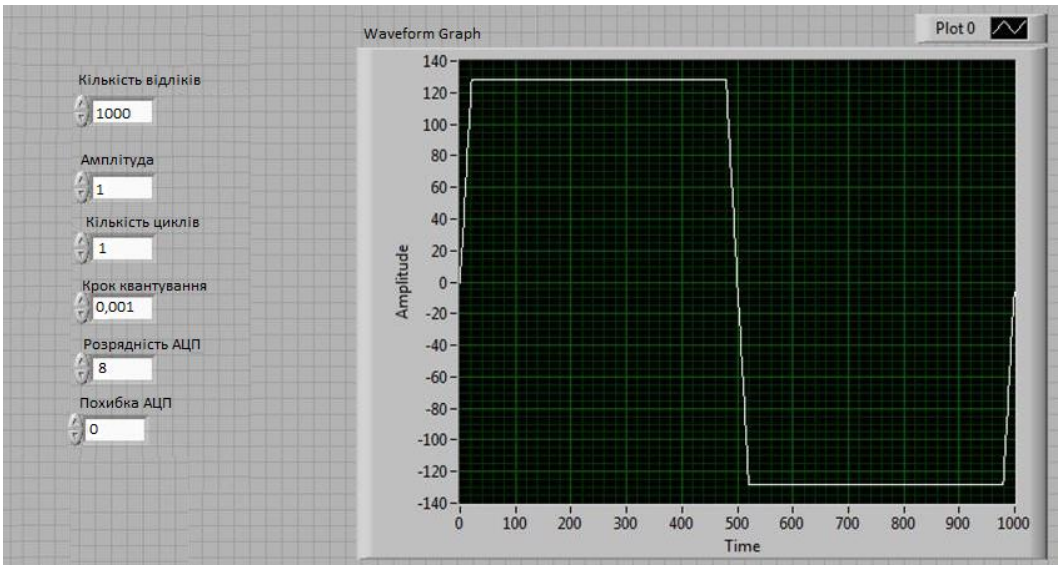


Рисунок 8.4 — Встановлення перших тестових даних ВП з прикладу 8.2, передня панель

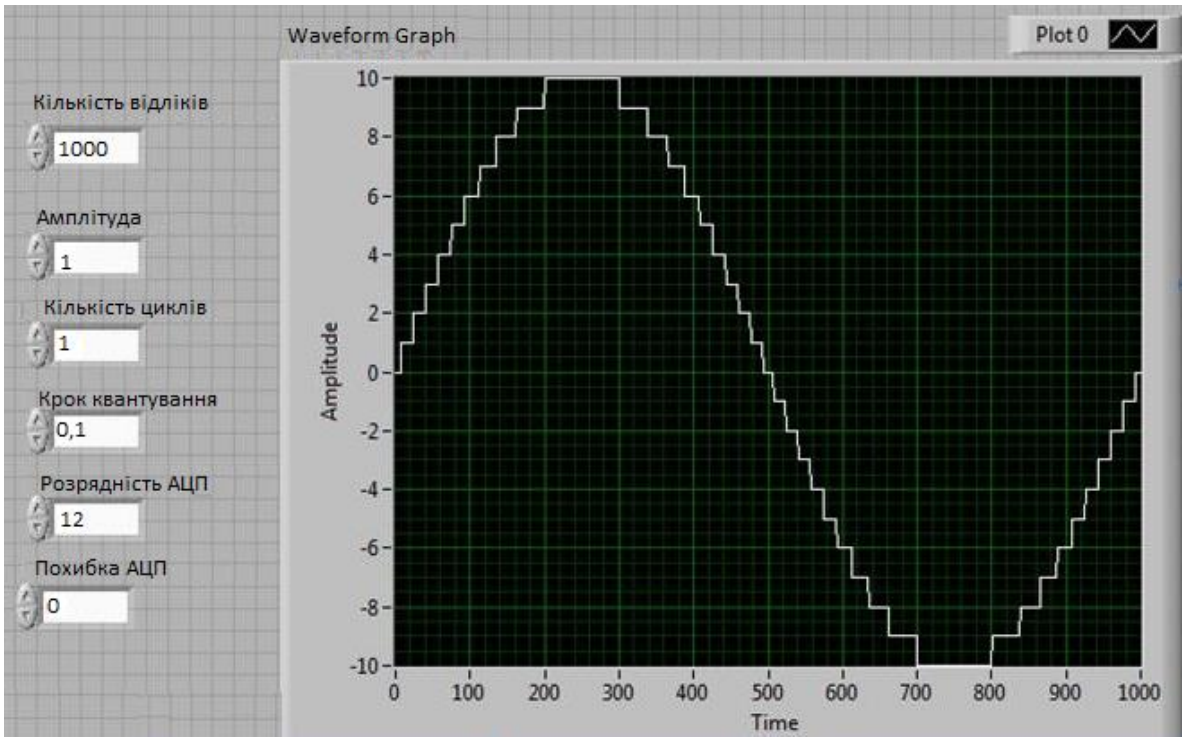


Рисунок 8.5 — Встановлення других тестових даних ВП з прикладу 8.2, передня панель

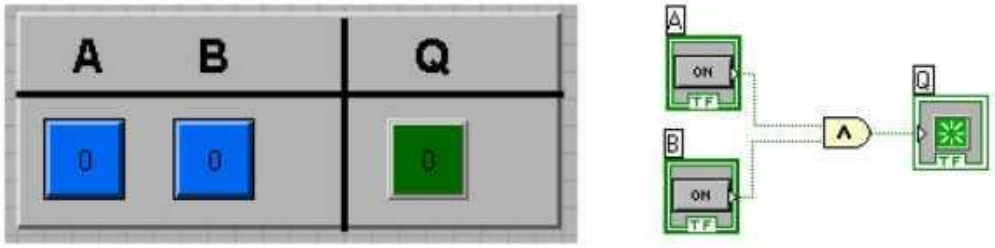
Зробіть висновки.

Побудуємо модель логічних елементів в середовищі LabVIEW

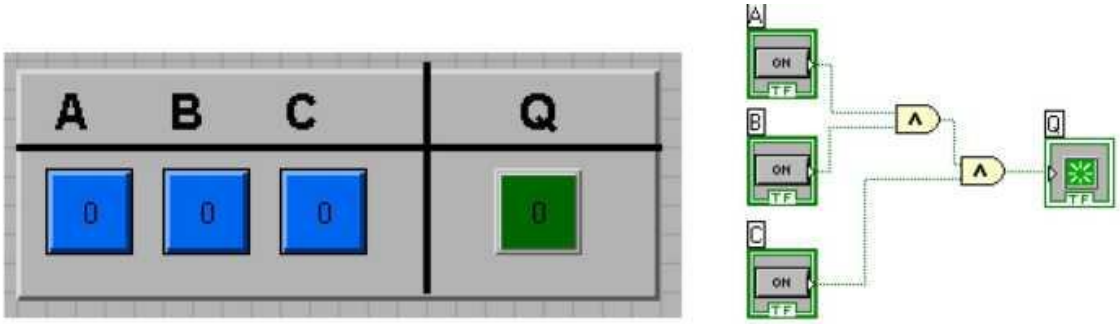
Логічні елементи − це базові блоки цифрових логічних схем. Вони можуть пропускати чи не пропускати логічний сигнал. На основі невеликої кількості основних логічних елементів (І, АБО, ЗАПЕРЕЧЕННЯ) може бути побудована величезна кількість логічних функцій.

Приклад 8.3.

Створіть віртуальний прилад, що моделює роботу елементу І. Використайте для приладу елементи з групи *Boolean*, а саме елемент *AND*. Натискуючи на дві вхідні кнопки, спостерігайте зміни вихідного індикатора (Рис. 8.6). Перевірте таблицю істинності.



а)



б)

Рисунок 8.6 — Встановлення логічного елементу, передня панель:
а) двовходовий; б) тривходовий

Аналогічним чином може бути змодельована робота інших елементів. У пакеті *LabVIEW* є всі основні двовходові логічні елементи, можна використовувати і більше входів. З двох двовходових елементів можливо побудувати віртуальний прилад, що реалізовує елемент І з трьома входами.

Підпрограми. Будь-яка програма (*VI*, віртуальний прилад) може бути використана в блок-схемі іншої програми як її складова частина. Іншими словами, вона може бути, вкладена як підпрограма, *SubVI*. Ця особливість дозволяє основній (головній програмі) бути модульною, легко читаною і простішою для розуміння.

Для вставки раніше розробленого *VI* (*SubVI*) в програму більш високого рівня необхідно використати опції *Select VI* у палітрі *Function* (рис. 8.7). У відповідь на запит діалогового вікна необхідно вибрати файл підпрограми і встановити її на діаграмі основної програми (рис. 8.8).

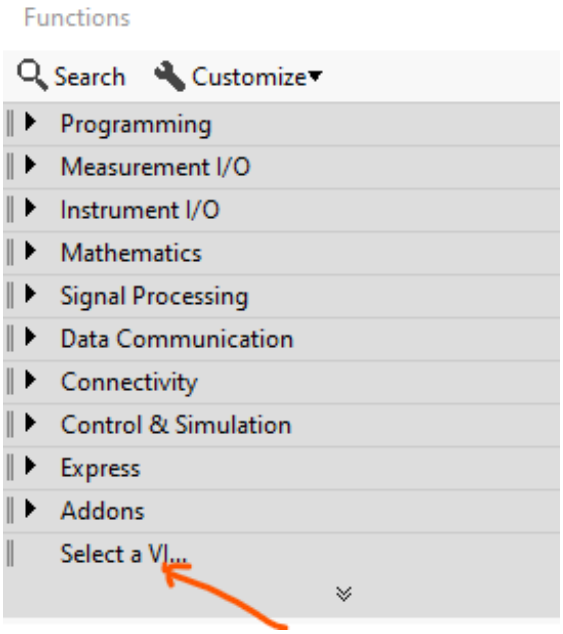


Рисунок 8.7 — Опції *Select VI* у палітрі *Function*

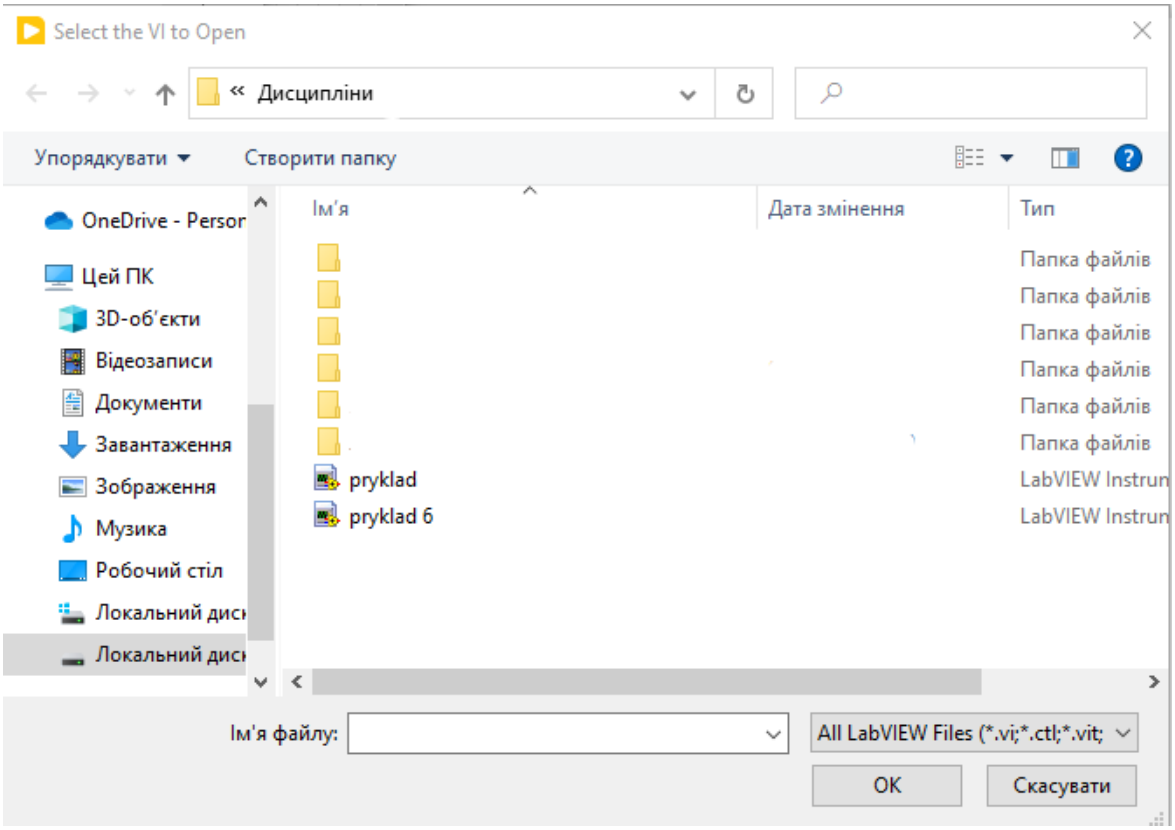


Рисунок 8.8 — Вибір файлу підпрограми

Головна програма може мати безліч викликів підпрограми. На діаграмі головної програми підпрограма з'явиться у вигляді «квадрата» із знаком, заданим за умовчуванням. За допомогою подвійного натискання можна відкрити цю підпрограму, а у разі потреби і провести деякі налаштування.

Приклад 8.4.

Створимо функціональну підпрограму раніше створеної програми, що моделює роботу логічного елементу І.

Зовнішній вигляд лицьової панелі і структурної схеми підпрограми показано на рис. 8.6. Просте включення «квадрата» підпрограми нічого не дасть головній програмі, немає можливості щось передати і щось отримати назад. Щоб підпрограма отримувала дані з основної програми і передавала їх назад в основну програму необхідно виконати певні дії.

Спочатку відкрийте лицьову панель підпрограми і, розмістивши курсор в правому верхньому кутку, на іконці, викличте контекстне меню (рис. 8.9). Вибравши в цьому меню пункт «Edit Icon», можна відредагувати зовнішній вигляд іконки створюваної підпрограми (рис. 8.10).

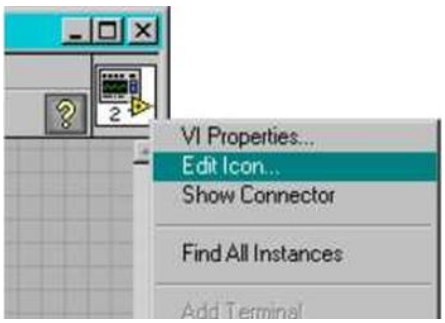


Рисунок 8.9 — Виклик контекстного меню

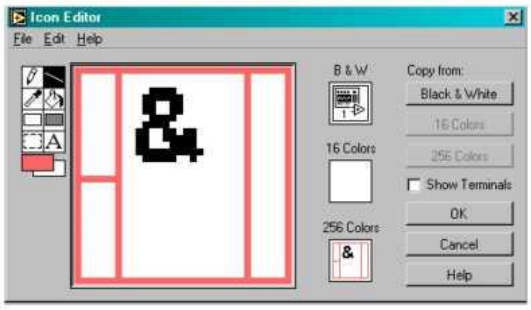


Рисунок 8.10 — Редагування зовнішнього вигляду іконки створюваної підпрограми

Далі знову слід викликати контекстне меню (рис. 8.9) і вибрати пункт «Show Connector». Через цей пункт відкривається доступ до параметрів підпрограми − конекторів або з'єднувачів, елементів взаємодії підпрограми із зовнішніми елементами. Конектор передає і приймає дані від затребуваної програми. Відразу після вибору цього пункту вигляд іконки програми зміниться (рис. 8.11), а курсор отримає вигляд сполучної котушки.

Рис. 11

Рисунок 8.11 — Змінена іконка програми

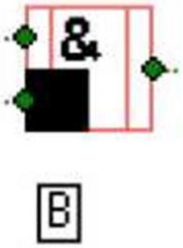


Рисунок 8.12 — Вигляд підпрограма, після її установки в головну програму

LabVIEW забезпечить появу конекторів − квадрати і прямокутник − по числу елементів управління та індикації на лицьовій панелі приладу. Тепер необхідно встановити відповідності між конектором і елементом управління або індикації на лицьовій панелі. Для цього необхідно вказати курсором-котушкою елемент на лицьовій панелі і натиснути по конектору-квадрату, за яким він буде закріплений.

Ознакою з'єднання буде зміна кольору цих квадратів. У цьому прикладі квадрати зліва − це логічні сигнали *A* і *B*, прямокутник справа − результат логічного множення *Q*. Підпрограма, після її установки в головну програму, дасть можливість бачити вхідні і вихідні змінні (рис. 8.12).

При дуже великій кількості підпрограм їх доцільно об'єднати в бібліотеки, файли з розширенням «*LLB*». Це можна зробити на етапі збереження програми. У діалозі збереження необхідно вибрати кнопку «*New VI Library*», вказавши після натиснення цієї кнопки ім'я нової бібліотеки, а далі зберегти поточний файл в створеній бібліотеці.

Для обслуговування бібліотек може бути використаний пункт меню *LabVIEW* «*LabVIEW VI library manager*».

Логічні пристрої розділяють на два класи: комбінаційні і послідовні. Пристрій називають комбінаційним, якщо його вихідні сигнали в деякий момент часу однозначно визначаються вхідними сигналами, що мають

місце у цей момент часу. Інакше пристрій називають послідовним або кінцевим автоматом (цифровим автоматом, автоматом з пам'яттю).

У послідовних пристроях обов'язково є елементи пам'яті. Стан цих елементів залежить від передісторії надходження вхідних сигналів. Вихідні сигнали послідовних пристроїв визначаються не лише сигналами, що є на входах в певний момент часу, але і станом елементів пам'яті. Таким чином, реакція послідовного пристрою на певні вхідні сигнали залежить від передісторії його роботи.

Дешифратор. Дешифратором називається комбінаційний пристрій, що перетворює n-розрядний двійковий код в логічний сигнал, що з'являється на відповідному виході, десятковий номер якого відповідає двійковому коду. Число входів і виходів в так званому повному дешифраторі зв'язані співвідношенням $k=2^n$, де n − число входів, а k − число виходів.

Роботу дешифратора з трьома входами і вісьма виходами можна представити таблицею істинності (рис. 8.13).

X0	X1	X2	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Рисунок 8.13 — Таблиця істинності

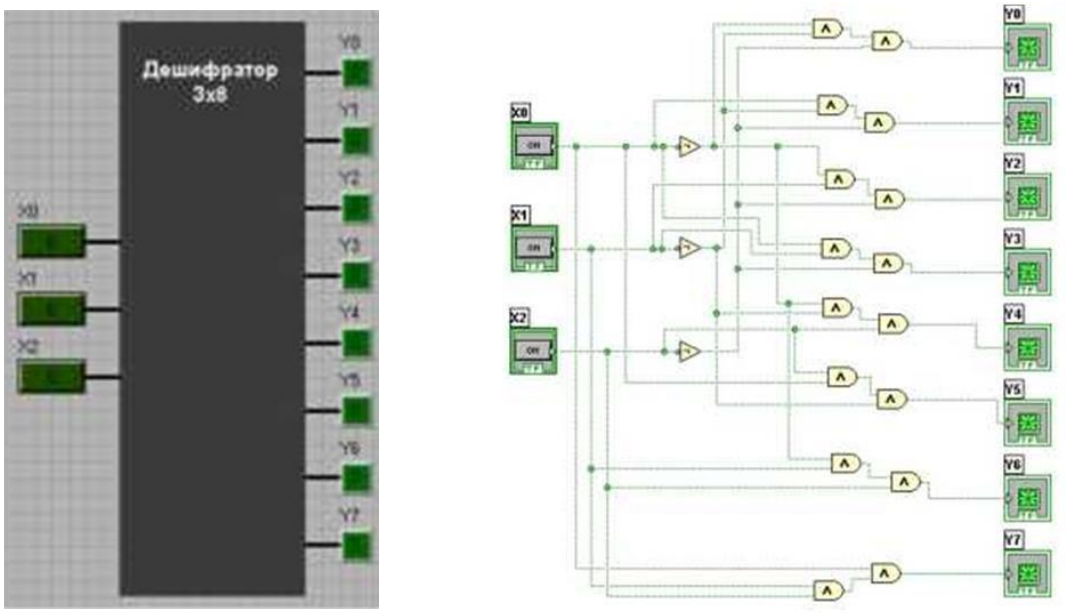
Приклад 8.5

Створимо віртуальний дешифратор в *LabVIEW*.

Розташуйте на лицьовій панелі три перемикачі і вісім світлодіодних індикаторів. Перемикачі моделюватимуть вхідний цифровий код, що поступає на дешифратор, а світлодіодні індикатори − вихідний сигнал з дешифратора.

Реалізація структурної схеми дешифратора можлива декількома способами. Перший спосіб − створити структурну схему цього приладу,

грунтуючись на базових логічних елементах, вивчених у попередніх лабораторних роботах і таблиці істинності цього пристрою (рис. 8.14).



а) б)

Рисунок 8.14 — Дешифратор, на базових логічних елементах: а) передня панель; б) блок-діаграма

Другий спосіб − при створенні структурної схеми приладу використати керуючі структури *LabVIEW* (*Case Structure*) (рис. 8.15). Структура *LabVIEW* Варіант (*Case Structure*) − управляє виконанням одного з двох або більше фрагментів коду і при виборі за умови аналогічна операторові «*If-Then-Else*» текстових мов програмування, а при виборі за значенням числової або строкової змінної аналогічна операторові *Case*.

Структура може мати усередині одну або більше піддіаграм-умов, які працюватимуть при виконанні заданих користувачем умов.

На структурній схемі, представленій на рис. 8.15, сигнал з вхідних терміналів за допомогою функціонального вузла *Boolean to* 0,1 (рис. 8.16, а) конвертується в числовий тип, далі за допомогою вузла *Multiply* (рис. 8.16, б) помножується на вагову константу біта вхідного сигналу, і в блоці *Compound Arithmetics* (рис. 8.16, в) складається. На вхід структури Варіант

подається ціле двійкове число, яке закодоване бітами вхідних терміналів приладу.

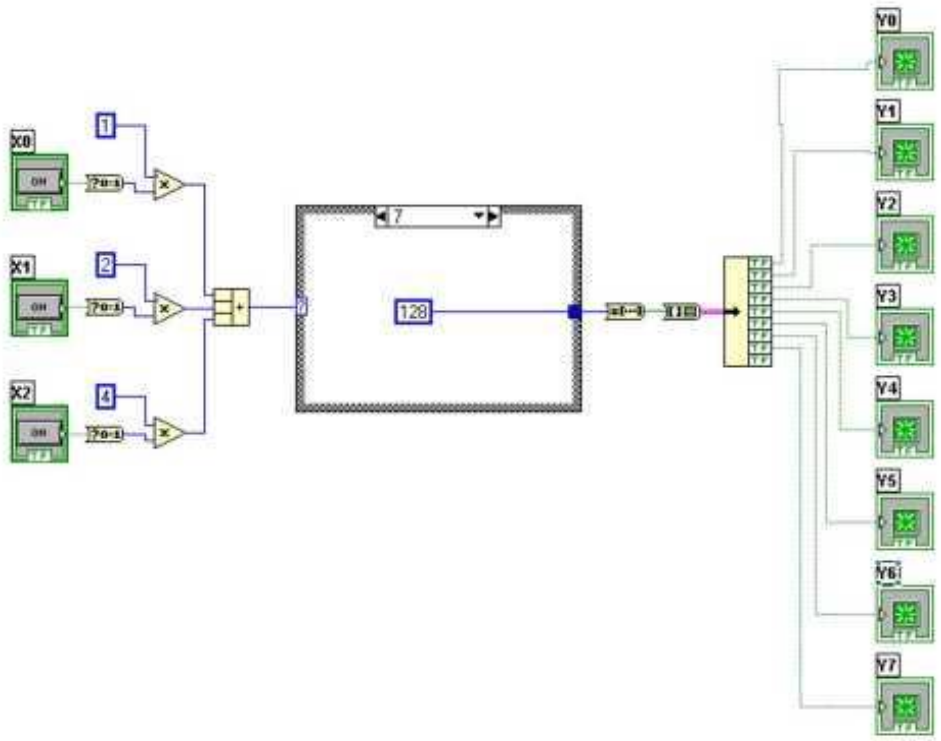


Рисунок 8.15 — Дешифратор, на керуючих структурах *LabVIEW* (*Case Structure*), блок-діаграма

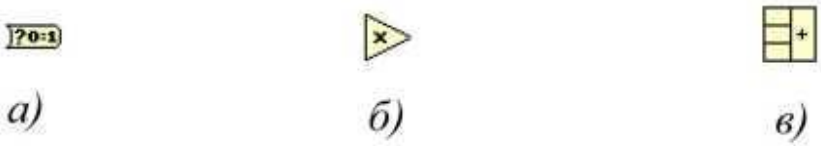


Рисунок 8.16 — Функціональний вузол Boolea: а) to *0,1; б)* Multiply*; в)* Compound Arithmetics

Далі сигнал поступає в структуру Варіант, в якій задано вісім умов, для кожного з восьми можливих чисел на вході. На вихід структури поступає двійкове число, відповідне відображенню у восьми вихідних бітах для одного з випадків вхідної комбінації біт.

Для того, щоб відобразити це число на світлодіодних індикаторах, воно з двійкового представлення числа повинно перетворитися в одновимірний масив біт (рис. 8.17, а), який перетвориться в кластер, що складається з 8 біт

(рис. 8.17, б), далі в блоці Unbundle (рис. 8.17, в) кластер розбивається на складові його потоки біт, які поступають на вихідні світлодіодні індикатори.

Рисунок 8.17 — Процес перетворення числа для відображення його на світлодіодних індикаторах

Хоча в цьому випадку другий спосіб створення дешифратора виявляється складнішим, він дає можливість зрозуміти основні прийоми роботи з логічними і цілими типами даних, способи їх конвертації з одного типа в інший. Аналогічним чином можна створити інші віртуальні прилади, такі як шифратор, мультиплексор і демультиплексор.

Хід роботи

8.1) Виконати приклади для створення віртуальних приладів.

8.2) Отримати і ознайомитися з індивідуальним завданням.

8.3) Сформувати необхідні інструменти.

8.4) Налагодити програму і представити викладачеві для перевірки.

8.5) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

• звітність про виконання прикладів для створення віртуальних приладів;

• варіант і зміст індивідуального завдання;

• скріншоти складових програми і результати її виконання;

• відповіді на запитання до лабораторно роботи.

8.6) Представити звіт на захист.

Варіанти завдань

Завдання №8.1. Створіть віртуальний прилад, що демонструє роботу основних логічних елементів, лицьова панель приладу має бути виконана приблизно так, як показано на рис. 8.18.

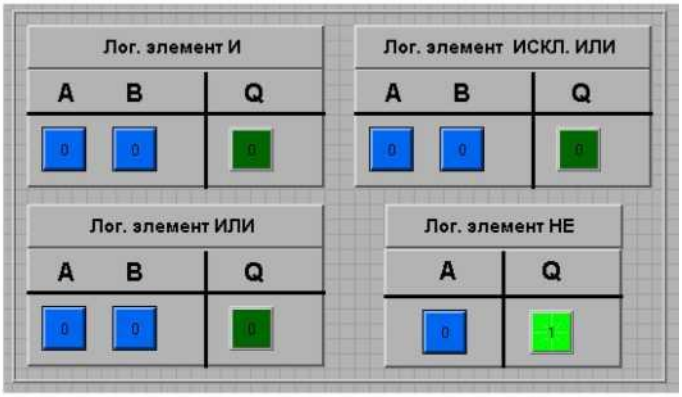


Рисунок 8.18 — До завдання 8.1

Завдання №8.2. Оформіть віртуальні прилади АБО та ЗАПЕРЕЧЕННЯ у вигляді підпрограм і збережіть їх.

Завдання №8.3. Реалізуйте логічну функцію: $Y = \left(\overline{A \bullet \overline{B}}\right)$.

Завдання №8.4. Створіть віртуальний прилад − дешифратор.

Завдання №8.5. Створіть віртуальні прилад - шифратор.

Завдання №8.6. Створіть віртуальні прилад – мультиплексор.

Завдання №8.7. Створіть віртуальні прилад – демультиплексор.

Контрольні запитання

8.1) У цій роботі всі віртуальні прилади оперують з типом даних Boolean, що це за тип даних?

8.2) Навіщо потрібно створювати підпрограми (SUBVI)? Які переваги вони дають?

8.3) Які з базових логічних функцій вже реалізовані в LabVIEW?

8.4) Навіщо потрібні бібліотеки підпрограм? Чи можна без них обійтися?

8.5) Скільки разів основна програма може мати в своєму тілі викликів підпрограм?

8.6) Чи може основна програма при виклику підпрограми передавати туди які-небудь дані і отримувати їх назад?

8.7) На структурних схемах приладів в цій лабораторній роботі ви бачили сполучні дроти зеленого кольору, що він позначає в LabVIEW?

8.8) З якими типами даних ви оперували в цій роботі?

8.9) Який тип даних позначається на рис. 15 синіми провідниками?

8.10) Який тип даних позначається жирним рожевим провідником? Який тип даних позначається жирним зеленим провідником?

8.11) Яке призначення функціонального вузла Multiply? Яке призначення функціонального вузла Compound Arithmetic?

8.12) Яке призначення функціонального вузла Unbundle? Яке призначення функціонального вузла Array to cluster?

8.13) Яке призначення функціональних вузлів Boolean, Number to Boolean array?

8.14) Що ви знаєте про структуру Варіант (Case structure)?

8.15) Для якого типу даних виконувала вибір умов для дешифратора в цій роботі структура «Варіант»?

8.16) Скільки умов було реалізовано в структурі Варіант?

ЛАБОРАТОРНА РОБОТА № 9
ДИСТАНЦІЙНИЙ ДОСТУП ТА КЕРУВАННЯ ВІРТУАЛЬНИМИ ПРИЛАДАМИ LABVIEW

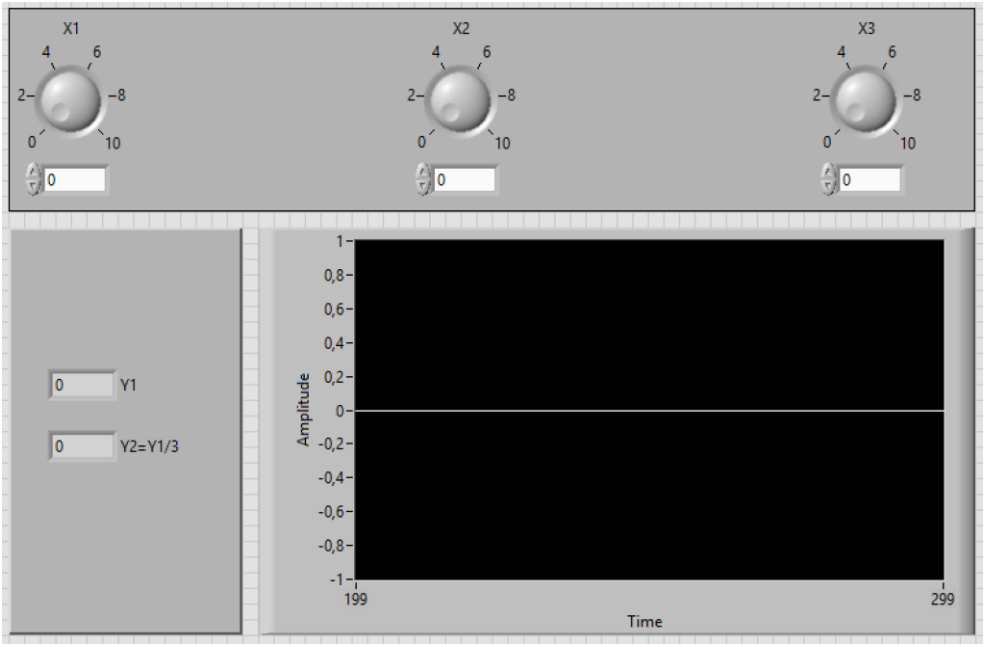
Мета роботи: Вивчення засобів дистанційного доступу до віртуальних приладів *LabVIEV*.

Стислі теоретичні відомості

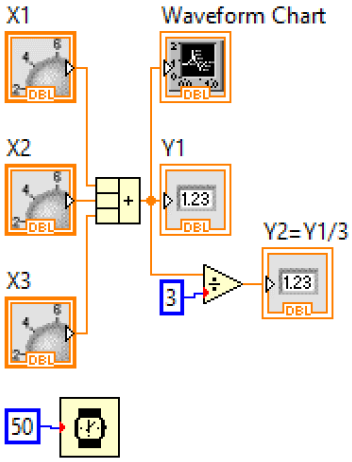
LabVIEW дозволяє створювати локальний веб-сервер для того, щоб надати іншим людям можливість подивитися на роботу створеного віртуального приладу, або навить надати можливість керувати приладом дистанційно. В рамках цієї лабораторної роботи буде описано як створити такий сервер та передавати керування іншій людині.

Приклад 9.1

Створимо віртуальний прилад, який визначає суму трьох чисел та їх середньоквадратичне значення, будує графік залежності суми трьох доданків від часу. Доданки задаються на лицьовій панелі, результати обрахунку також виводяться на лицьову панель приладу. Для на блок-діаграму потрібно добавити арифметичні функції складання та ділення (*Numeric -> Add* та *Numeric -> Divide* відповідно або *Numeric -> Compound Arithmetic* яка дозволяє виконувати однакову дію з великою кількістю членів), функцію затримки, функцію виведення графіків *Wavaform Chart* та індикаційні табло окремо для суми, окремо для середньоквадратичного значення.. Керування значеннями доданків виконується за допомогою регулятора *Knob* в межах від 0 до 10 з кроком 0,01. Кожен з регуляторів має індикаційне табло, яке показує встановленне значення (визивається за допомогою контекстного меню, *Visible Items -> Digital Display*). Обрахунок виконується кожні 50 мс, тому слід додати функцію *Timing -> Wait* (*ms*) та встановити затримку саме на цей час (рис 9.1).



а)



б)

Рисунок 9.1 — ВП до прикладу 9.1: а) передня панель; б) блок-діаграма

Приклад 9.2

Додамо можливість дистанційного доступу до приладу та дистанційного керування. Для цього відкриємо додаток *Web Publishing Tool* (*Tools -> Web Publishing Tool*). Зовнішній вигляд цього додатку наведено на рисунку 9.2.

Розділ *VI name* дозволяє вибрати віртуальний прилад, який буде транслюватися у мережу. Можливо обрати вже відкритий файл чи обрати сторонній.

Розділ *Viewing Mode* дозволяє вибрати в якому режимі буде відбуватися трансляція, а саме:

• *Embedded* – вбудовує лицьову панель приладу, тому клієнти зможуть віддалено контролювати дані приладу та навіть керувати ним. Галка напроти тексту *Request control when connection is established* вказує на те, що при під'єднанні користувач відразу отримає доступ до керування віртуальним приладом;

• *Snapshot* – відображає статичну картинку лицьової панелі віртуального приладу. Оновлення виконується власноруч користувачем;

• *Monitor* – виводить статичну картинку, яка оновлюється через вказаний проміжок часу.

Слід зазначити що лише режим *Embedded* дозволяє керувати віртуальним приладом, інші дозволяють тільки спостерігати за його роботою.

Пункт *Show border* дозволяє вмикати чи вимикати відображення границь приладу.

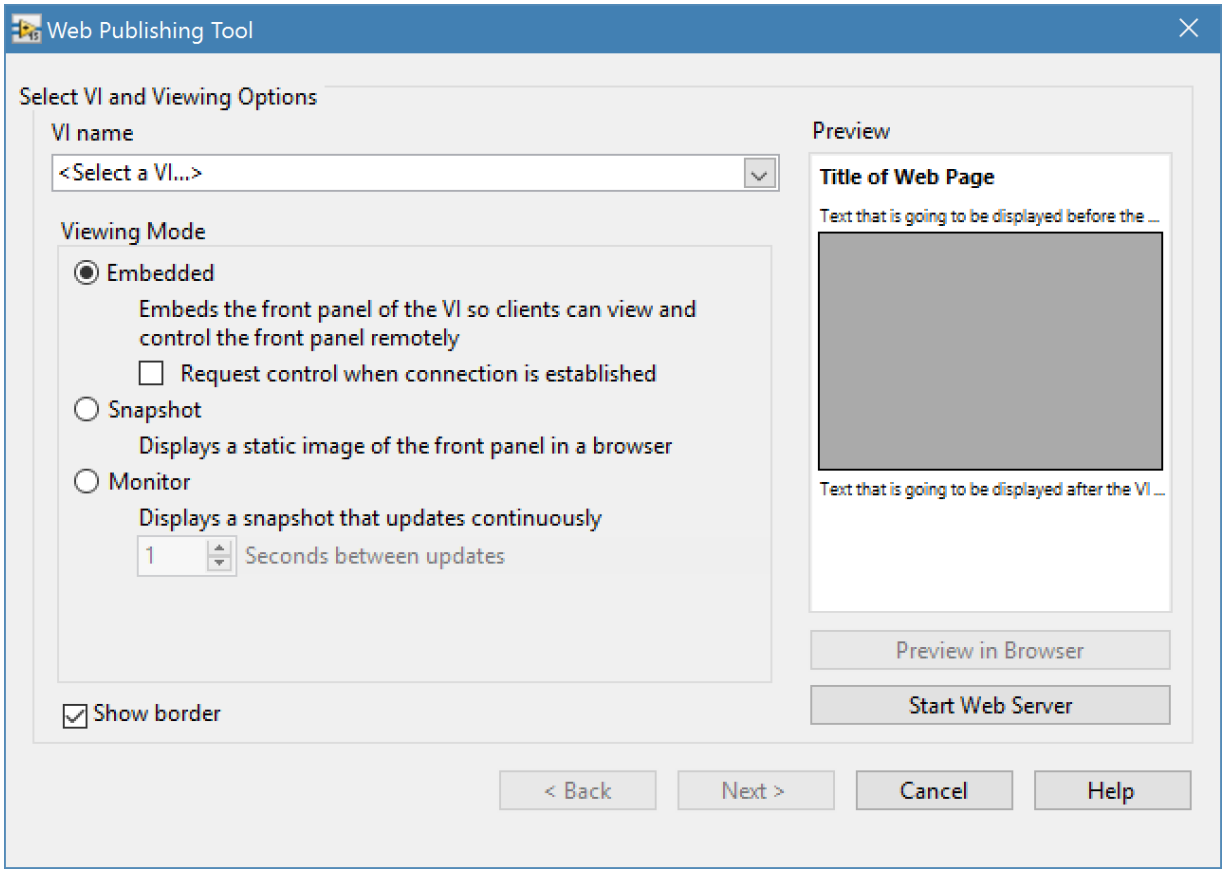


Рисунок 9.2 — ВП до прикладу 9.2

Налаштуємо дистанційний доступ та керування для приладу з прикладу 9.1. Для цього виконаємо наступні дії:

1) У додатку *Web Publishing Tool* виберемо цей пристрій, виберемо режим *Embedded* та нажмемо на кнопку *Start Web Server*. Після вибору приладу справа можливо отримати попередній вигляд веб-сторінки. Нажимаємо *Next*;

2) На наступному вікні (рис. 9.3) є можливість змінити назву сторінки, текст до та після лицьової панелі приладу. Заповнивши як бажаємо нажимаємо *Next*;

3) Останнє вікно (рис. 9.4) дозволяє вибрати де буде зберігатися на локальному комп'ютері веб-сторінка, яке буд мати ім'я. Крім цього виводиться адреса, за якої можна отримати доступ до приладу у веб-браузері. Натискання на кнопку *Save to Disk* завершує роботу додатка, зберігає файли на диск за запускає трансляцію, про що говорить нове спливаюче вікно (рис. 9.5). Запам'ятовуємо адресу та нажимаємо *Ok* чи *Connect*.

Щоб побачити створену веб-сторінку необхідно у веб-браузері *Internet Explorer* перейти по запропонованій адресі. Інші браузери не підходять оскільки трансляція виконується з використанням технології *Microsoft Silverlight*, підтримка якої є тільки у цьому браузері.

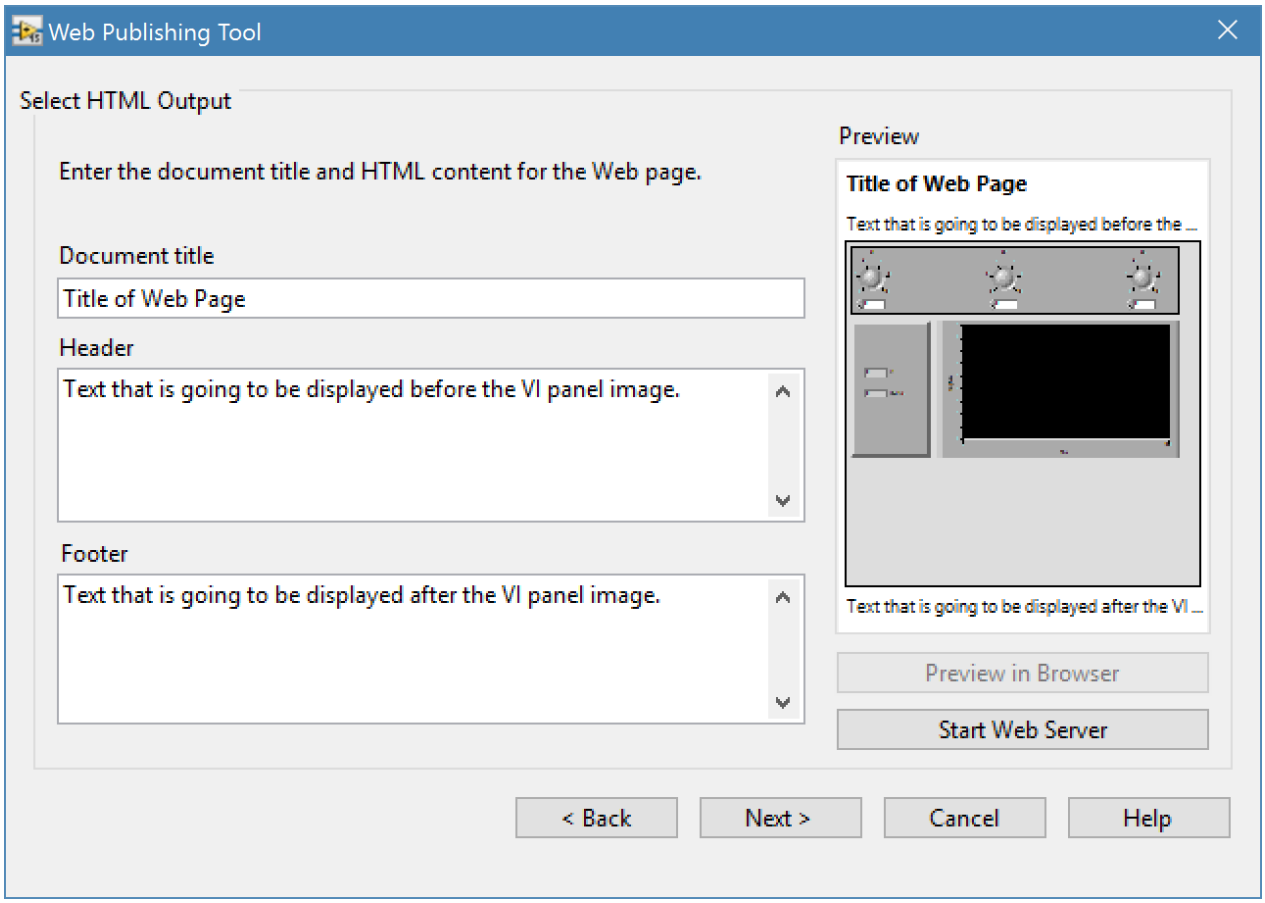


Рисунок 9.3 — Налаштування браузера, проміжне

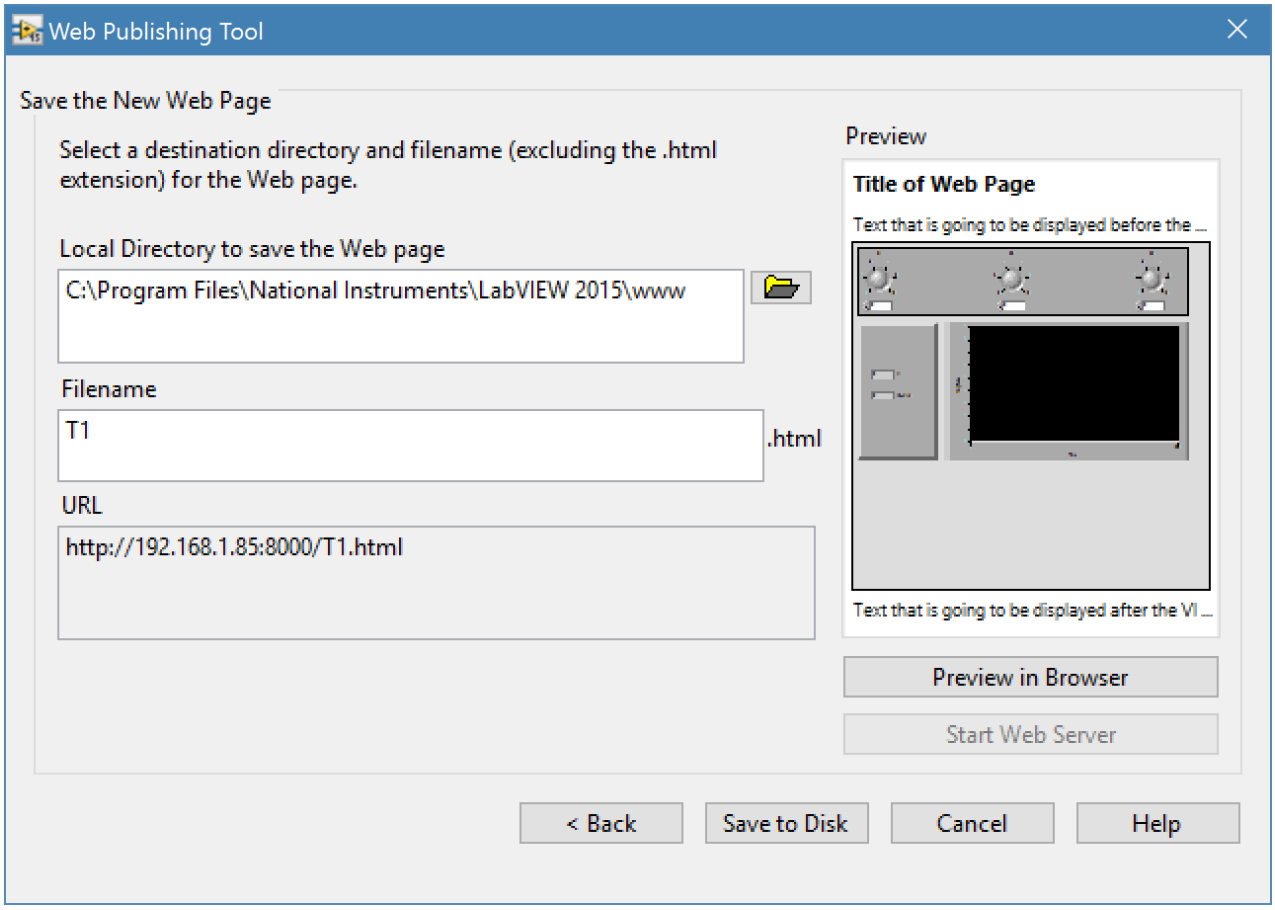


Рисунок 9.4 — Налаштування браузера, останнє вікно

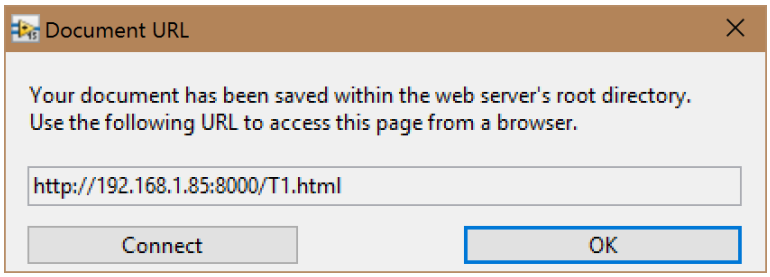
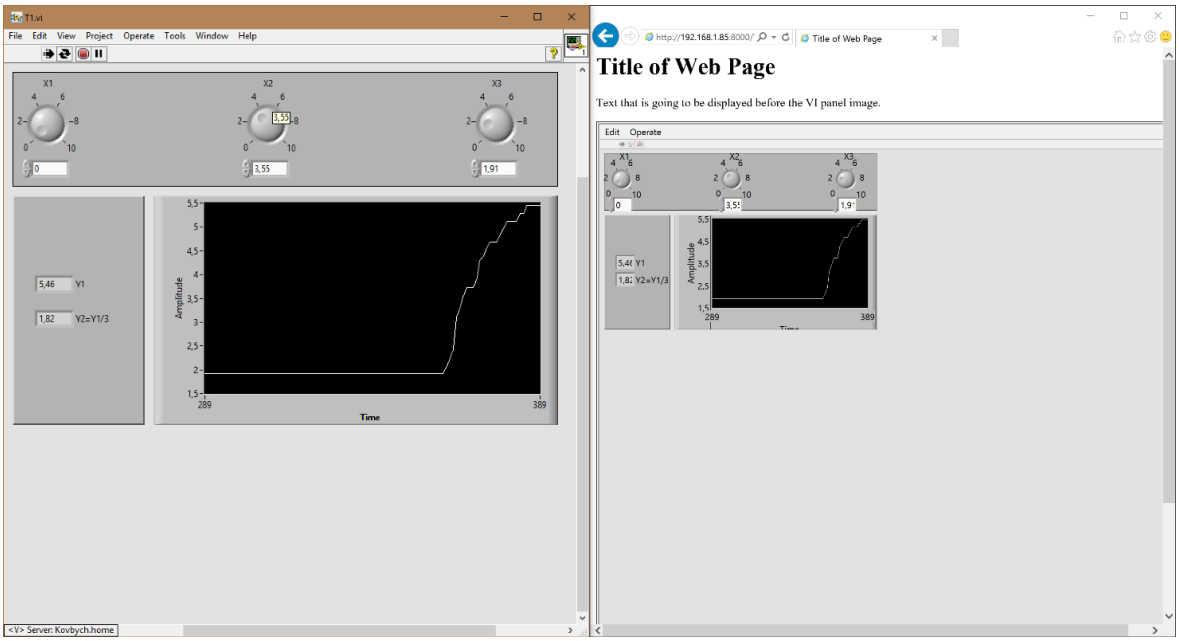


Рисунок 9.5 — Вікно запуску трансляції

Після цього все, що виконується в *LabVIEW* буде відображатися у веб-браузері. Будь-який ПК, що находиться всередині мережі може під'єднатися до створеної сторінки та спостерігати за роботою приладу (див. рис. 9.6). За бажанням статистику трансляції та кількість під'єднаних клієнтів, їх статус можливо побачити у додатку *Remote Panel Connection Manager*.



а) б)

Рисунок 9.6 — Робота ВП через браузер: а) передня панель ВП;
б) відображення роботи ВП в браузері

Для того, щоб отримати контроль над віртуальним приладом у веб-браузері необхідно клацнути правою клавішею миші по лицьовій панелі приладу у веб-браузері та обрати пункт *Request Control of VI*. Після цього керування приладом буде можливе тільки з веб-браузера. Для повернення контролю програмі *LabVIEW* потрібно у контекстному меню обрати *Release Control of VI* або у *LabVIEW* у контекстному меню обрати *Regain Control*.

Керувати пристроєм може лише один клієнт. Також *LabVIEW* дозволяє транслювати декілька віртуальних приладів з одного ПК, але тоді кожен прилад повинен мати своє унікальне ім’я щоб уникнути повторів та мати унікальне посилання на веб-сторінку для кожного приладу.

Хід роботи

9.1) Виконати приклади для створення віртуальних приладів.

9.2) Продемонструвати викладачу працездатність трансляції на одному чи декількох ПК.

9.3) Скласти індивідуальний звіт про виконану роботу, який повинен включати:

• звітність про виконання прикладів для створення віртуальних приладів;

• скріншоти складових програми і результати її виконання, відповіді на запитання до лабораторно роботи.

9.4) Представити звіт на захист.

Контрольні запитання:

9.1) Яку користь дає дистанційний доступ до віртуального приладу?

9.2) За допомогою якого додатку робиться веб-сервер у *LabVIEW*?

9.3) Чи можливо дистанційно керувати віртуальним приладом?

9.4) Що потрібно зробити щоб почати керувати віртуальним приладом?

9.5) Чи можливо транслювати одночасно декілька віртуальних приладів у мережу?

9.6) Які є режими трансляції?

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт компанії National Instruments: [Електронний ресурс]. — Режим доступу : http://www.ni.com

2. Ronald W. Larsen LabVIEW for Engineers. / Larsen Ronald. — Pentice Hall. — 2011. — 396 p

3. Visual Programming Languages. Огляд візуальних мов програмування (in English) URL : http://blog.interfacevision.com/design/design-visualprogarmming-languages-snapshots/ .

4. ДСТУ 3008:2015 Звіти у сфері науки і техніки.

5. Кащеєв Л. Б., Коваленко С. В., Коваленко С. М. Інформатика. Основи візуального програмування : навчальний посібник. Харків : Веста, 2011. 192 с.

6. NI-DAQmx Express VI Tutorial [Електронний ресурс]. — Режим доступу : http://www.ni.com/whitepaper/2744/en. — Назва з екрану.

7. Дорожковець М. Опрацювання результатів вимірювань / Дорожковець М. — Львів : Видавництво Національного університету «Львівська політехніка», 2007. — 624 с.

8. Мірошніченко І.Г. Вивчення віртуальних вимірювальних приладів [Електронний ресурс] / І.Г. Мірошніченко_Волинський державний університет ім. Лесі Українки, м. Луцьк 2006. – http://journals.uran.ua/index.php/2307-4507/article/download/35332/31428.

9. Луценко Г.В. Використання засобів LabVIEW у процесі обробки експериментальних даних статистичними методами [Електронний ресурс] / Г.В. Луценко Черкаський національний університет імені Богдана Хмельницького, м. Черкаси 2013. – https://journal.iitta.gov.ua/index.php/itlt/article/view/816/633.