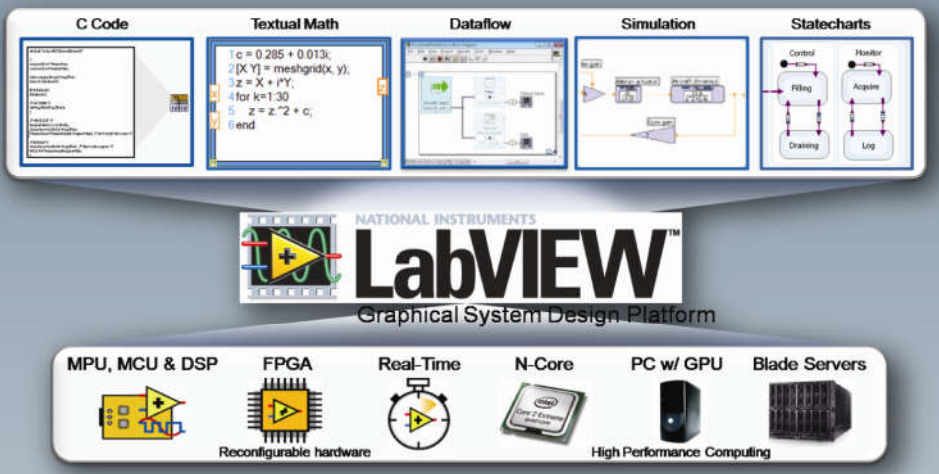
ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС
LabVIEV

Лабораторний практикум
для студентів спеціальностей «Галузеве машинобудування»
і «Електроенергетика, електротехніка та електромеханіка»



Хмельницький національний університет

ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС LabVIEW

Лабораторний практикум для студентів спеціальностей «Галузеве машинобудування» і «Електроенергетика, електротехніка та електромеханіка»

Затверджено на засіданні кафедри машин і апаратів, електромеханічних та енергетичних систем. Протокол № 10 від 22.05.2018

Хмельницький 2018

Програмно-апаратний комплекс LabVIEW : лабораторний практикум для студентів спеціальностей «Галузеве машинобудування» і «Електроенергетика, електротехніка та електромеханіка» / С. В. Смутко, П. С. Майдан, С. П. Лісевич. – Хмельницький : ХНУ, 2018. – 100 с.

Укладачі: Смутко С. В., канд. техн. наук, доц.;
Майдан П. С., канд. техн. наук, доц.;
Лісевич С. П., ст. викл.

Відповідальний за випуск: Поліщук О. С., канд. техн. наук, доц.

Редактор-коректор: Яремчук В. С.

Технічне редагування, коректування і верстка: Чопенко О. В.

Макетування та друк здійснено редакційно-видавничим центром Хмельницького національного університету (м. Хмельницький, вул. Інститутська, 7/1). Підп. до друку 4.09.2018. Зам. № 60е/18, електронне видання, 2018.

ХНУ, 2018

ВСТУП

Основна вимога сучасної освіти – підготовка фахівців, максимально адаптованих до самостійної роботи і подальшого самонавчання в обраній сфері діяльності. Випускники, крім якісної теоретичної підготовки, повинні мати практичні навички до роботи в реальних виробничих умовах. Тому на сьогодні використання технології віртуальних приладів є вигідним та виправданим кроком.

Комп'ютерне моделювання фізичних явищ і засобів вимірювань фізичних величин стало невід'ємною частиною сучасної технічної освіти. Створювані комп'ютерні моделі можуть мати широке призначення, в т. ч. можуть використовуватися для створення комп'ютерних лабораторних практикумів. Для їх реалізації необхідне належне розуміння суті процесу, уміння визначити початкові та граничні умови, обирати відповідний метод розрахунків і спосіб представлення результатів.

Професійна версія **LabVIEW** непогано зарекомендувала себе в багатьох науково-технічних проектах і є міжнародним стандартом систем збору даних та керування вимірюваннями. Вона досить зручно інтегрується в складні технічні апаратно-програмні комплекси. Разом з тим, **LabVIEW** поряд з професійною спрямованістю має широкі можливості для використання у навчальному процесі. Лабораторний практикум, що базується на технологіях National Instruments, дозволяє підвищити якість вивчення технічних дисциплін та сформувати початкові навички і вміння, необхідні для успішного оволодіння обраним фахом.

Використання віртуальних вимірювальних приладів дозволяє ввести студента у світ сучасних лабораторних та інформаційних технологій і змінити суть методології лабораторної роботи.

1 ЗАГАЛЬНІ ВИМОГИ ТА РЕКОМЕНДАЦІЇ

1.1. Середовище розробки віртуальних приладів **LabVIEW.** У зв'язку з бурхливим розвитком технологій, включаючи різке збільшення за останні роки продуктивності напівпровідникових приладів і зменшення їхніх розмірів, широким впровадженням комп'ютерів і мікропроцесорів, розвитком стандартів зв'язку й мережевих технологій, інженери змушені рівною мірою збільшувати складність процесів розробки, виробництва й тестування нових продуктів. Важливим компонентом таких процесів стало їх моделювання. При цьому вже близько 30 років інженери та учені використовують середовище графічного програмування **National Instruments LabVIEW** для моделювання й створення автоматизованих систем збору даних і керування приладами.

LabVIEW (**Laboratory Virtual Instrument Engineering Workbench**) – середовище розробки прикладних програм, створене фірмою **National Instruments** (США). У ній використовується інтуїтивно зрозуміла мова графічного програмування G. Її освоєння не вимагає знання традиційних текстових мов програмування. **LabVIEW** надає широкі можливості для проведення обчислень і математичного моделювання. Щодо цього середовище **LabVIEW** конкурентоспроможне в порівнянні з такими відомими системами комп'ютерної математики, як **MATLAB**, **MathCAD**, **Mathematica**, **MAPLE**. Однак найбільш повно можливості **LabVIEW** розкриваються при створенні приладів і систем для вимірювань фізичних величин у наукових експериментах, лабораторних і промислових установках.

Компанія **National Instruments** була створена в 1976 р. трьома засновниками – Джеффом Кодоскі, Джеймсом Тручардом та Біллом Новліним в американському місті Остін штату Техас. Основною спеціалізацією компанії були інструментальні засоби для вимірювань і автоматизація виробництва.

Перша версія **LabVIEW** побачила світ через десять років після створення компанії – в 1986 р. (це версія для **Apple Mac**). Інженери фірми вирішили кинути виклик «традиційним» мовам програмування і створили повністю графічне середовище розробки. Основним ідеологом графічного підходу став Джефф Кодоскі. Рік за роком випускалися нові версії. Першою крос-платформенною версією (включаючи **Windows**) була третя версія, випущена в 1993 році.

Весь час середовище **LabVIEW** постійно вдосконалюється завдяки регулярному виходу нових версій, а також випуску спеціалізованих модулів, бібліотек і доповнень. Фактично, воно стало стандартом у ряді галузей науки й техніки. Завдяки своїм ідеям використання розподіленого інтелекту, **LabVIEW** дозволила інженерам, що не мають

досвіду в традиційному програмуванні, швидко створювати складні моделі систем вимірювання й керування та переносити їх у практику.

У своєму розвитку середовище надало користувачам широку гаму інструментів, які утворюють графічну платформу програмного забезпечення для моделювання, керування і тестування, що надає їм конкурентні переваги в трьох базових галузях застосування:

– автоматизованих системах вимірювання й тестування;

– промислових системах контролю й керування;

– проектуванні й налагодженні систем, що монтуються.

Платформа графічної розробки **LabVIEW** збільшує продуктивність праці інженерів і вчених. Сполучення інтуїтивно зрозумілої графічної мови програмування, підтримки широкого набору пристроїв вводу/виводу й зростаючого співтовариства користувачів, що беруть участь у розвитку платформи **LabVIEW**, робить успішним створення принципово нових додатків. Використовуючи відкрите середовище програмування **LabVIEW** для втілення розроблених алгоритмів і обміну даними із засобами моделювання, можна модернізувати засоби розробки й скоротити тимчасові витрати на всіх етапах життєвого циклу виробів.

Сьогодні розрізнені контрольно-вимірювальні системи підприємств поєднуються в розподілені системи більш високого рівня з повною інтеграцією обчислювальних і керуючих ресурсів. У цьому плані **LabVIEW** є високоефективним та простим у використанні середовищем для проектування, керування, запуску й синхронізації розподілених систем.

Важливою перевагою **LabVIEW** є можливість керування процесом вимірювання в автоматичному або інтерактивному режимі.

Для обробки й аналізу даних використовується великий набір функціональних бібліотек (загального призначення й спеціалізованих). Взаємодія з дослідником або оператором здійснюється за допомогою продуманого й простого в програмуванні графічного інтерфейсу. За допомогою програм-драйверів **LabVIEW** ефективно взаємодіє з різноманітними платами вводу/виводу аналогових і цифрових сигналів, модулями введення відеосигналів, а також зі спеціалізованими модульними приладами (осцилографи, аналізатори спектра, генератори сигналів і т.д.).

Останні версії **LabVIEW** орієнтовані на створення розподілених і дистанційних систем вимірювань. Це дозволяє забезпечити доступ на відстані до унікальних експериментальних стендів і організувати дистанційне навчання. Можливості базового пакета можуть бути розширені за допомогою спеціалізованих модулів і функціональних бібліотек.

Для задоволення поточних і перспективних потреб користувачів **LabVIEW** забезпечує:

– підтримку різної архітектури і платформ виконання, таких, як персональні, промислові, портативні та вбудовані комп'ютери;

– моніторинг і керування розподіленими вузлами системи з єдиної інтерактивної оболонки (**LabVIEW Project**);

– спрощення передачі даних між різними обчислювальними вузлами за допомогою нової змінної загального доступу (**LabVIEW Shared Variable**);

– підтримку багатьох варіантів синхронізації та тактування вузлів розподілених систем через нову технологію детермінованого Ethernet.

LabVIEW підтримує величезний спектр обладнання різних виробників і має у своєму складі (або дозволяє додавати до базового пакету) численні бібліотеки компонентів.

1.2. Організація та порядок виконання лабораторних робіт. Роботи виконуються кожним студентом індивідуально, на своєму робочому місці, за своїм варіантом завдання. Лабораторна робота завершується оформленням звіту та її захистом і вважається зарахованою, якщо звіт містить необхідні блок-діаграми, схеми, графіки, виконані правильно та акуратно, а також, якщо студент відповів на питання викладача, показавши знання з будови та принципу роботи об'єкта досліджень і розуміння процесів, що пояснюють отримані результати. Крім того, студент повинен знати призначення всіх елементів блок-діаграм і вміти пояснити порядок дій при їх складанні.

1.3. Обробка результатів експерименту та оформлення звіту. Кожен студент повинен самостійно обробити результати виконаних ним дослідів і скласти звіт з лабораторної роботи. Звіт, крім номера та назви роботи, індексу навчальної групи, повинен містити наступні відомості:

– вигляд лицьової панелі віртуального приладу і його блок-діаграма;

– опис побудови віртуального приладу;

– висновки щодо роботи.

Усі блок-діаграми, таблиці, графіки, схеми, що приводяться в звіті, повинні мати найменування. При виконанні розрахунків рекомендується користуватися калькуляторами.

Особливу увагу необхідно надати виконанню графіків залежностей. На кожній координатній осі має бути зазначене умовне літерне позначення величини, що відкладається та одиниця її вимірювання. В останньому розділі звіту – у висновку про виконану роботу – студент повинен дати оцінку побудованому віртуальному пристрою, його властивостям. Звіт повинен бути лаконічним, але таким, щоб його зміст був зрозумілим без додаткових усних пояснень.

Обсяг звіту не повинен перевищувати чотирьох сторінок ф. А4.

1.4. Критерії оцінювання. При виконанні лабораторної роботи та захисту звіту до неї, використовують наступну систему критеріїв оцінювання.

Оцінку «***відмінно***» (шкала ECTS – A) студент отримує за глибоке і повне опанування теоретичного матеріалу; легко в ньому орієнтується і вміло використовує понятійний апарат; уміння пов'язувати теорію з практикою, вирішувати практичні завдання, впевнено висловлювати і обґрунтовувати свої судження. Відмінна оцінка передбачає грамотний, логічний виклад відповіді (як в усній, так і у письмовій формі), якісне зовнішнє оформлення роботи. Студент не вагається при видозміні запитання, вміє робити детальні та узагальнюючі висновки. При відповіді допустив дві-три несуттєві ***похибки***.

Оцінку «***добре***» (шкала ECTS – B) студент отримує за повне засвоєння теоретичного матеріалу, володіння понятійним апаратом, орієнтування у вивченому матеріалі; свідомо використовує теоретичні знання для вирішення практичних задач; виклад відповіді грамотний, але у змісті і формі відповіді можуть мати місце окремі неточності, нечіткі формулювання закономірностей тощо. Відповідь студента має будуватися на основі самостійного мислення, але з наявністю нечітких формулювань.

Оцінку «***добре***» (шкала ECTS – C) студент отримує за повне засвоєння теоретичного матеріалу, володіння понятійним апаратом, орієнтування у вивченому матеріалі; свідомо використовує теоретичні знання для вирішення практичних задач; виклад відповіді грамотний, але у змісті і формі відповіді можуть мати місце окремі неточності, нечіткі формулювання закономірностей тощо. Відповідь студента має будуватися на основі самостійного мислення. Студент у відповіді допустив дві–три ***суттєві помилки***.

Оцінку «***задовільно***» (шкала ECTS – D) студент отримує за достатні знання основного програмного матеріалу в обсязі, необхідному для подальшого навчання та практичної діяльності за професією, справляється з виконанням практичних завдань, передбачених програмою. Як правило, відповідь студента будується на рівні репродуктивного мислення, студент має слабкі знання структури курсу, допускає неточності і ***суттєві помилки*** у відповіді, вагається при відповіді на видозмінене запитання. Разом з тим набув навичок, необхідних для виконання нескладних практичних завдань, які відповідають мінімальним критеріям оцінювання і володіє знаннями, що дозволяють йому під керівництвом викладача усунути неточності у відповіді.

Оцінку «***задовільно***» (шкала ECTS – E) студент отримує за неповне опанування теоретичного матеріалу, однак отримані знання відповідають мінімальним критеріям оцінювання; розрахунки, гра-

фіки, блок-схеми та висновки виконані з певними неточностями; звіт захищений після закінчення встановленого терміну.

Оцінку «***незадовільно***» (шкала ECTS – FX) студент отримує за розрізнені, безсистемні знання, не вміє виділяти головне і другорядне, допускається помилок у визначенні понять, перекручує їх зміст, хаотично і невпевнено викладає матеріал, не може використовувати знання при вирішенні практичних завдань.

Оцінку «***незадовільно***» (шкала ECTS – F) студент отримує за повне незнання і нерозуміння теоретичного матеріалу та невиконання роботи.

При оцінюванні використовуються різні засоби контролю, зокрема: засвоєння теоретичного матеріалу перевіряється тестовим контролем; якість виконання, набуття теоретичних знань і практичних навичок перевіряється шляхом захисту кожної лабораторної роботи.

Оцінка, яка виставляється за лабораторне заняття, складається з таких елементів: усне опитування студентів перед допуском до виконання лабораторної роботи; знання теоретичного матеріалу з теми; вільне володіння студентом спеціальною термінологією і уміння професійно обґрунтувати прийняті конструктивні рішення; своєчасний захист лабораторної роботи.

Термін захисту лабораторної роботи вважається своєчасним, якщо студент захистив її на наступному після виконання роботи занятті. Пропущене лабораторне заняття студент зобов'язаний відпрацювати в лабораторіях кафедри у встановлений викладачем термін з реєстрацією у відповідному журналі кафедри, але не пізніше, ніж за два тижні до кінця теоретичних занять у семестрі.

1.5. Результати навчання. Студенти, які завершили вивчення дисциплін, повинні: ***уміти*** розробляти прогресивні технологічні процеси на прості види продукції та забезпечувати їх відповідність технічним завданням і чинним нормативно-технічним документам; здійснювати складні вимірювання електричних параметрів устаткування та обробку їх результатів із застосуванням контрольно-вимірювальної та обчислювальної техніки; ***володіти*** навиками проведення експериментальних досліджень з використанням сучасного вимірювального обладнання та комп'ютерних методів обробки інформації.

Лабораторна робота 1

Основи програмування в середовищі LabVIEW

***Мета роботи*:** ознайомлення з організацією програмного середовища **LabVIEW**: вивчення компонент діалогового вікна **LabVIEW LabVIEW**, лицьовій панелі і блок-діаграми, вивчення палітри інструментів **Tools Palette**, палітр елементів контролю **Controls Palette** і функцій **Function Palette**; придбання практичних навичок створення, редагування та налагодження комп'ютерних приладів.

Порядок виконання

1. Ознайомитись з відомостями щодо основних елементів середовища програмування **LabVIEW**.
2. Створити програми, що вказані у завданнях 1.1–1.5 та запустити їх у роботу.
3. Скласти звіт і зробити висновок про виконану роботу.

Теоретичні відомості

Для створення власних програм у середовищі **LabVIEW** використовуються такі інструменти [1]: лицьова панель, блок-діаграма, палітри елементів управління і відображення даних і палітри функцій.

При запуску **LabVIEW** з меню стартового діалогового вікна командами **New→Blank VI** відкриваються два вікна – лицьова панель та блок-діаграма (рис. 1.1 та 1.2).

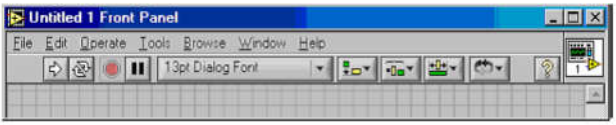


Рис. 1.1 – Лицьова панель LabVIEW

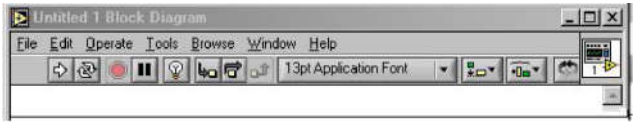


Рис. 1.2 – Панель блок-діаграм LabVIEW

У правому верхньому куті кожного вікна знаходиться піктограма для архівування створеної програми в якості нового комп'ютерного приладу. Тут же розміщена традиційна для додатків **Windows** смуга головного меню з однаковими для обох вікон пунктами: **File**, **Edit**, **Operate**, **Tools**, **Browse**, **Windows**, **Help**. Короткий опис функцій пунктів головного меню наведено в таблиці 1.1.

Таблиця 1.1 – Короткий опис функцій головного меню

Пункти меню	Переклад	Функції
File	Файл	Відкриття, закриття, збереження і друк програм
Edit	Правка	Редагування панелей, пошук об'єктів
Operate	Керування	Запуск та переривання виконання програм
Tools	Інструменти	Керування бібліотеками програм
Browse	Перегляд	Перегляд ієрархій програм
Windows	Вікно	Відображення вікон та палітр **LabVIEW**
Help	Довідка	Додаткова інформація про елементи та функціях **LabVIEW**

Нижче смуг головного меню розташовані лінійки інструментів, які різні для лицьової панелі і блок-діаграми за рахунок додаткових кнопок для налагодження програм (табл. 1.2).

Таблиця 1.2 – Призначення кнопок палітри інструментів

Піктограма	Призначення кнопок	Піктограма	Призначення кнопок
	Кнопка запуск **Run** при правильно складеній програмі		Вид кнопки запуск **Run** при наявності помилок в програмі
	Вид кнопки запуск **Run** в процесі виконання програми		Вид кнопки запуск **Run** в процесі виконання підпрограми
	Кнопка непрервний (циклічний) запуск **RunContinuoslv**		Кнопка зупинка виконання програми **Abort Execution**
	Кнопка тимчасової паузи виконання програми **Pause**		Анімація потоків даних при налагодження програм
	Початок покрокового виконання налагодження програм		Покрокове виконання
	Вихід з покрокового виконання програм		Редагування тексту (шрифт, розмір, стиль и текст)

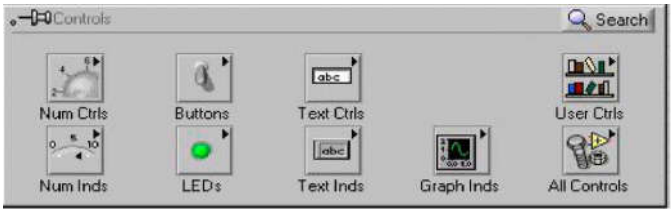
Вільний простір кожної панелі утворює робочу область, забезпечену горизонтальною та вертикальною смугами прокрутки. При розробці програм в робочій області лицьової панелі розміщуються візуальні елементи управління та індикації, що формують інтерфейс користувача, а на панелі блок-діаграми складається графічний код створюваного додатка. Для одночасного відображення на екрані монітора обох панелей доцільно використовувати команду: **Windows Tile Left and Right** або **Tile Up and Down**.

Розробка програм здійснюється за допомогою трьох допоміжних палітр (рис. 1.3):

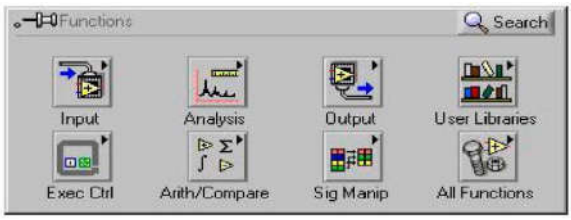
– елементів управління та індикації **Controls Palette** на лицьовій панелі;

– функцій **Functions Palette** на блок-діаграмі;

– інструментів **Tools Palette**, доступної на обох панелях.



а



б *в*

Рис. 1.3 – Допоміжні палітри:
***а*) елементів контролю та індикації; *б*) функцій; *в*) інструментів**

Інструменти мають наступне призначення:

 – керування – для зміни значення елементів керування або введення тексту;

 – переміщення – для активізації, переміщення і зміни розмірів об'єктів;

– введення тексту – для редагування тексту і створення вільних міток;

– з'єднання – створює провідники даних, з'єднуючи об'єкти на блок-діаграмі;

– виклик контекстного меню – викликає контекстне меню відповідного об'єкта по клацанню лівої кнопки миші;

– швидка прокрутка екрану – для перегляду вікна без використання смуги прокрутки;

– введення контрольної точки – дозволяє розставляти контрольні точки у функціях, вузлах, провідниках даних, структурах і припиняти в них виконання програми;

– установка індикаторів налагодження – показує поточне значення змінних в провідниках блок-діаграми, використовується при налагодженні програм для перегляду проміжних значень;

– копіювання кольору – призначений для копіювання і подальшої вставки кольору;

– розфарбування – дозволяє змінити колір об'єкта і відображає поточний фон.

У середовищі **LabVIEW** використовуються різні типи даних (рис. 1.4; табл. 1.3).

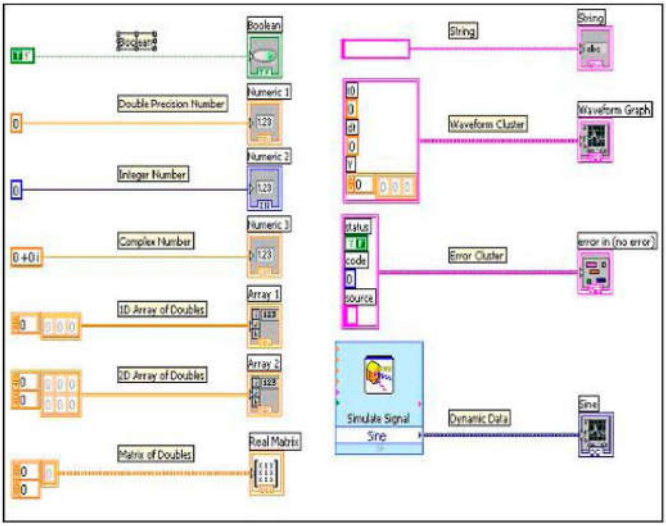


Рис. 1.4 – Типи даних в LabVIEW

Таблиця 1.3 – Типи даних в LabVIEW

Тип даних	Колір
Логічний	Зелений
Число з плаваючою комою	Помаранчевий
Комплексне число	Помаранчевий
Ціле число	Синій
Рядок	Рожевий
Кластер (включає різні типи даних)	Рожевий
Динамічний (інформація про сигналі – ім'я, дата і час отримання даних)	Фіолетовий
Масив (включає тип даних у дужки і приймає колір даних цього типу)	Різний

Завдання до лабораторної роботи

Завдання 1.1. Здійсніть запуск середовища **LabVIEW** з каталогу **C: \ LABV**. У який з'явився головному вікні програми виберіть команди: **New Blank VI** для створення нового файлу. Далі виберіть меню: **Window The Left and Right** для одночасного відображення на екрані двох вікон програми: сірої і білої панелей. Сіра лицьова панель (зазвичай розташовується зліва) – інструмент користувача, який призначений для розміщення елементів введення і виведення даних у вигляді звичних технічних пристроїв, таких як: цифрові покажчики, повзункові реостати, регулятори гучності, осцилографи, самописці, графопобудувачі і т.д.; біла (зазвичай розташовується праворуч) – блок-діаграма, на якій викликаються піктограми різних функцій і структур та складається графічний код програми. Для здійснення різних операцій за допомогою курсору необхідно викликати палітру інструментів за допомогою меню: **Window Show Tools Palette** на лицьовій панелі або на блок-діаграмі.

Завдання 1.2. Створити програму, яка виводить на індикатор значення з елемента управління (рис. 1.5).

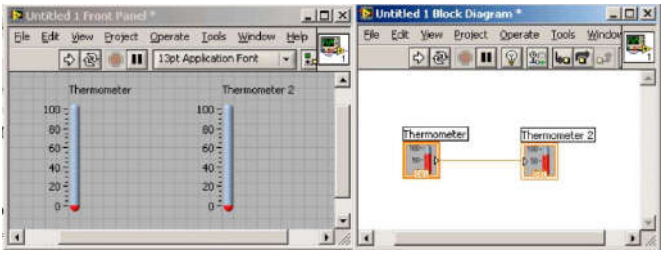


Рис. 1.5 – Лицьова панель та блок-діаграма віртуального приладу (завдання 1.2)

Для цього на лицьовій панелі розмістимо два індикатори **Thermometer** (на палітрі елементів управління і індикаторів виберемо: **Moderns→Numerics Thermometer**). На блок-діаграмі з'явилися дві іконки, звані терміналами даних, які відповідають розміщеним на лицьовій панелі індикаторам. Один з індикаторів необхідно зробити елементом управління. Для цього, наведемо на нього курсор мишки (це можна зробити, як на лицьовій панелі, так і на блок-діаграмі) і, натиснувши праву клавішу у спливаючому меню, виберемо пункт **Change to Control** – змінити на елемент управління. При цьому дії на блок-діаграмі у відповідної іконки трикутник з правого боку перемістився на ліву. У індикаторів трикутник означає вхід, на який необхідно подавати відповідні значення, а у елементів управління – вихід, з якого зчитуються значення. При підведенні курсору мишки трикутнику (виходу) елемента управління, курсор прийме вигляд котушки. Після одноразового натискання лівої клавіші мишки, потягніть мишку до входу індикатора **Thermometer 2**. За курсором потягнеться провідник. Підіб'ємо провідник до входу індикатора і, як тільки курсор перетвориться знову в котушку, зробимо одноразове натиснення мишки. Після цього блок діаграма і лицьова панель будуть виглядати, як це показано на рис. 1.5.

Працездатність програми краще перевіряти натисканням кнопки **Run Continuously** – циклічний запуск. При зміні мишкою значення елементу управління **Thermometer** змінюються значення на індикаторі **Thermometer 2**.

Зупиніть програму натисканням клавіші **Stop**, розірвіть провідник, що з'єднує елемент керування та індикатор (для цього виділіть його мишкою і натисніть клавішу **Delete**). Запустіть програму циклічно ще раз і порівняйте результат роботи.

Завдання 1.3. Створимо програму, що конвертує температуру представлену в градусах Цельсія в температуру за Фаренгейтом, використовуючи формулу:

$$F = 1,8C + 32\,, \tag{1.1}$$

де F – температура за Фаренгейтом, C – температура в градусах Цельсія.

Виконаємо наступні дії:

– створіть новий віртуальний прилад **Blank VI**;

– на лицьовій панелі розмістіть два індикатори **Thermometer** (на палітрі елементів керування та індикаторів виберемо: **Modern→ Numeric→Thermometer**);

– подвійним натисканням лівої клавіші мишки по мітках індикаторів виділіть їх і перейменуйте в «C» і «F»;

– індикатор з міткою «C» зробіть елементом управління;

– перейдіть на блок-діаграму.

Так само створіть другу константу і надайте їй значення 32, як показано на рис. 1.6.

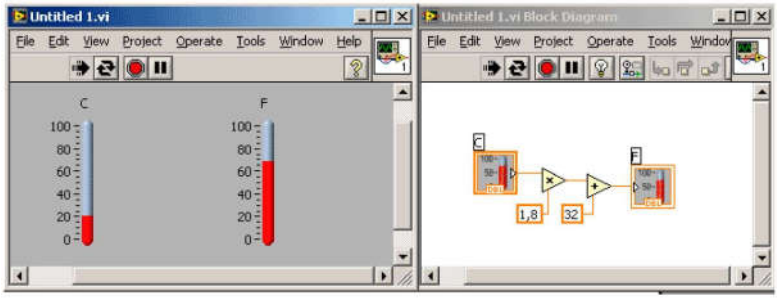


Рис. 1.6 – Програма конвертації температури в градусах Цельсія в температуру за Фаренгейтом

Запустіть програму, натиснувши клавішу **Run Continuously**.

Зупиніть програму натисканням клавіші **Stop**, розірвіть провідник, що з'єднує елемент керування і індикатор (для цього виділіть його мишкою і натисніть клавішу **Delete**). Запустіть програму циклічно ще раз і порівняйте результат роботи.

Завдання 1.4. Створіть нову програму. На лицьовій панелі розмістіть два числові елемента керування, назвіть їх «*X*» і «*Y*» і два числових індикатора (рис. 1.7). На блок-діаграмі реалізуйте алгоритм, такий, щоб на одному індикаторі виводилася сума, а на іншому різниця значень, введених в елементи управління.

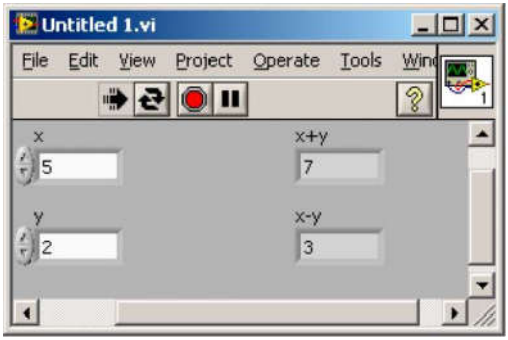


Рис. 1.7 – Лицьова панель віртуального приладу (завдання 1.4)

Завдання 1.5. Клацанням ПКМ на лицьовій панелі викликаємо палітру елементів контролю і управління та закріплюємо її в стаціонарному положенні за допомогою інструменту «кнопка» в лівому верхньому кутку палітри. У ній активізуємо елементи контролю – перша піктограма у першому ряду – для завдання вихідних параметрів. Виділяємо курсором по черзі «цифровий регулятор», «реостат», «ручку регулятора гучності» і переносимо їх на верхню частину лицьовій панелі.

Створимо п'ять елементів індикації роботи приладів: «стрілочний амперметр», «манометр», «термометр», «лінійний індикатор» і «осцилограф». Для цього активізуємо піктограму «елементи індикації», вибираємо в ній відповідні прилади і переносимо їх на вільну частину лицьової панелі. Звернемо увагу, що при появі будь-якого нового елемента на лицьовій панелі одночасно з'являється його модифіковане зображення на блок-діаграмі. Подальше програмування в середовищі **LabVIEW** практично зводиться до з'єднання елементів блок-діаграми провідниками даних. При цьому вид провідника автоматично вибирається відповідним типом даних (див. рис. 1.4).

Для роботи з блок-діаграмою потрібні додаткові інструменти, які викликаються з головного меню як палітра інструментів **Tools Palette**, доступна на обох панелях – **Window→Show Tools Palette**.

Подамо вихідні сигнали керуючих елементів на входи довільних індикаторів, з'єднуючи їх провідниками даних за допомогою інструменту «котушка». Оскільки керуючих елементів менше, ніж індикаторів, розділимо вихід одного з них на два за рахунок приєднання додаткового провідника до будь-якої з ліній передачі даних. Білою стрілкою **Run** включаємо періодичний запуск роботи складених програм. Змінюючи на лицьовій панелі значення вихідних величин, простежимо відображення цих змін на показують приладах. Звернемо увагу на відповідність шкал керуючих елементів і показують приладів. При необхідності скоректуйте їх за допомогою інструменту «редагування тексту».

Клацанням ПКМ на блок-діаграмі викличемо панель всі функції і закріпимо її. У ній знаходимо палітру «арифметичні дії», відкриваємо і переносимо на блок-діаграму два елементи: «підсумовування» – **Add** та «генератор випадкових чисел» – **Random Num**. Для цього вибираємо: **Functions Arith→Compared Numeric**. Виділяємо клацанням ПКМ провідник, що з'єднує обраний регулятор з осцилографом і видаляємо провідник.

Виділити елемент «підсумовування» і викликаємо довідку **Help**, яка показує схему його підключення. Відповідно до цієї схеми, підводимо до одного з входів суматора сигнал з обраного регулятора, а до іншого – генератор випадкових чисел (рис. 1.8).

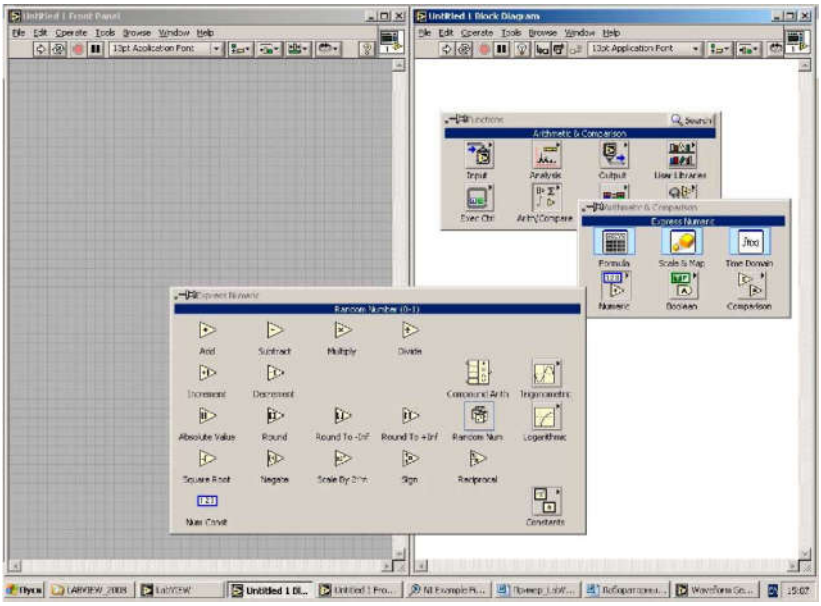


Рис. 1.8 – Палітра арифметичних дій

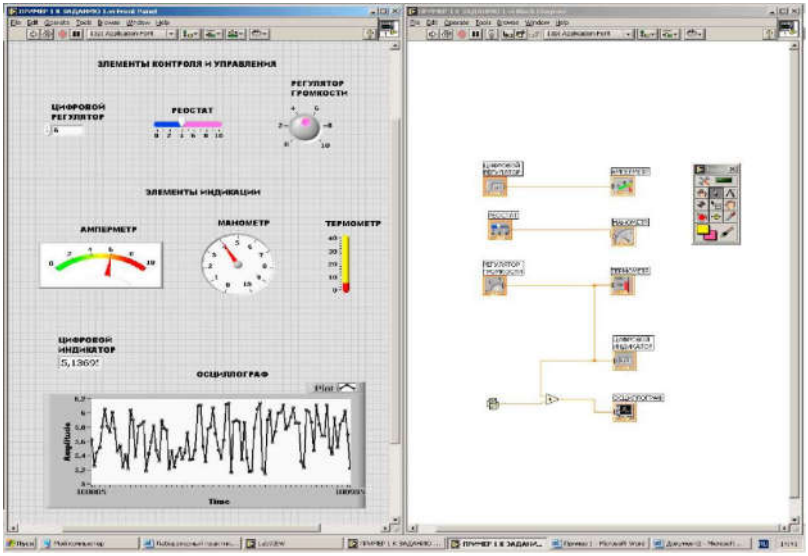


Рис. 1.9 – Лицьова панель та блок-діаграма

Результат підсумовування подаємо на вхід осцилографа і при білій стрілкою **Run** включаємо періодичний запуск. При працюючій програмі розгляньте і перепишіть у звіт різні форми представлення результатів обчислення на графіку – у вигляді суцільної лінії, окремих точок, відрізків прямих, що з'єднують сусідні точки і т.д.

Зупиніть програму кнопкою **Stop**. За допомогою інструменту «лампочка» і кнопки «періодичний запуск» включіть режим анімації потоків даних, використовуваний при налагодженні програм. Прослідкуйте рух даних по провідниках і їх перетворення на елементах блок-діаграми (див. рис. 1.9).

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу.
4. Висновки з роботи.

Контрольні питання

1. Які команди використовуються в середовищі **LabVIEW** для налагодження програм?
2. Чи має якесь значення порядок підключення провідників до елементів підсумовування і віднімання, множення і ділення?
3. Яка форма графічного представлення результатів роботи програми в найбільшій мірі відображає дискретний принцип роботи ПК? У яких випадках доцільніше використовувати інші графіки?

Література: [1, 2, 4]

Лабораторна робота 2

Цикли та умовні оператори в середовищі LabVIEW

***Мета роботи*:** ознайомлення з організацією циклів в програмному середовищі **LabVIEW**; використання термінала вихідних даних циклу **While**; використання регістрів зсуву у циклі **For**.

Порядок виконання

1. Ознайомитись з теоретичними відомостями щодо виконання циклічних операцій в середовищі **LabVIEW**.
2. Створити програми, вказані в завданнях 2.1 та 2.2, і запустити їх у роботу.
3. Скласти звіт і зробити висновок про виконану роботу.

Теоретичні відомості

Для виконання операції в програмі **LabVIEW** або її частині певну кількість разів або поки не виконається якась умова, наприклад, натискання кнопки **Stop**, використовуються цикли.

В **LabVIEW** використовується два варіанти конструкції «цикл»:

– **While Loop** із невідомим числом ітерацій, що виконується доки не буде виконана умова виходу з циклу;

– **For Loop** із відомим числом ітерацій.

Конструкції цих циклів розташовуються в палітрі **Functions** у розділі **Programming Structures**.

Цикл за умовою **While Loop** [1] аналогічний циклу **While**, використовуваному в текстовій мові програмування Сі, виконує багаторазове повторення операції над потоком даних, поки не виконається логічна умова виходу. Цикл **While** розташований на палітрі функцій у розділі структури – **Programming→Structures**.

Після того, як цикл знайдений і обраний на палітрі функцій, треба за допомогою курсору змінити проміжні границі структури для виділення частини блок-діаграми, яку необхідно помістити в цикл. Після відпускання кнопки миші, виділена область блок-діаграми поміщається в тіло циклу. Додавання об'єктів блок-діаграми в тіло циклу здійснюється приміщенням або перетаскуванням об'єкта.

Блок-діаграма циклу за умовою **While Loop** виконується доти, поки не виконається умова виходу. За замовчуванням, термінал умови виходу вказує, що цикл буде виконуватися до надходження на термі-

нал значення «неправда» **FALSE**. У цьому випадку термінал умови виходу називається терміналом «продовжити якщо істина – **Continue If True**».

Термінал лічильника ітерацій, показаний ліворуч, містить значення кількості виконаних ітерацій. Початкове значення термінала «**i**» завжди дорівнює нулю.

Як приклад, створимо програму виводу в масив чисел від 0 до 9 [4]. Для цього організовуємо цикл **While**, термінал **Loop Iteration** порівнюємо з константою (лічильник ітерацій починає свою роботу з 0, то константу беремо рівною 9) і зупиняємо цикл, якщо рівність справджується. Результат виводимо на числовий індикатор (рис. 2.1).

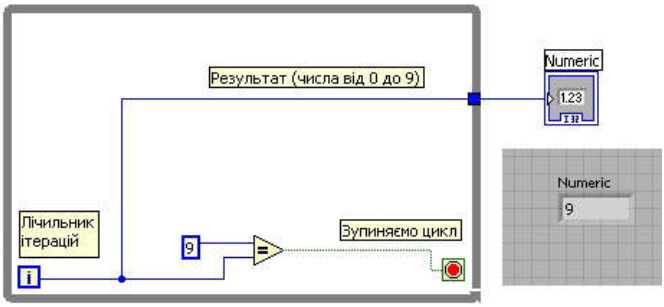


Рис. 2.1 – Приклад циклу While Loop

Програма видала не всі числа, а тільки останнє з них (та й індикатор ніяк не схожий на масив). Це тому, що для циклів типу **While Loop** за замовчуванням вимкнено автостворення масивів – **Indexing**. Спробуємо це виправити. Натискаємо праву кнопку миші на тунель (синій квадратик на межі циклу) і в контекстному меню обираємо **Enable Indexing** (рис. 2.2).

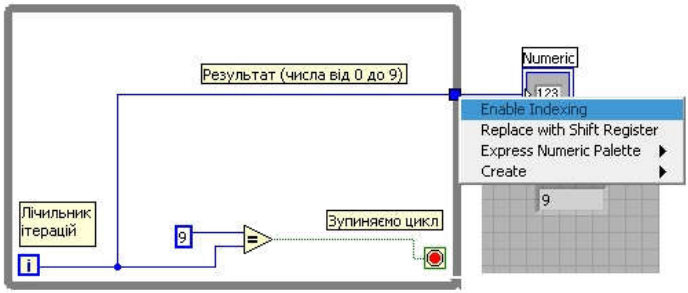


Рис. 2.2 – Вмикання автостворення масивів

Насправді, як тільки ми увімкнемо автостворення масивів, програма відразу «зламається», бо числовий індикатор не підходить у даному випадку. Тому видаляємо індикатор і створюємо його заново (в контекстному меню тунелю вибираємо **Create→Indicator**). Зверніть увагу на те, що сам тунель зараз має дещо інший вигляд: це вже не синій квадратик, а квадратик з квадратними дужками [], які, власне, і вказують на те, що результатом операції буде масив.

На лицьовій панелі з'явився індикатор-масив, його треба ще розтягнути, аби було видно всі десять елементів масиву (рис. 2.3).

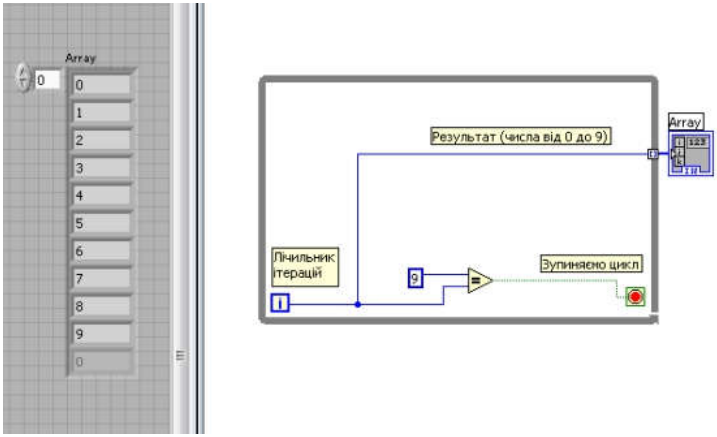


Рис. 2.3 – Приклад циклу While Loop з автостворенням масиву

Загалом все добре і така програма навіть працюватиме. Та насправді для роботи із масивами більше підходить цикл з параметром, в якому кількість ітерацій задана наперед (цикл **For**). У **LabVIEW** такий цикл є і знаходиться він у палітрі **All Functions→Structures→For Loop**.

Цикл із фіксованим числом ітерацій **For Loop** виконує повторювані операції над потоком даних певну кількість разів. Цикл із фіксованим числом ітерацій **For Loop**, розташований на палітрі функцій у розділі **Programmings→Structures**. Значення, привласнене терміналу максимального числа ітерацій «**N**» циклу визначає максимальну кількість повторень операцій над потоком даних. Термінал лічильника ітерацій «**i**» містить кількість виконаних ітерацій. Початкове значення лічильника ітерацій завжди дорівнює нулю.

Цикл з фіксованим числом ітерацій **For** відрізняється від циклу за умовою **While** тим, що завершує роботу, виконавши задане максимальне число ітерацій «**N**». Цикл за умовою – **While** – завершує роботу після виконання заданої умови виходу із циклу.

Термінал «параметр» (позначено буквою **N**) служить для задавання та відображення загальної кількості ітерацій, а термінал «лічильник ітерацій» показує номер поточної (рис. 2.4).

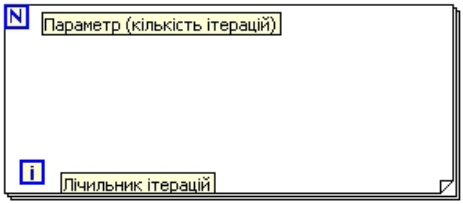


Рис. 2.4 – Приклад циклу з параметром For Loop

Важливо те, що для таких циклів автостворення масивів увімкнено за замовчуванням, тому така програма працюватиме аналогічно попередній, тільки без зайвих маніпуляцій (рис. 2.5).

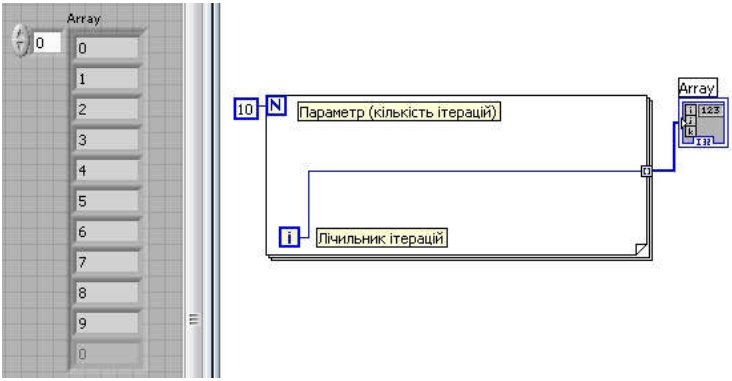


Рис. 2.5 – Приклад створення масиву у циклі For Loop

Зверніть увагу на той факт, що в даному випадку ми вказуємо кількість ітерацій, а не номер останньої. Тобто якщо у першому випадку писали цифру 9, бо нумерація ітерацій починається з 0 і десята має номер 9, то тут треба вказувати 10. Так само легко можна створити і двовимірний масив (таблицю) за допомогою двох вкладених циклів (див. рис. 2.6).

Внутрішній цикл задає кількість рядків, а зовнішній – кількість стовбців. Якщо на вхід циклу подати масив, то кількість ітерацій **N** можна не задавати – вона вибереться автоматично відповідно до кількості даних у масиві. Це називається автоіндексацією.

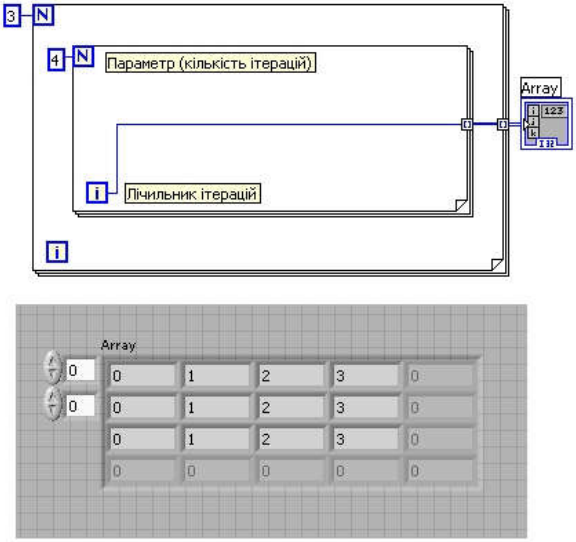


Рис. 2.6 – Створення двовимірного масиву (таблиці)

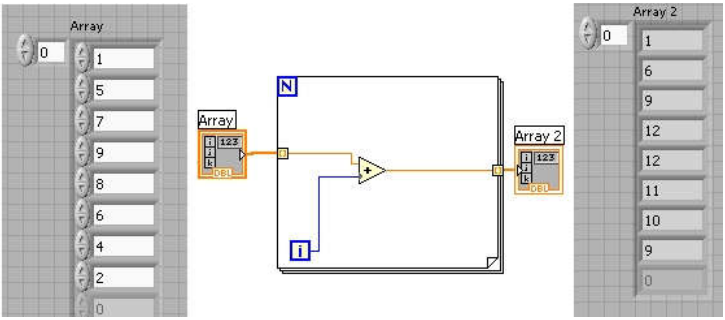


Рис. 2.7 – Автоматичне визначення кількості ітерацій циклу

Тобто така програма працюватиме без підключення терміналу **N**, хоча завжди можна вказати і кількість ітерацій, відмінну від кількості елементів у масиві.

Зауважу ще, що для циклу з параметром теж можна вимкнути автостворення масивів (вимкнеться тоді і автоіндексація), це робиться у контекстному меню тунелю (опція **Disable Indexing**).

Регістри зсуву використовуються при роботі із циклами для передачі значень від поточної ітерації циклу до наступної. Регістр зсуву

створюється клацанням правої клавіші мишки на границі циклу й вибором пункту **Add Shift Register** з контекстного меню.

Регістр зсуву виглядає як пари терміналів, що розташовані безпосередньо один проти одного на протилежних вертикальних сторонах границі циклу. Правий термінал містить стрілку «догори» і зберігає дані по завершенню поточної ітерації **LabVIEW**, передає дані із цього регістру до наступної ітерації.

Регістр зсуву передає будь-який тип даних і автоматично приймає тип перших же даних, переданих на нього. Дані, передані на термінали регістру зсуву, повинні бути одного типу. Передбачено можливість створення декількох регістрів зсуву в одній структурі циклу. До того ж регістр зсуву може мати кілька лівих терміналів для можливості роботи з декількома значеннями попередніх ітерацій.

Регістри зсуву можна використовувати для запам'ятовування значень попередніх ітерацій, що корисно при створенні алгоритмів усереднення. Установка додаткових терміналів регістру зсуву для переносу значень на наступну ітерацію здійснюється клацанням правою кнопкою мишки на лівому терміналі й вибором **Add Element** з контекстного меню. Наприклад, якщо додати два додаткових термінали до лівого термінала регістру зсуву, то значення останніх трьох ітерацій надійдуть на поточну ітерацію.

Щоб ініціалізувати регістр зсуву, необхідно передати на його лівий термінал будь-яке значення ззовні тіла циклу. Якщо не ініціалізувати регістр, цикл використовує значення, записане в регістр під час останнього виконання циклу або значення, використовуване за замовчуванням для даного типу даних, якщо цикл ніколи не виконувався. Наприклад, якщо тип даних регістру зсуву логічний **Boolean**, початкове значення «неправда» – **FALSE**. Точно так само, якщо тип даних регістру зсуву числовий, то початкове значення – 0.

Цикл із ініціалізованим регістром зсуву використовується при кількаразовому запуску віртуального приладу (ВП) для присвоєння вихідному значенню регістру зсуву значення, узятого з останнього виконання ВП. Щоб зберегти інформацію про стан між наступними запусками ВП, варто залишити вхід лівого термінала регістру зсуву невизначеним. На рис. 2.8 наведений приклад, що демонструє роботу регістру зсуву.

На індикаторі **Numeric3** з інтервалом 2 секунди будуть з'являтися числа від 0 до 9, на індикаторі **Numeric2** з тим же інтервалом будуть з'являтися числа від -1 до 8, що відповідають числам індикатора **Numeric3** на попередній ітерації (-1 – початкове значення регістру зсуву), на індикаторі **Numeric** числове значення 9 з'явиться після виконання всього циклу.

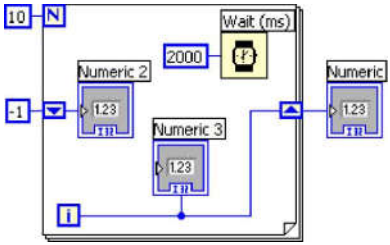


Рис. 2.8 – Приклад циклу з регістром зсуву

Завдання до лабораторної роботи

Завдання 2.1. Створіть ВП, що генерує випадкові числа доти, поки одне з них не виявиться рівним значенню, уведеному в елемент керування. При цьому повинне відображатися кількість ітерацій, виконана циклом.

Завдання виконуємо за наступними кроками.

1. Відкрийте нову лицьову панель. Створіть лицьову панель, розмістивши на ній елементи керування й відображення, як показано нижче на рис. 2.9:

– помістіть на лицьову панель числовий елемент керування, що знаходиться на палітрі **Controls→Numeric**. Назвіть елемент «Задане число для порівняння». Цей елемент задає число, з яким буде проводитися порівняння;

– помістіть на лицьову панель числовий елемент відображення, що знаходиться на палітрі **Controls→Numeric.** Назвіть елемент «Поточне випадкове число». Цей елемент відображає поточне значення, видане функцією «Генератор випадкових чисел»;

– помістіть ще один числовий елемент відображення на лицьову панель. Назвіть елемент «Кількість ітерацій». Цей елемент показує номер поточної ітерації.

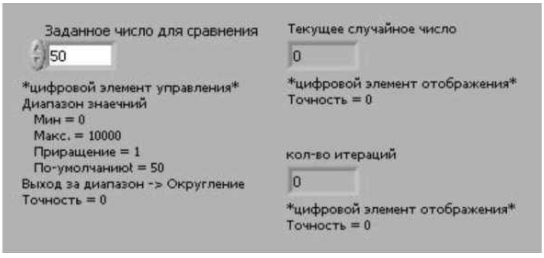


Рис. 2.9 – Лицьова панель з елементами керування

Установка діапазону даних. Щоб значення елемента «Задане число для порівняння» не виходили за рамки діапазону значень, що видаються функцією «Генератор випадкових чисел», варто використовувати діалогове вікно **Data Range**. Виконайте наступні кроки для настроювання діапазону вихідних значень елемента «Задане число для порівняння» від 0 до 10 000 із кроком зміни 1 і значенням, за замовчуванням рівним 50.

2. Клацніть правою кнопкою миші по елементу «Задане число для порівняння». З контекстного меню виберіть пункт **Data Range.** З'явиться діалогове вікно, показане на рис. 2.10.

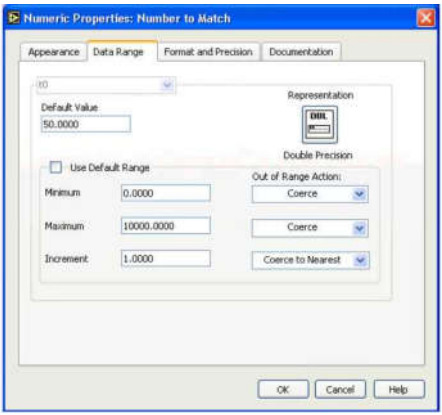


Рис. 2.10 – Приклад діалогового вікна Format&Precision

3. Зніміть виділення з пункту **Use Defaults** (використовувати значення за замовчуванням).

4. Виберіть пункти, показані в діалоговому вікні (див. рис. 2.10), встановивши:

– **Default Value** (значення за замовчуванням) рівним 50;

– **Minimum Value** (мінімальне значення) рівним 0 і виберіть **Coerce**;

– **Maximum Value** (максимальне значення) рівним 10 000 і виберіть **Coerce**;

– **Increment** (значення приросту) рівним 1 і виберіть **Coerce to Nearest.**

5. Виберіть розділ **Format&Precision** (формат і точність). Установка кількості знаків після коми. За замовчуванням **LabVIEW** відображає числові елементи керування й відображення у вигляді десяткових чисел з точністю до двох знаків після коми (3,14). За допомогою опції **Format&Precision** можна змінити точність і вид подання значень елементів (наукова нотація, інженерна нотація, формат часу).

6. Клацніть правою кнопкою миші по елементу «Поточне випадкове число» й виберіть у контекстному меню пункт **Format&Precision.** З'явиться наступне діалогове вікно **Format&Precision**.

7. Зробіть настроювання, показані на рис. 2.11. У поле введення **Digits of Precision** варто ввести значення 0.

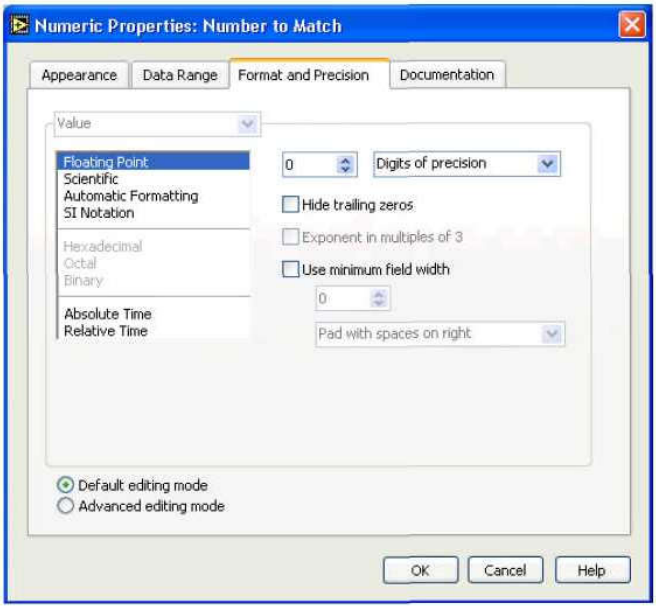


Рис. 2.11 – Настроювання діалогового вікна

8. Повторіть кроки 6 і 7 для елементів відображення «Поточне випадкове число» та «Кількість ітерацій».

9. Створіть блок-діаграму, як це показано на рис. 2.12.

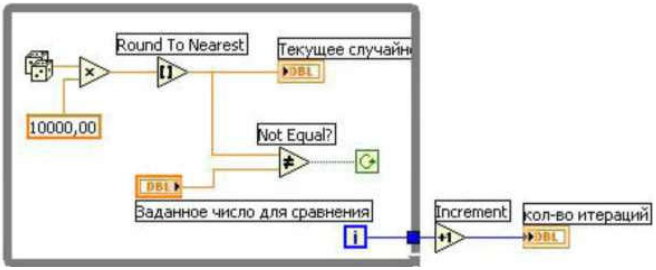


Рис. 2.12 – Блок-діаграма програми

Кроки створення елементів блок-діаграми наведені нижче.

Помістіть на блок-діаграму функцію **Random Number** – «генератор випадкових чисел», розташовану на палітрі функцій у розділі **Function→Numeric.** Ця функція генерує випадкові числа в межах від 0 до 1.

Помістіть на блок-діаграму функцію **Multiply** – «множення», розташовану в палітрі функцій у розділі **Function→Numeric.** Ця функція множить поточне значення з виходу функції **Random Number** на 10 000.

Створіть константу. Для цього варто навести курсор на поле введення даних функції **Multiply**, клацнути по ньому правою кнопкою миші та вибрати в контекстному меню пункт **Create→Constant.** За допомогою інструмента уведення тексту надайте їй значення 10 000.

Помістіть на блок-діаграму функцію **Round To Nearest** – «округлення до найближчого цілого», розташовану в палітрі функцій у розділі **Function→Numeric.** Ця функція буде округляти отримане в межах від 0 до 10000 випадкове число до найближчого цілого числа.

Помістите на блок-діаграму функцію **Not Equal?** (≠), розташовану в палітрі функцій у розділі **Function→ Comparison.** Ця функція призначена для порівняння випадкового числа із числом, введеним в елемент керування Задане число для порівняння. Якщо значення не рівні, функція видає значення **TRUE**.

Помістіть на блок-діаграму цикл **While,** розташований у палітрі функцій у розділі **Function→Structures.** Наведіть курсор на термінал умови виходу, клацніть по ньому правою кнопкою миші й виберіть пункт **Continue if True** – «продовжити якщо істина».

Приєднайте термінал лічильника ітерацій до границі області циклу **While.** На границі циклу з'явиться синій прямокутник. Термінал вихідних даних циклу приєднаний до функції збільшення. При виконанні циклу лічильник ітерацій отримує збільшення, рівне 1. Після завершення циклу значення лічильника ітерацій передається на вихід через термінал виходу циклу. Поза тілом циклу значення лічильника ітерацій збільшується на одиницю для відображення кількості виконаних ітерацій.

Помістіть на блок-діаграму функцію **Increment** (збільшення на 1), розташовану в палітрі функцій у розділі **Function→ Numeric.** Ця функція додає 1 до значення лічильника ітерацій після завершення виконання циклу. Варто звернути увагу, що на терміналі елемента «кількість ітерацій» є сіра крапка, що означає, що **LabVIEW** автоматично здійснює перетворення типу даних лічильника ітерацій до типу даних термінала елемента «кількість ітерацій».

10. Збережіть ВП під ім'ям «Підрахунок ітерацій.Vi».

11. Перейдіть на лицьову панель і змініть значення елемента «Задане число для порівняння»**.**

12. Запустіть ВП. Змініть значення елемента «Задане число для порівняння» й запустіть ВП знову. При цьому елемент «Поточне випадкове число» обновляється після кожної ітерації циклу, тому що його термінал даних розташований усередині тіла циклу. Значення ж елемента «Кількість ітерацій» обновляється після завершення циклу, тому що термінал даних цього елемента розташований поза тілом циклу.

13. Щоб подивитися, як ВП обновляє значення елементів відображення інформації, необхідно запустити ВП у режимі анімації. Для цього варто натиснути на панелі інструментів кнопку **Highlight Execution,** показану ліворуч. Режим налагодження анімує потік даних, що проходять по блок-діаграмі. Таким чином, є можливість спостерігати зміни значень на кожному етапі їхньої генерації.

14. Змініть значення елемента «Задане число для порівняння» таким чином, щоб воно зі збільшенням на 1 виходило за встановлений діапазон значень від 0 до 10 000.

15. Запустіть ВП. **LabVIEW** автоматично приведе нове значення до найближчого значення в зазначеному діапазоні вхідних даних елемента.

16. Закрийте ВП.

Завдання 2.2. Використання регістрів зсуву у циклі **For** для підсумовування елементів масиву.

Завдання виконуємо за наступними кроками.

1. Створіть ВП «Вузол зворотного зв'язку VI.» Лицьова панель ВП показана на рис. 2.13.

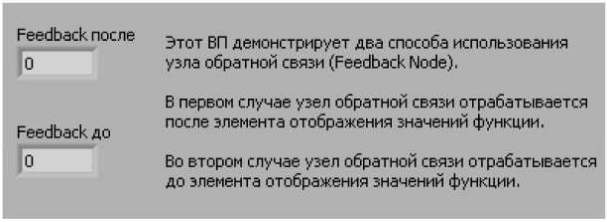


Рис. 2.13 – Лицьова панель ВП

2. Відобразіть блок-діаграму, показану на рис. 2.14. Розмістіть лицьову панель і блок-діаграму на екрані так, щоб вони не пере-

кривали один одного. Перемістіть палітри **Tools** і **Functions,** якщо це необхідно. Одиниця, з'єднана з лівим терміналом циклу **For**, ініціалізує вузол зворотного зв'язку. Таймер **Wait Until Next ms Timer** формує мілісекундну затримку, сповільнюючи процес виконання програми. Для гальмування виконання процесу можна також використовувати режим **Highlight Execution** (анімації виконання блок-діаграми). На цій блок-діаграмі той самий процес виконується двічі, при цьому вузол зворотного зв'язку поміщений у різних місцях з'єднання.

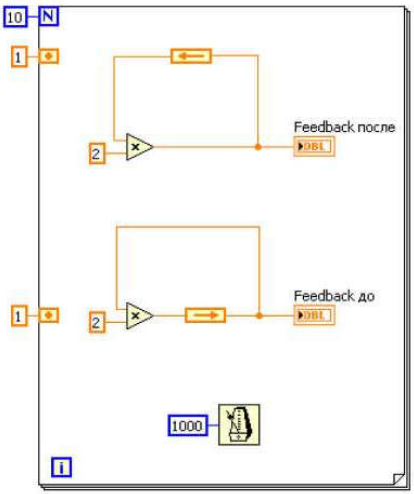


Рис. 2.14 – Блок-діаграма програми

3. Запустіть ВП. Програма у верхній частині спочатку зчитує значення вузла зворотного зв'язку, ініціалізованого значенням 1. Потім це значення передається функції **Multiply** – «множення». Програма в нижній частині спочатку зчитує значення вузла зворотного зв'язку, ініціалізованого значенням 1. Потім це значення передається на цифровий елемент відображення. Функція **Multiply** – «множення» не буде виконуватися до наступної ітерації циклу.

4. Активуйте режим анімації виконання блок-діаграми, натиснувши на показану ліворуч кнопку **Highlight Execution**. Запустіть ВП ще раз для спостереження порядку виконання програми. Відключіть режим анімації для роботи ВП у нормальному режимі.

Замініть вузол зворотного зв'язку регістром зсуву, як показано на наступній блок-діаграмі (див. рис. 2.15):

– виділіть нижній вузол зворотного зв'язку й натисніть клавішу **Delete**, щоб видалити його;

– клацніть правою кнопкою миші по межі циклу й виберіть пункт контекстного меню **Add Shift Register;**

– ініціалізуйте регістр зсуву значенням 1;

– перейменуйте верхній і нижній елементи відображення відповідно «Вузол зворотного зв'язку» та «Регістр зсуву».

5. Запустіть ВП. Зверніть увагу, що вузол зворотного зв'язку й регістр зсуву виконують однакові функції.

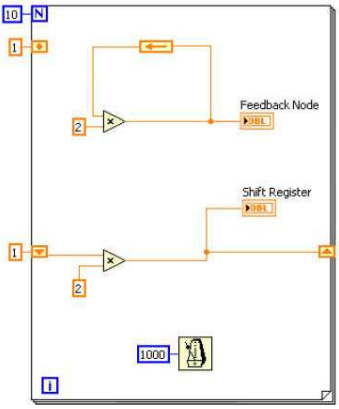


Рис. 2.15 – Блок-діаграма з регістром зсуву

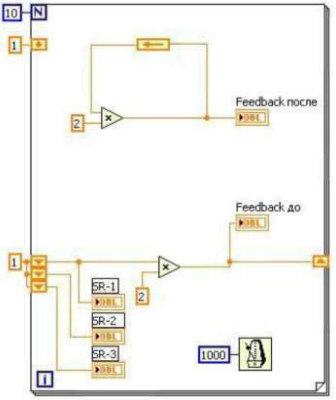


Рис. 2.16 – Блок-діаграма для відображення останніх трьох ітерацій циклу For

6. Модифікуйте регістр зсуву, щоб він відображав останні три ітерації циклу **For**, як показано на наступній блок-діаграмі (див. рис. 2.16):

– змініть розмір лівого регістру зсуву до трьох елементів;

– ініціалізуйте всі елементи регістру зсуву значенням 1;

– клацніть правою кнопкою миші по кожному елементу й виберіть пункт контекстного меню **Create→Indicator**. Назвіть кожний елемент відображення.

7. Запустіть ВП.

8. Закрийте ВП і збережіть його.

Зміст звіту

1. Тема та мета лабораторної роботи.

2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.

3. Опис побудови віртуального приладу.
4. Висновки з роботи.

Контрольні питання

1. Які варіанти конструкції «цикл» використовується в **LabVIEW**?
2. В яких випадках використовується цикл **While**?
3. В яких випадках використовується цикл **For**?
4. В яких випадках використовуються регістри зсуву?

Література: [1–4]

Лабораторна робота 3

Створення простого віртуального приладу «Спектральний аналізатор прямокутного імпульсу»

***Мета роботи*:** ознайомлення з основними вузлами керування та їх функціями при створенні віртуальних приладів.

Порядок виконання

1. Ознайомитись з основними вузлами керування та їх функціями в середовищі **LabVIEW**.

2. Створити віртуальний прилад «Спектральний аналізатор прямокутного імпульсу», згідно вказівок та запустити його в роботу.

3. Скласти звіт і зробити висновок про виконану роботу.

Теоретичні відомості

Основними функціональними вузлами цього віртуального приладу є генератор прямокутного імпульсу та обчислювач спектра потужності.

Генератор прямокутного імпульсу «**Pulse Pattern.vi**» розміщений в палітрі **Functions→Analysis→Signal Generation**. Цей генератор формує на своєму виході **Pulse Pattern** одновимірний масив відліків прямокутного імпульсу із заданою затримкою, шириною та амплітудою. При цьому кількість відліків сигналу задається цілим числом на вході **samples**, тривалість та затримка імпульсу (в кількості відліків) задаються цілими числами на входах **width** та **delay** відповідно, а амплітуда (в умовних одиницях) задається реальним десятковим числом на вході **amplitude**.

Обчислювач спектра потужності «**Power Spectrum.vi**» розміщений в палітрі **Functions→Analysis→Digital Signal Processing.**

1. Створення нового віртуального приладу найкраще почати з лицьової панелі **Windows→Show Front Panel.**

2. Виберіть з підпалітри **Graph** палітри **Control** – осцилограф (**Waveform Graph**) і перенесіть його на лицьову панель. У мітці, що з'явилася введіть назву індикатора «Сигнал».

3. Ще один такий же індикатор виберіть та розмістіть на лицьовій панелі нижче першого індикатора. Введіть в мітку його назву «Спектр». В об'єктному меню цього індикатора зніміть виділення з опції **X Scale→AutoScale X**. За допомогою міткового інструменту введіть кінцеве значення горизонтальної шкали 50.

4. 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 та введіть їх назви в мітки. За допомогою міткового інструменту змініть кінцеві значення регулювання на шкалі регуляторів «Тривалість» і «Затримка» на 50.

5. З підпалітри **Boolean** палітри **Controls** виберіть кнопку «Стоп» – **Rectangular Stop Button** та розмістіть її на лицьовій панелі знизу від регуляторів. Лицьова панель повинна виглядати, як це показано на рис. 3.1.

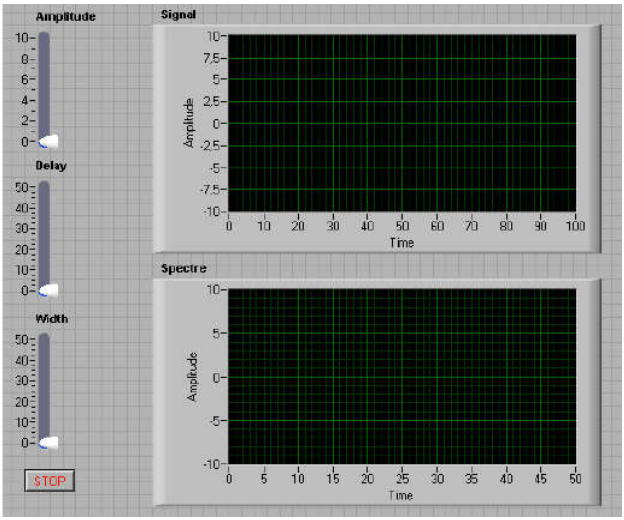


Рис. 3.1 – Лицьова панель приладу

6. З підпалітри **Numeric** палітри **Functions** виберіть числову константу **Numeric Constant** та розмістіть її на блок-діаграмі вище терміналів органів керування. За допомогою міткового інструменту введіть в константу значення 100.

7. З палітри **Functions** виберіть підпалітру **Analysis**, а з неї – підпалітру **signal Generation.** З цієї підпалітри виберіть генератор прямокутного імпульсу «**Pulse_Pattern.vi**» та розмістіть його на блок-діаграмі праворуч від терміналів органів керування.

8. З підпалітри **Analysis** виберіть підпалітру **Digital Signal Processing**. З неї виберіть обчислювач спектра потужності «**Power_Spectrum.vi**» та розмістіть його на блок-діаграмі між генератором прямокутного імпульсу та терміналами індикаторів.

9. З підпалітри **Advanced** палітри **Functions** виберіть функцію **Stop** та розмістіть її на блок-діаграмі поруч з терміналом кнопки **Stop**.

10. З палітри **Tools** виберіть «котушку». З'єднайте між собою термінали органів керування, функцій, констант та індикаторів так, як показано на рис. 3.2.

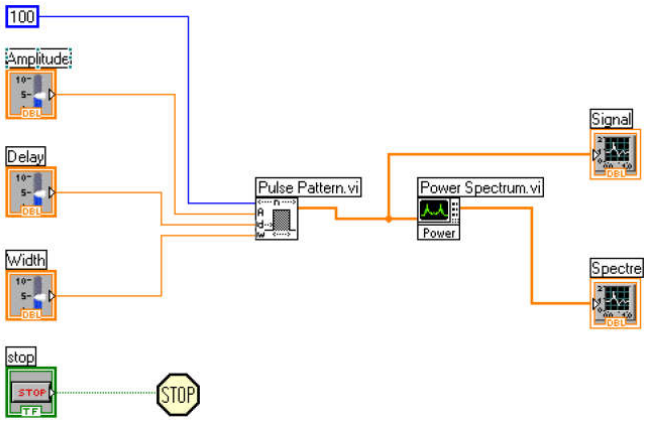


Рис. 3.2 – Блок-діаграма приладу

11. Перейдіть у вікно лицьовій панелі та запустіть віртуальний прилад, натиснувши кнопку **Run Continuously**.

12. Зупинити роботу віртуального приладу можна, натиснувши на кнопку **Stop** на лицьовій панелі або кнопку **Abort Execution** – «Перервати виконання» в лівому верхньому кутку вікна.

При виникненні питань по роботі з **LabVIEW** можна викликати інтерактивну довідку по **LabVIEW**, вибравши з меню **Help→VI, Functions, & How to Help** або **Help→Search the LabView bookshelf**. Крім того, можна викликати інтерактивну довідку щодо практично кожного об'єкта структурної схеми, викликавши **Help→Show context help**.

Завдання до лабораторної роботи

1. Створіть описаний віртуальний прилад «Спектральний аналізатор прямокутного імпульсу».

2. Перегляньте роботу цього приладу, проходження сигналів блок-діаграмою (за допомогою кнопки із зображенням лампочки вгорі вікна структурної схеми).

3. Для звіту зробіть знімки екрану (screenshots) блок-діаграми та лицьовій панелі створеного приладу.

4. Вивчіть основні вузли, органи управління і функції, використані у віртуальному приладі.

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу.
4. Висновки з роботи.

Контрольні питання

1. У чому відмінність програмного пакета **LabVIEW** від інших мов програмування?
2. Поясніть, як ви розумієте сутність принципу потоку даних.
3. Поясніть призначення лицьовій панелі приладу і блок-діаграми приладу.
4. Розкажіть про основні робочих інструментах в **LabVIEW**.
5. Поясніть за блок-діаграмою вашого віртуального приладу призначення його вузлів, функцій, органів керування та індикаторів, порядок роботи віртуального приладу.

***Література*:** [1–3]

Лабораторна робота 4

Моделювання роботи базових елементів цифрової техніки

***Мета роботи*:** вивчення та моделювання роботи найпростіших базових логічних елементів в середовищі **LabVIEW**; ознайомлення з типом даних **Boolean**; створення та використання бібліотеки підпрограм.

Порядок виконання

1. Ознайомитись з відомостями щодо моделювання роботи найпростіших логічних елементів в середовищі **LabVIEW**.
2. Створити програми, що вказані в завданнях 4.1 та 4.2 та запустити їх у роботу.
3. Скласти звіт і зробити висновок про виконану роботу.

Теоретичні відомості

Логічні елементи – це базові блоки цифрових логічних схем. Вони можуть відкриватися або закриватися, дозволяючи або відмовляючи пропускати логічний сигнал. На основі невеликої кількості основних логічних елементів («ТА», «АБО», «ЩО ВИКЛЮЧАЄ АБО», «НЕ») може бути побудована величезна кількість логічних функцій.

Логічний елемент «ТА». Базовий логічний елемент «ТА» складається з двох входів і виходу. Два входи назвемо відповідно A і B. Вихід назвемо Q). Вихід знаходиться в стані «ввімкнено» тільки тоді, коли обидва входи A і B знаходяться в стані «ввімкнено».

У цифровій електроніці стан «ввімкнено» зазвичай представляється у вигляді 1, а стан «вимкнено» у вигляді 0. Співвідношення між вхідними та вихідними сигналами представляють у вигляді таблиці істинності, в якій порівнюються всі можливі стани входів та результуючих виходів. Для логічного елемента «ТА» існують чотири можливі комбінації вхідного стану: A = 0, B = 0; A = 0, B = 1; A = 1, B = 0 та A = 1, B = 1. Ці значення представлені в таблиці істинності в лівому і середньому стовпчиках (див. табл. 4.1). Вихід логічного елемента «ТА» відображений в правому стовпчику.

У середовищі **LabVIEW** можна визначати стан логічного входу перемиканням логічного вимикача, а логічний світлодіодний індикатор може показувати стан виходу. Оскільки в середовищі **LabVIEW** елемент «ТА» є однією з основних вбудованих функцій, то можна легко

створити найпростіший віртуальний прилад, що демонструє роботу цього логічного елемента, приєднавши два вимикача до його входу та світлодіодний індикатор до його виходу.

Таблиця 4.1 – Таблиця істинності логічного елемента «ТА»

A	B	Q=A ТА B
0	0	0
0	1	0
1	0	0
1	1	1

Створений віртуальний прилад моделює роботу елемента «ТА». Використовуйте для лицьової панелі приладу елементи з групи **Boolean**, для блок-діаграми елемент **AND** з групи **Boolean**. Натискаючи на дві вхідні кнопки, спостерігайте зміни вихідного індикатора. Перевірте таблицю істинності.

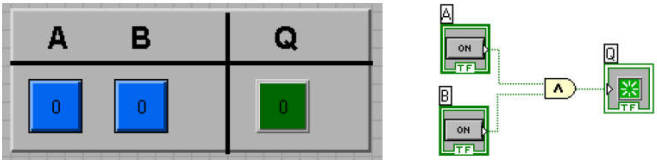


Рис. 4.1 – Функція «ТА», приєднана до вхідних та вихідного терміналів

Аналогічним чином може бути промодельована робота елементів «ТА», «АБО», «НЕ», «ЩО ВИКЛЮЧАЄ АБО» тощо.

У пакеті **LabVIEW** містяться всі основні двовходові логічні елементи, але можна використовувати і більше входів. З двох двовходових елементів можна побудувати віртуальний прилад, який реалізує елемент «ТА» з трьома входами.

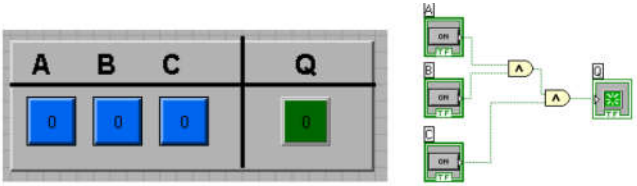


Рис. 4.2 – Модель логічного елемента «ТА» з трьома входами

Підпрограми. Будь-яка програма (віртуальний прилад – VI) може бути використана в блок-діаграмі іншої програми як її складова час-

тина. Іншими словами, вона може бути, вкладена як підпрограма, **SubVI**. Ця особливість дозволяє основній, головній програмою бути модульною, краще читатись і бути простішою для розуміння.

Для вставки раніше розробленого ВП (**SubVI**) в програму вищого рівня необхідно використовувати опції **Select a VI ...** в палітрі **Function**. У відповідь на запит діалогового вікна необхідно вибрати файл підпрограми і встановити її на діаграмі основної програми (рис. 4.3).

Рис. 4.3 – Робота з підпрограмами через елемент Select a VI

Головна програма може мати безліч викликів підпрограми. На діаграмі головної програми підпрограма з'явиться у вигляді кубика із позначкою, заданою за замовчуванням. За допомогою подвійного клацання можна відкрити цю підпрограму, а в разі потреби і провести деякі налаштування. Вище була створена програма, що моделює роботу логічного елемента «ТА». Тепер з цієї програми створимо повнофункціональну підпрограму. Зовнішній вигляд лицьової панелі та блок-діаграми підпрограми показаний на рис. 4.4.

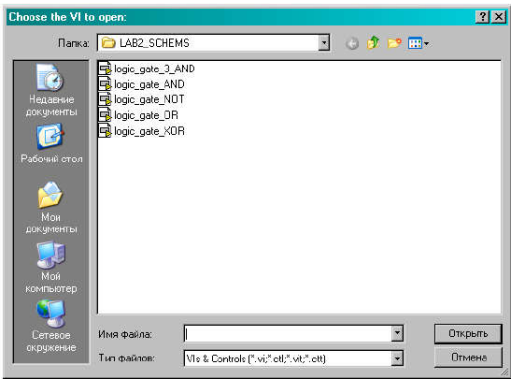


Рис. 4.4 – Діалогове вікно вибору підпрограми

Просте включення кубика підпрограми нічого не дасть головній програмі, тому що немає можливості щось передати та отримати назад. Щоб підпрограма отримувала дані з основної програми та передавала їх назад в основну програму необхідно виконати певні дії.

Спочатку слід відкрити лицьову панель нашої підпрограми, і, розмістивши курсор у правому верхньому кутку, на іконці, викликати контекстне меню (рис. 4.5).

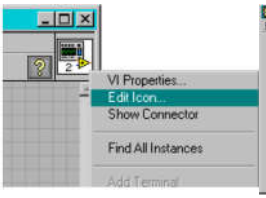


Рис. 4.5 – Вибір режиму редагування іконки підпрограми

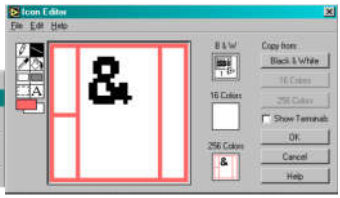


Рис. 4.6 – Редактор зображення іконки

Рис. 4.7 – Іконка програми в режимі Show Connector

Вибравши в цьому меню пункт **Edit Icon**, можна відредагувати зовнішній вигляд іконки створюваної підпрограми (рис. 4.6). Далі знову слід викликати контекстне меню (див. рис. 4.5) і вибрати пункт **Show Connector**. Через цей пункт відкривається доступ до параметрів підпрограми – конекторам або з'єднувачів, елементам взаємодії підпрограми із зовнішнім світом. Конектор передає і приймає дані від програми, що викликається. Відразу після вибору цього пункту вид іконки нашої програми зміниться (рис. 4.7), а курсор набуде вигляду сполучної котушки.

LabVIEW подбає про те, щоб з'явилися конектори – квадратики і прямокутник – по числу елементів управління і 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 та клацнути по конектор-квадратику, за яким він буде закріплений. Ознакою під'єднання буде зміна кольору цих квадратиків. В даному прикладі квадратики зліва – це логічні сигнали A і B, прямокутник праворуч – результат логічного множення Q.

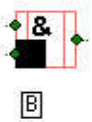


Рис. 4.8 – Елемент підпрограми головної програми, що викликається

Тепер підпрограма, після її установки в головну програму, дасть можливість бачити вхідні та вихідні змінні (рис. 4.8).

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 ви-

брати кнопку **New VI Library**, вказавши після натискання цієї кнопки ім'я нової бібліотеки, а далі зберегти поточний файл у створеній бібліотеці.

Завдання до лабораторної роботи

Завдання 4.1. Створіть віртуальний прилад, що демонструє роботу основних логічних елементів, лицьова панель приладу повинна бути виконана приблизно так, як це показано на рис. 4.9.

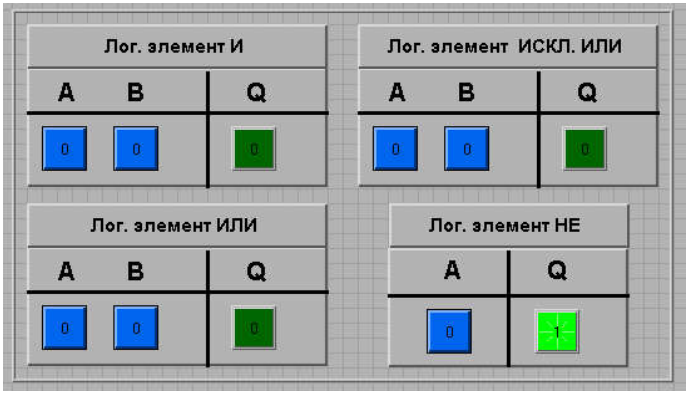


Рис. 4.9 – Можливий вигляд лицьової панелі приладу

Для звіту збережіть знімок екрана (screenshot) лицьової панелі приладу та блок-діаграми приладу. Розроблений віртуальний прилад збережіть на своєму носії інформації для його демонстрації при захисті лабораторної роботи.

Завдання 4.2. Оформіть створені віртуальні прилади – «ТА», «АБО», «НЕ», «ЩО ВИКЛЮЧАЄ АБО» у вигляді підпрограм і збережіть їх в бібліотеці файлів (ця бібліотека знадобиться у наступних лабораторних роботах, тому збережіть її на своєму носії інформації).

Завдання 4.3. Реалізуйте логічну функцію:

$$Y = \left(A \cdot \overline{B}\right)\left(\overline{A} \cdot B\right). \qquad (4.1)$$

При її реалізації використовувати тільки **SubVI**, створені в ході виконання завдання 4.2. Лицьова панель приладу повинна містити два перемикачі і один елемент індикації типу **Boolean**.

Для звіту збережіть знімок екрана (screenshot) лицьовій панелі приладу і блок-діаграми приладу. Розроблений віртуальний прилад збережіть на своєму носії інформації для його демонстрації при захисті лабораторної роботи. Визначте, яку з вивчених найпростіших логічних функцій реалізує цей прилад.

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу.
4. Висновки з роботи.

Контрольні питання

1. У цій роботі всі віртуальні прилади оперують з типом даних **Boolean**, що це за тип даних?
2. Навіщо потрібно створювати підпрограми **SubVI**? Які переваги вони дають?
3. Які з базових логічних функцій вже реалізовані в **LabVIEW**?
4. Навіщо потрібні бібліотеки підпрограм? Чи можна без них створювати віртуальні прилади?
5. Скільки разів основна програма може мати в своєму складі викликів підпрограм?
6. Чи може основна програма при виклику підпрограми передавати туди будь-які дані та отримувати їх назад?
7. На структурних схемах приладів в цій лабораторній роботі ви бачили з'єднувальні дроти зеленого кольору, що він позначає в **LabVIEW**?

***Література*:** [2, 3, 5]

Лабораторна робота 5

Моделювання роботи комбінаційних цифрових пристроїв

Мета роботи: вивчення та моделювання роботи комбінаційних цифрових пристроїв у середовищі **LabVIEW**: шифратор, дешифратор, мультиплексор, демультиплексор; створення бібліотеки підпрограм цих елементів; вивчення числових типів даних; конвертація одного типу даних в інший; використання масивів і кластерів; вивчення структури **Case**.

Порядок виконання

1. Ознайомитись з відомостями щодо моделювання роботи комбінаційних цифрових пристроїв у середовищі **LabVIEW**: шифратор, дешифратор, мультиплексор, демультиплексор.
2. Створити бібліотеки підпрограм даних елементів; вивчення числових типів даних; конвертація одного типу даних в інший; використання масивів і кластерів; вивчення структури **Case**.
3. Скласти звіт і зробити висновок про виконану роботу.

Теоретичні відомості

Логічні пристрої розділяють на два класи: комбінаційні й послідовні. Пристрій називають комбінаційним, якщо його вихідні сигнали в деякий момент часу однозначно визначаються вхідними сигналами, що мають місце в 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 тільки сигналами, наявними на входах у цей момент часу, але й станом елементів пам'яті. Таким чином, реакція послідовного пристрою на певні вхідні сигнали залежить від передісторії його роботи.

Дешифратором називається комбінаційний пристрій, що перетворить n-розрядний двійковий код у логічний сигнал, що з'являється на тому виході, десятковий номер якого відповідає двійковому коду. Число входів та виходів у так званому повному дешифраторі зв'язаний співвідношенням $m = 2 \cdot n$, де n – число входів, а m – число виходів.

Роботу дешифратора із трьома входами та вісьма виходами можна представити наступною таблицею істинності (табл. 5.1).

Таблиця 5.1 – Таблиця істинності дешифратора

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

Для створення подібного віртуального дешифратора в **LabVIEW** розташуєте на лицьовій панелі три перемикачі та вісім світлодіодних індикаторів (рис. 5.1). Перемикачі будуть моделювати вхідний цифровий код, що надходить на дешифратор, а світлодіодні індикатори – вихідний сигнал з дешифратора.

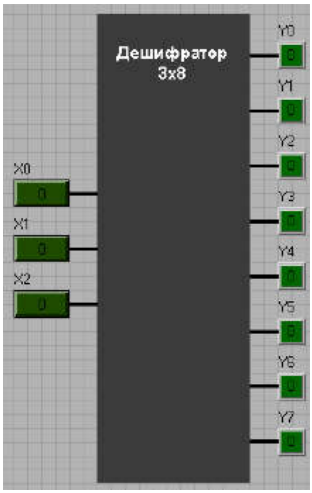


Рис. 5.1 – Лицьова панель віртуального приладу «Дешифратор 3×8»

Реалізація структурної схеми дешифратора можлива декількома способами. Перший спосіб – створити структурну схему даного приладу, ґрунтуючись на базових логічних елементах вивчених у лабораторній роботі 4 та таблиці істинності цього пристрою (див. табл. 5.1).

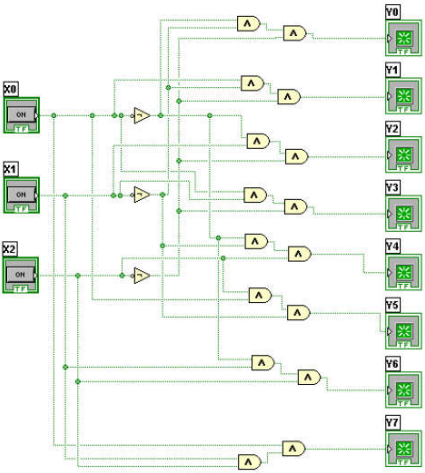


Рис. 5.2 – Структурна схема на логічних елементах віртуального приладу «Дешифратор 3×8»

Другий спосіб – при створенні структурної схеми приладу використовувати керуючі структури **LabVIEW Case Structure** (рис. 5.3).

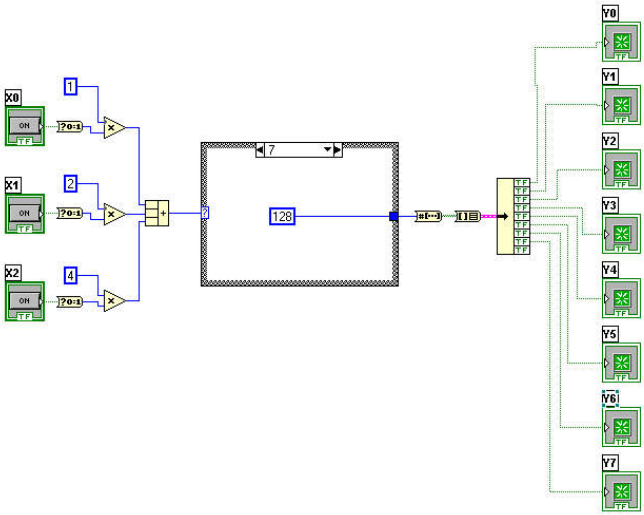


Рис. 5.3 – Структурна схема на логічних елементах віртуального приладу «Дешифратор 3×8» з використанням структури Case

Структура **LabVIEW** «Варіант» – **Case Structure** – керує виконанням одного із двох або більше фрагментів коду та при виборі за умовою аналогічна операторові **If-Then-Else** текстових мов програмування, а при виборі за значенням числової або строкової змінної аналогічна операторові **Case**. Структура може містити одну або більше піддіаграмумов, які будуть працювати при виконанні умов, заданих користувачем.

На структурній схемі, що представлена на рис. 5.3, сигнал із вхідних терміналів за допомогою функціонального вузла **Boolean to (0,1)** (рис. 5.4, *а*) конвертується в числовий тип, далі за допомогою вузла **Multiply** (рис. 5.4, *б*) множиться на вагову константу біта вхідного сигналу та в блоці **Compound Arithmetics** (рис. 5.4, *в*) додається.

а *б* *в*

Рис. 5.4 – Вхідні компоненти та дешифратори: *а*) функціональний вузол Boolean to (0,1); *б*) блок арифметичного множення Multiply; *в*) блок складальної арифметики Compound Arithmetic

На вхід структури **Case** подається ціле двійкове число, що закодоване бітами вхідних терміналів приладу.

Далі сигнал надходить у структуру **Case**, у якій задано вісім умов, для кожного з восьми можливих чисел на вході. На вихід структури надходить двійкове число, що відповідає відображенню у вісьмох вихідних бітах для одного з випадків вхідної комбінації біт.

Для того, щоб відобразити це число на світлодіодних індикаторах, воно з двійкового подання числа перетворюється в одновимірний масив біт (рис. 5.5, *а*), що перетвориться в кластер, що складається з 8 біт (рис. 5.5, *б*), далі в блоці **Unbundle** (рис. 5.5, *в*) кластер розбивається на його складові потоки біт, які надходять на вихідні світлодіодні індикатори.

а *б* *в*

Рис. 5.5 – Вихідні компоненти та дешифратори: *а*) Number to boolean array; *б*) Array to cluster; *в*) Unbundle

Хоча в цьому випадку другий спосіб створення дешифратора виявляється більш складним, він дає можливість зрозуміти основні

прийоми роботи з логічними та цілими типами даних, способи їхнього конвертування з одного типу в інший; також тут вводиться поняття масиву та кластера і прийоми роботи з керуючою структурою «Варіант». Аналогічним образом можна створити віртуальні прилади та шифратор, мультиплексор і демультиплексор.

Завдання до лабораторної роботи

Завдання 5.1. Отримайте у викладача віртуальний прилад дешифратор, описаний вище. Вивчіть роботу даного приладу для обох випадків його реалізації. Вивчіть призначення всіх нових функціональних вузлів, використаних у структурній схемі та проаналізуйте принципи їхнього функціонування.

Для звіту зробіть знімки екрана (screenshot) лицьової панелі та структурної схеми приладу. У звіті також дайте опис функціональних вузлів, представлених на структурній схемі приладу.

Завдання 5.2. Створіть віртуальні прилади – шифратор, мультиплексор і демультиплексор. При створенні шифратора використовуйте перший спосіб реалізації (описаний вище). При створенні мультиплексора використовуйте другий спосіб реалізації. При створенні демультиплексора використовуйте обидва способи реалізації або запропонуйте свій спосіб.

Завдання 5.3. Оформіть всі створені віртуальні прилади – дешифратор, шифратор, мультиплексор, демультиплексор у вигляді підпрограм і збережіть їх у бібліотеці файлів (ця бібліотека знадобиться у наступних лабораторних роботах, тому збережіть її на своєму носії інформації).

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу.
4. Висновки з роботи.

Контрольні питання

1. З якими типами даних ви оперували в даній роботі?
2. Який тип даних позначається на рис. 5.3 синіми провідниками?
3. Який тип даних позначається на рис. 5.3 жирним рожевим провідником?

4. Який тип даних позначається на рис. 5.3 жирним зеленим провідником?

5. Яке призначення функціонального вузла **Multiply**? Яке призначення функціонального вузла **Compound Arithmetic**?

6. Яке призначення функціонального вузла **Unbundle**? Яке призначення функціонального вузла **Array to cluster**?

7. Яке призначення функціональних вузлів **Boolean to (0,1)**, **Number to Boolean array**?

8. Що ви знаєте про структуру **Case structure**?

9. Для якого типу даних робили вибір умов для дешифратора в даній роботі структура **Case**? Скільки умов було реалізовано в структурі **Case**?

***Література*:** [2–4]

Лабораторна робота 6

Дослідження функцій та побудова графіків у середовищі LabVIEW

***Мета роботи*:** вивчити елементи графічного відображення інформації та режими їхньої роботи.

Порядок виконання

1. Ознайомитись з теоретичними відомостями.
2. Відкрити програмний пакет **LabVIEW**. Створити новий документ – **New VI**. Вивчити елементи й функції графічного відображення інформації.
3. Відповідно до варіанта (див. табл. 6.1) створити віртуальний прилад для відображення заданої функції на графіках **Waveform Chart, Waveform Graph, XY Graph**.
4. Дослідити режими оновлення відображуваних даних на графіках і розгорненнях.
5. Скласти звіт і зробити висновок про виконану роботу.

Теоретичні відомості

У **LabVIEW** є різні види графіків. У деяких, найпростіших, практично все зроблено за користувача. Тому залишилося лише викликати з набору приладів цей графік і підключити його до виходу своєї програми. Крива на графіку є графічним відображенням залежності величини Y від величини X. Часто величина Y представляє значення даних, тоді як величина X – час [1].

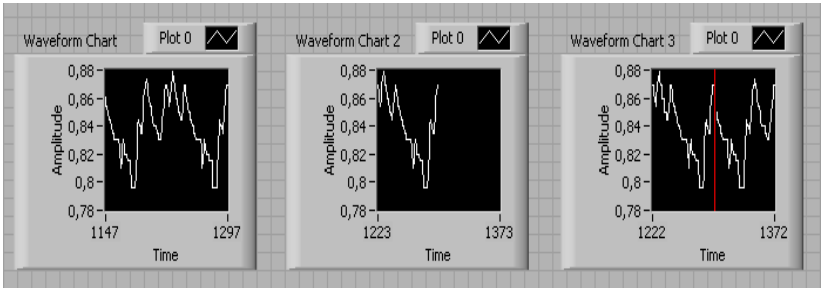


Рис. 6.1 – Режими відновлення розгортання осцилограми

Розгорнення осцилограми **Waveform Chart** є особливим числовим елементом відображення, що може зобразити в графічному вигляді одну або кілька кривих. Найбільше часто розгорнення осцилограм використовуються всередині циклів. У них зберігаються й відображаються на постійно відновлюваному дисплеї дані, які були отримані раніше, а також нові дані в міру їхнього надходження. У розгорненні осцилограми величина Y являє собою дані, створювані під час ітерації циклу, а X – час виконання циклу. Розгорнення осцилограми має три режими відновлення: панорамне розгорнення **strip chart mode**, часове розгорнення **scope chart mode** і часове розгорнення з маркером **sweep chart mode**, як зображено на рис. 6.1.

При роботі розгорнення в панорамному режимі в осцилограмі з'являється зображення, що прокручується подібно до паперової стрічкової діаграми. Осцилограми в режимах часового розгорнення й часового розгорнення з маркером мають зображення зі зворотним ходом, як в осцилографі. У режимі часового розгорнення крива, досягши правої межі, стирається й знову починає прорисовуватися з лівої межі. Режим часового розгорнення з маркером працює аналогічно, але зображення не зникає при досягненні правої межі: початок надходження нових даних зазначає вертикальна лінія, що рухається, – маркер.

При підключенні скалярної величини до терміналу розгорнення на блок-діаграмі на кожній ітерації циклу рисується одна точка на розгорненні осцилограми. Для побудови багатопроменевих розгорнень осцилограм дані необхідно об'єднати (функція **Bundle**). Для зручного подання даних можна сполучати графіки або використати кілька шкал X та Y.

На відміну від розгорнень, графіки відразу відображають сформовані масиви даних. Вони не можуть додавати нові значення до вже створеного. **LabVIEW** пропонує кілька видів графіків: графіки осцилограм (**Waveform Graph**), графіки двох координат (**XY Graph**), графіки інтенсивності, тривимірні графіки, графіки цифрових осцилограм і деякі особливі види графіків (криві Сміта, графіки в полярних координатах, криві максимумів-мінімумів і криві розподілу). На лицьовій панелі графіки осцилограм і графіки XY мають ідентичний вигляд, але зовсім різні типи вхідних даних і функції.

Графік осцилограми рисує тільки однозначні функції (одне значення Y відповідає певному значенню X) з однорідно-розташованими точками. На рис. 6.2 зображено використання графіка осцилограми для побудови однопроменевої та багатопроменевої осцилограм.

Графік XY – це декартовий графік, що використовується для відображення масивів даних з тимчасовими інтервалами, що зміню-

ються, для даних з декількома значеннями координати Y для кожного значення X, наприклад, графік кола.

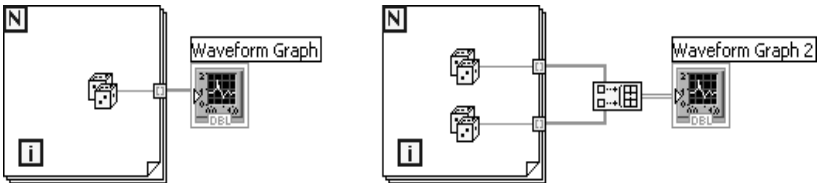


Рис. 6.2 – Побудова одно- та багатопроменевої осцилограм

Однопроменева осцилограма XY і відповідний термінал на блок-діаграмі зображені на рис. 6.3 та 6.4.

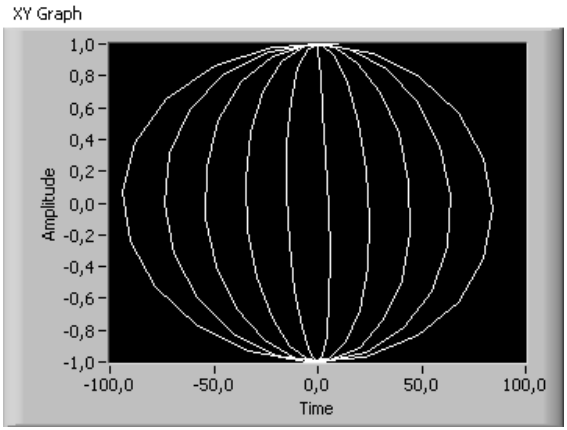


Рис. 6.3 – Однопроменева осцилограма XY

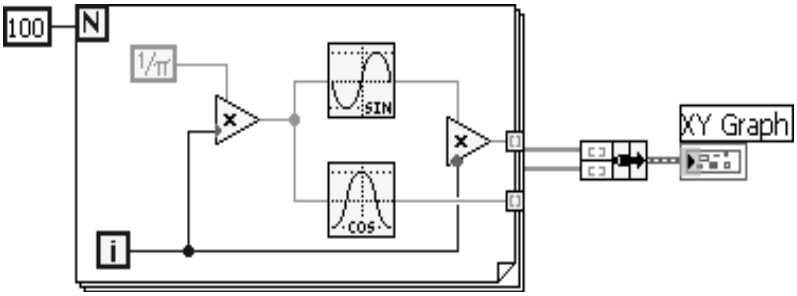


Рис. 6.4 – Блок-діаграма однопроменевої осцилограми XY

Розгорнення й графіки інтенсивності **Intensity charts** та **graphs** відображають дані в трьох вимірах на площині шляхом використання кольору як третього виміру даних і є необхідними інструментами для відображення таких даних, як топографічні або температурні карти.

Три типи інструментів для побудови тривимірних графіків:

3D-графік поверхні викреслює просту поверхню, що визначена матрицею *z*. Поверхня будується за рахунок зміни векторів *x* та *y*. На вхід даної функції необхідно подавати двовимірний масив і два необов'язкові одновимірні масиви;

3D-параметричний графік викреслює поверхню на основі площин *x*, *y* та *z*. На вхід даної функції необхідно подавати три двовимірні масиви або матриці, які визначають кожну із площин *x*, *y* та *z*;

3D-графік кривої описує лінію на основі точок *x*, *y* та *z*. На вхід даної функції необхідно подавати три одновимірні масиви, які визначають кожну точку на графіку.

Для відображення й аналізу цифрових сигналів («істинно» або «хибно», 0 або 5В, «ввімкнено» або «вимкнено») у **LabVIEW** використовується спеціальний тип графіка – графік цифрової осцилограми **Digital Waveform Graph**.

Завдання до лабораторної роботи

Завдання 6.1. Створити програму, яка виводить на графіки значення за варіантами завдань (табл. 6.1).

Таблиця 6.1 – Варіанти завдань

Номер варіанта	Завдання	Номер варіанта	Завдання
1	$y = \tan 2x$	6	$y = \sqrt{1 - \cos^2 x}$
2	$y = \cos\left(\frac{\pi}{2} + 2x\right)$	7	$y = \sin\frac{x}{2}$
3	$y = \frac{1}{3}\operatorname{ctg} x$	8	$y = -\tan(1 - x)$
4	$y = \cos^2 x$	9	$y = \frac{1}{\cos x + \sin x}$
5	$y = \sin\frac{1}{x+2}$	10	$y = \cos(\arccos x)$

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу.
4. Висновки з роботи.

Контрольні питання

1. Які у **LabVIEW** є елементи графічного відображення інформації?
2. Що являє собою розгорнення осцилограми?
3. Які є режими відображення розгорнення осцилограми?
4. У чому полягають функціональні відмінності розгорнень та графіків осцилограм?
5. Для чого використовуються графіки й розгорнення інтенсивності, тривимірні графіки?

***Література*:** [1, 2, 4]

Лабораторна робота 7

Визначення струму в ланцюзі з використанням структурного вузла Formula Node

***Мета роботи*:** визначити сили струму в замкненому колі за законами Ома та Кірхгофа за допомогою структурного вузла **Formula Node**.

Порядок виконання

1. Ознайомитись з теоретичними відомостями.
2. Відкрити програмний пакет **LabVIEW**. Створити новий документ – **New VI**. Вивчити елементи та функції математичного опису.
3. Відповідно до варіанта створити віртуальний прилад для визначення сили струму в замкненому колі за законами Ома та Кірхгофа за допомогою структурного вузла **Formula Node**.
4. Оформити звіт з виконаної роботи.

Теоретичні відомості

При розробленні блок-діаграми програми відразу закладається алгоритм її роботи, порядок виконання операцій. У **LabVIEW** розміщені поруч (але не зв'язані між собою) програмні блоки, виконуються одночасно. Тому в тих випадках, коли важлива послідовність операцій, програмістові необхідно визначити порядок обчислень і закласти його в структуру програми.

У **LabVIEW** є розділ **Structures**, що дозволяє регламентувати цей процес. Виклик структур здійснюється з функціонального набору **Functions** меню **Structures** (рис. 7.1).

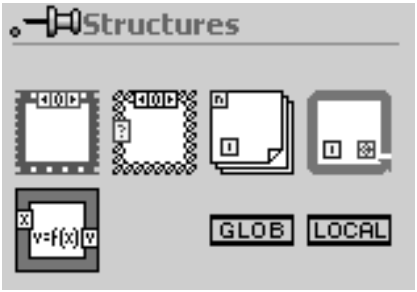


Рис. 7.1 – Виклик меню Structures з функціональної панелі

Цикл з фіксованим числом ітерацій **For Loop** (рис. 7.2) виконує код розміщений в його межах деяке число ітерацій **count**. Це число дорівнює величині, введеній у термінал числа ітерацій (**count terminal**).

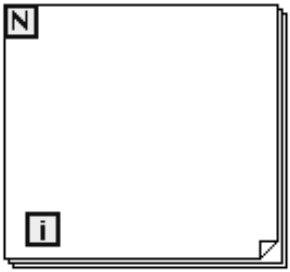


Рис. 7.2 – Цикл із фіксованим числом ітерацій For Loop

Цикл за умовою **While Loop** (рис. 7.3) виконує код, розміщений в його межах, доти, доки логічне значення **Boolean value**, підключене до терміналу умови виходу із циклу **conditional terminal**, не перейде в стан **False**. **LabVIEW** перевіряє термінал умови виходу по закінченні кожної ітерації. Якщо значення відповідає **True**, то виконується наступна ітерація. За замовчуванням термінал умови виходу перебуває в стані **False**. Якщо залишити його не підключеним, цикл виконуватися не буде.

Термінал лічильника ітерацій – **iteration terminal** – циклу за умовою поводиться так само, як і у випадку із циклом з фіксованим числом ітерацій.

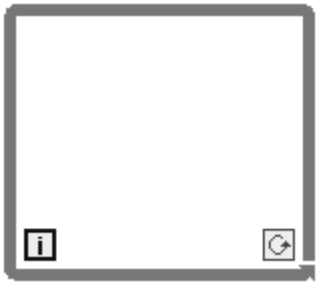


Рис. 7.3 – Цикл за умовою While Loop

Дані надходять до циклу і виходять із нього через маленьке вікно на межі циклу, що називається точкою входу/виходу до структури – тунель – **tunnel**. Оскільки **LabVIEW** працює відповідно до прин-

ципу потоку даних, то перш, ніж цикл почне виконуватися, вхідні точки повинні передати до нього свої дані. Вихідні точки циклу виводять дані лише після завершення всіх ітерацій.

Символи, що використовуються в математичних операціях, наведені у таблиці 7.1

Таблиця 7.1 – Стандартизовані символи для математичних операцій

Символ	Значення
**	Піднесення до степеня
+,-,!,++,--	Унарний плюс, унарний мінус, логічне НІ, перед- та постінкремент, перед- та постдекремент
*, /, %	Множення, ділення, модуль (залишок)
+ та -	Додавання та віднімання
>> та <<	Арифметичне зміщення вправо та вліво
>, <, >=, <=	Більше, менше, більше або рівне, менше або рівне
&	Бітове І
^	Бітове виключне АБО
&&	Логічне І
!= та ==	Нееквівалентне або еквівалентне
!	Бітове АБО
!!	Логічне АБО
?	Умовний вираз

Завдання до лабораторної роботи

Завдання 7.1. Створити програму, для знаходження сили струму в замкненому колі (рис. 7.4) використовуючи закони Ома та Кірхгофа, відповідно до варіантів завдань (табл. 7.2).

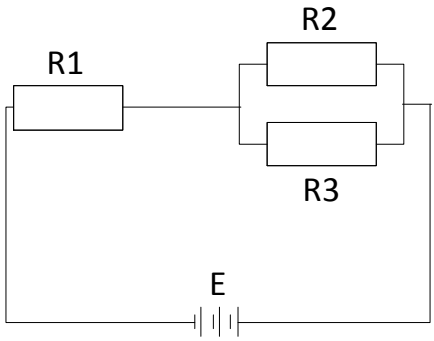


Рис. 7.4 – Електрична схема кола

Таблиця 7.2 – Варіанти завдань

Номер варіанта	Завдання	Номер варіанта	Завдання
1	$E = 100$ В; $R_1 = 10$ Ом; $R_2 = 11$ Ом; $R_3 = 12$ Ом	6	$E = 160$ В; $R_1 = 16$ Ом; $R_2 = 17$ Ом; $R_3 = 18$ Ом
2	$E = 120$ В; $R_1 = 12$ Ом; $R_2 = 13$ Ом; $R_3 = 14$ Ом	7	$E = 170$ В; $R_1 = 17$ Ом; $R_2 = 18$ Ом; $R_3 = 19$ Ом
3	$E = 130$ В; $R_1 = 13$ Ом; $R_2 = 14$ Ом; $R_3 = 15$ Ом	8	$E = 180$ В; $R_1 = 18$ Ом; $R_2 = 19$ Ом; $R_3 = 20$ Ом
4	$E = 140$ В; $R_1 = 14$ Ом; $R_2 = 15$ Ом; $R_3 = 16$ Ом	9	$E = 190$ В; $R_1 = 19$ Ом; $R_2 = 20$ Ом; $R_3 = 21$ Ом
5	$E = 150$ В; $R_1 = 15$ Ом; $R_2 = 16$ Ом; $R_3 = 17$ Ом	10	$E = 200$ В; $R_1 = 20$ Ом; $R_2 = 21$ Ом; $R_3 = 22$ Ом

Розрахунок за цими формулами можна виконати за допомогою формульного вузла **Formula Node**, який відноситься до елементів **Structures** і викликається правою клавішею миші на панелі блок-діаграм по шляху: **Programming→Structures→Formula Node** (рис. 7.5).

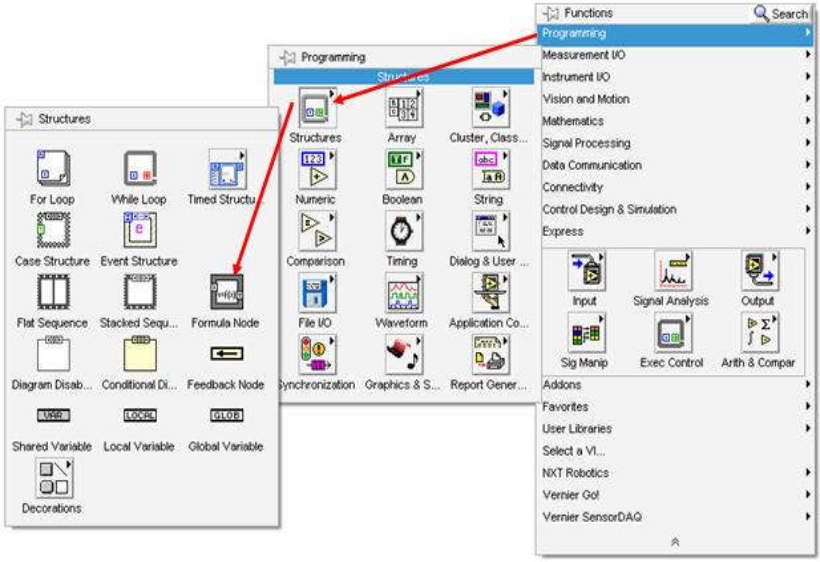


Рис. 7.5 – Вузол Formula Node

Отримана рамка формульного вузла розтягується до потрібного розміру і в неї вписуються розрахункові формули.

Невідомі записуються в лівій частині формул. Кожна формула пишеться на окремому рядку і закінчується крапкою з комою.

Потім в формули потрібно внести вихідні дані і вивести результати розрахунку. Для цього курсор встановлюється правою клавішею миші на рамці формульного вузла і з спливаючого меню лівою клавішею викликається **Add Input** – додати вхід – для вхідних величин і **Add Output** – додати вихід – для вихідних величин. У рамки, що з'явилися, вписуються найменування цих величин (рис. 7.6 та 7.7).

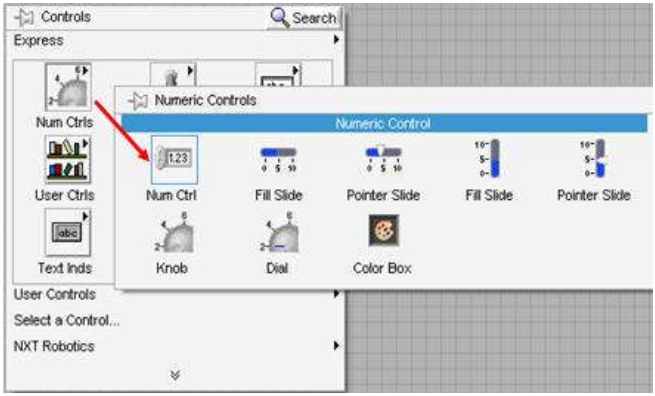


Рис. 7.6 – Типи цифрових керуючих елементів

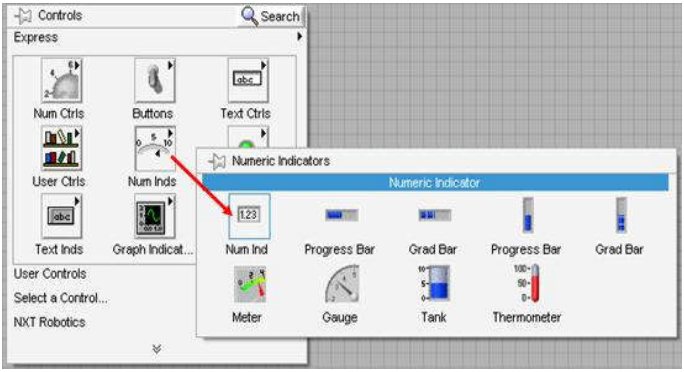


Рис. 7.7 – Типи цифрових індикаторів

До вхідних рамок підключаються цифрові керуючі елементи, до вихідних індикатори.

Входи і виходи можна встановлювати в будь-якому місці рамки. Найменування в рамках повинні бути точно такими ж, як у формульному вузлі. Допускається застосування одного і того ж найменування для вхідної і вихідної величини. За допомогою керуючих елементів задаються вихідні дані, після чого схема запускається на виконання.

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу.
4. Висновки з роботи.

Контрольні питання

1. Яким чином можна визначити послідовність виконання програми?
2. Що таке цикл?
3. Як працює цикл за умовою **While Loop**?
4. Як працює цикл із фіксованим числом ітерацій **For Loop**?
5. Як працює структурний вузол **Formula Node**.

***Література*:** [2, 3, 5]

Лабораторна робота 8

Використання циклу з фіксованим числом ітерацій та вузла формул

***Мета роботи*:** скласти блок-діаграму для моделювання напруги, струму і потужності в ланцюзі синусоїдального струму із застосуванням формульного вузла та циклу за завданням.

Порядок виконання

1. Ознайомитись з теоретичними відомостями.
2. Відкрити програмний пакет **LabVIEW**. Створити новий документ **New VI**. Вивчити елементи та функції математичного опису.
3. Відповідно до варіанта створити віртуальний прилад для визначення сили струму, миттєвої напруги та потужності за відомими формулами.
4. Оформити звіт з виконаної роботи.

Теоретичні відомості

При розробленні блок-діаграми програми відразу закладається алгоритм її роботи, порядок виконання операцій. У **LabVIEW** розміщені поруч (але не зв'язані між собою) програмні блоки, виконуються одночасно. Тому в тих випадках, коли важлива послідовність операцій, програміст має визначити порядок обчислень і закласти його в структуру програми.

У **LabVIEW** є розділ **Structures**, що дозволяє регламентувати цей процес. Виклик структур здійснюється з функціонального набору **Functions** меню **Structures** (див. рис. 7.1).

Цикл із фіксованим числом ітерацій **For Loop** (див. рис. 7.2) виконує код розміщений в його межах деяке число ітерацій – **count**. Це число дорівнює величині, уведеній до терміналу числа ітерацій – **count terminal**.

Дані надходять до циклу і виходять із нього через маленьке вікно на межі циклу, що називається точкою входу/виходу до структури – тунель – **tunnel**. Оскільки **LabVIEW** працює відповідно до принципу потоку даних, то перш ніж цикл почне виконуватися, вхідні точки повинні передати до нього свої дані. Вихідні точки циклу виводять дані лише після завершення всіх ітерацій.

Завдання до лабораторної роботи

8.1. Для вирішення завдання слід викликати цикл за завданням **For Loop** (рис. 8.1) і помістити в нього формульний вузол **Programming→ Structures→Formula Node**, в який вписати формулу $u = U_m \cdot \sin(\omega t)$, де аргумент ωt в радіанах записується як $\omega t = k \cdot \pi/180$.

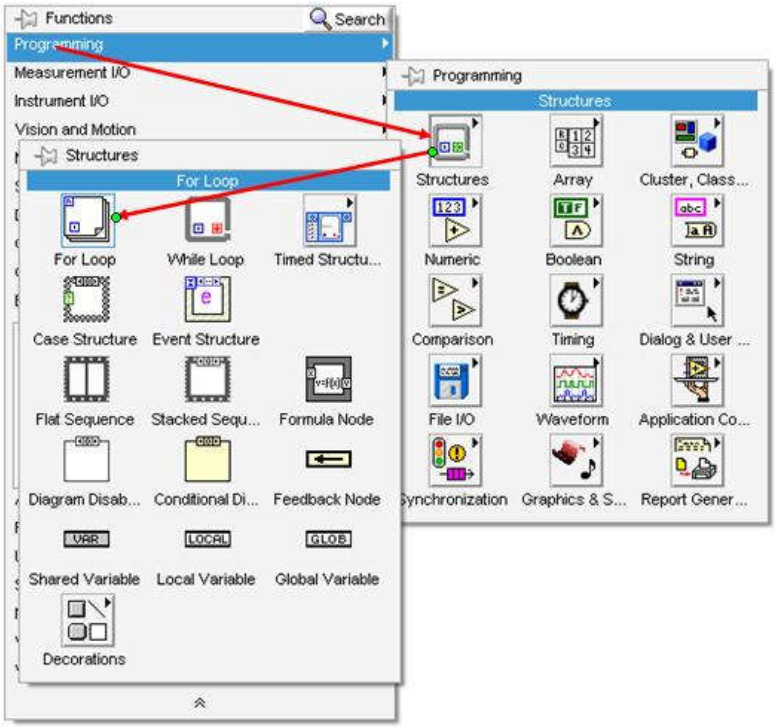


Рис. 8.1 – Піктограма циклу за завданням For Loop

Тут параметр лічильника операцій позначений буквою **k** для того, щоб відрізнити його від позначення струму. Для заміни досить встановити інструмент «введення тексту» (курсор А) на місце синьою букви «**i**» та натиснути **k**. Параметр **k** відповідає числу градусів $k = \varphi$ град. У нашому випадку доцільно провести розрахунок протягом одного – двох періодів. Тому потрібно поставити число відліків **N** =360 або 720 (рис. 8.2).

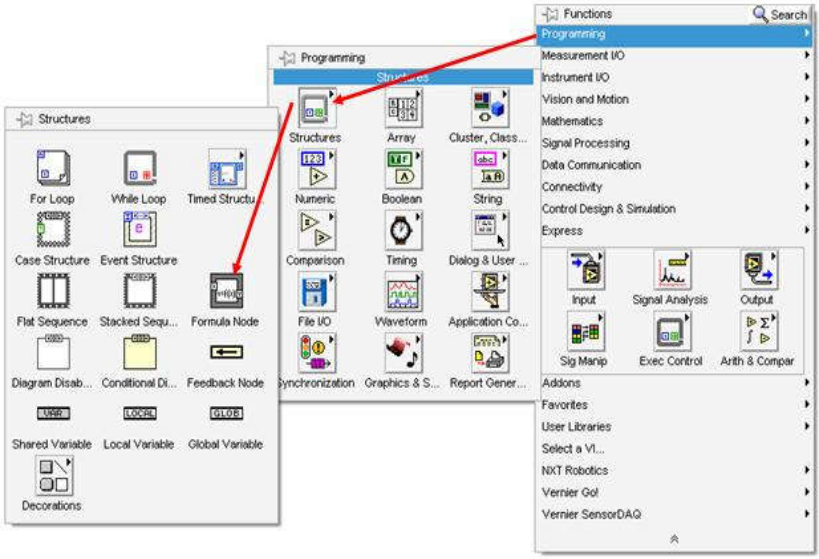


Рис. 8.2 – Піктограма формульного вузла Formula node

8.2. Натиснувши правою клавішею миші на рамку формульного вузла, вводимо вхідні і вихідні величини **Add Input**, **Add Output**.

8.3. Отримані значення напруги, струму і миттєвої потужності потрібно вивести на осцилографи **Waveform Graph** (див. рис. 8.3).

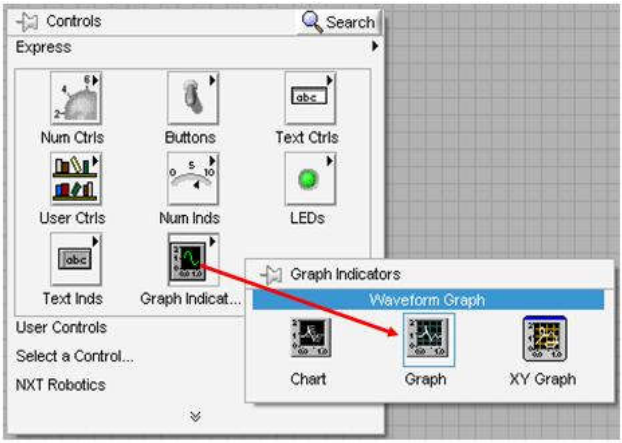


Рис. 8.3 – Піктограми різних графічних індикаторів

8.4. Діючі (середньоквадратичні) значення синусоїдальної напруги та струму вимірюються віртуальними приладами **RMS** (**Root Mean Square**), що викликаються на панелі блок-схем по шляху **Functions→ Mathematics→Probability and Statistics→RMS**, до виходів яких підключаються цифрові індикатори (рис. 8.4).

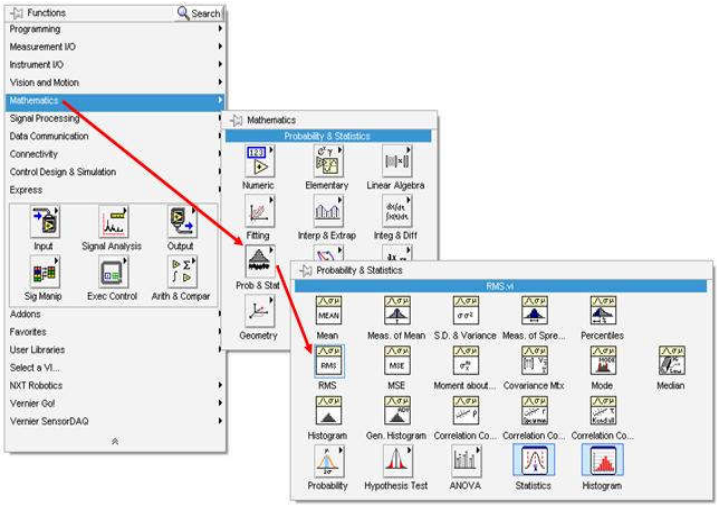


Рис. 8.4 – Піктограма для визначення синусоїдальної напруги і струму, що вимірюється віртуальними приладами RMS

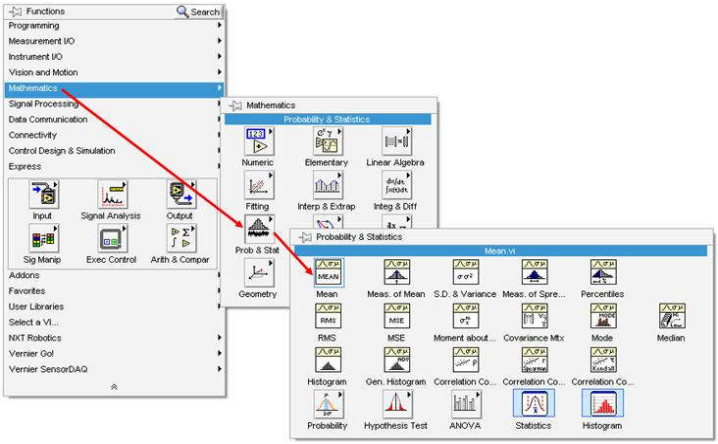


Рис. 8.5 – Піктограма віртуального приладу Mean для вимірювання активної потужності

8.5. Активна потужність, що представляє собою середнє за період значення миттєвої потужності, вимірюється віртуальним приладом **Mean**, що викликається аналогічно: **Functions→Analyze→Mathematics→Probability and Statistics→Mean**; на виході також потрібно індикатор (див. рис. 8.5).

Потрібно побудувати криві напруги, струму і миттєвої потужності при різних зсувах фаз між напругою та струмом. Результати вимірювань можна порівняти з результатами розрахунків за формулами.

Маємо отримати результат, як це показано на рис. 8.6.

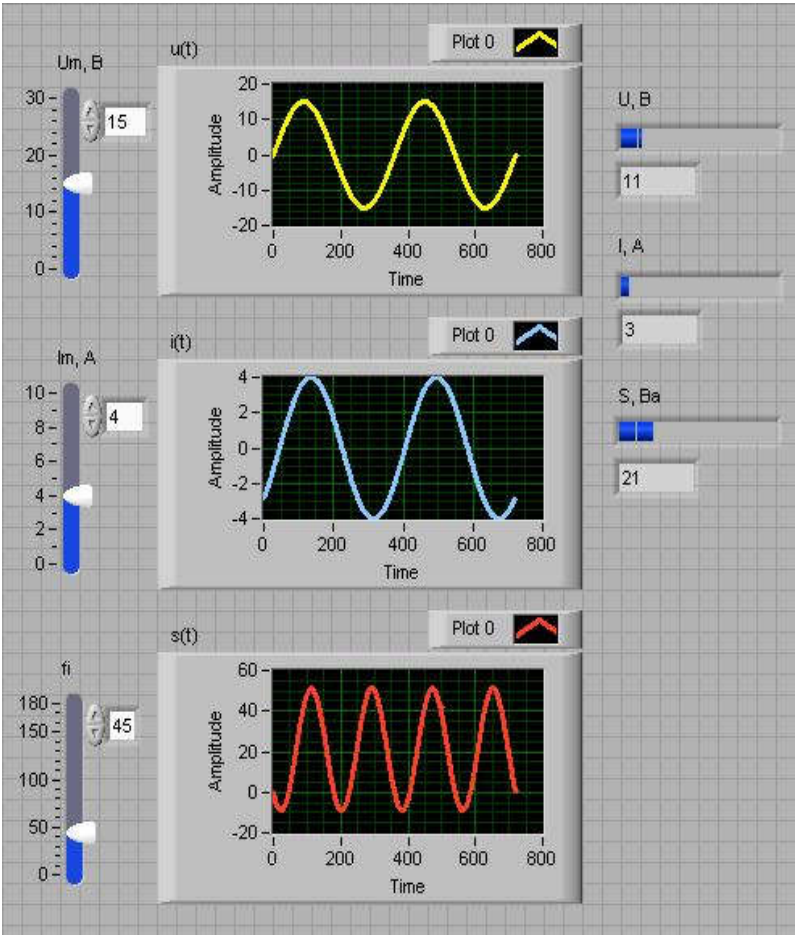


Рис. 8.6 – Лицьова панель віртуального приладу

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу (які елементи і функції використані).
4. Висновки з роботи.

Контрольні питання

1. Яким чином можна визначити послідовність виконання програми?
2. Що таке цикл?
3. Як працює цикл за умовою **While Loop**?
4. Як працює цикл з фіксованим числом ітерацій **For Loop**?
5. Що таке тунель структури?
6. У яких режимах працюють тунелі структур?
7. Як виконується автоматичне індексування при створенні масиву?

***Література*:** [2, 3, 5]

Лабораторна робота 9

Поняття масиву та кластера. Використання функцій для роботи з масивами і кластерами у LabVIEW

***Мета роботи*:** вивчити функції для роботи з масивами і кластерами у **LabVIEW** та побудувати віртуальні прилади для оброблення цих типів даних.

Порядок виконання

1. Ознайомитись з теоретичними відомостями.
2. Відкрити програмний пакет **LabVIEW**. Створити новий документ **New VI**.
3. Відповідно до варіанта створити віртуальний прилад для роботи з масивами та кластерами.
4. Оформити звіт з виконаної роботи.

Теоретичні відомості

Масивом називається впорядкована послідовність елементів одного типу. Основним параметром масиву є його розмірність. Масиви можуть мати розмірність від 1 і вище. При цьому кількість елементів у кожній розмірності може досягати $(2^{31}–1)$.

На лицьовій панелі створюються масиви у вигляді приладів (контролерів або індикаторів), а на функціональній панелі здійснюються різні операції з масивами, також створюються масиви-константи.

При створенні масиву в **LabVIEW** з інструментального набору **Controls** на лицьову панель викликається рамка масиву, у яку мишею треба внести елемент масиву. Тип цього елемента й буде визначати тип масиву (рис. 9.1).

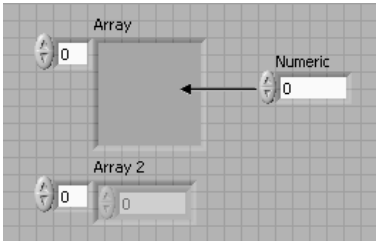


Рис. 9.1 – Створення масиву на лицьовій панелі

Як елементи масиву, можуть виступати будь-які розглянуті раніше прилади. Якщо елементами масиву є контролери, то й масив буде відповідно контролером. При заміні статусу елемента всередині масиву змінюється й статус масиву.

Цифрові масиви можна подавати на обидва входи математичної операції, але якщо вони одного порядку. При додаванні двох цифрових масивів результуючий масив складається із суми елементів з однаковими індексами. Якщо розміри масивів різні, то розмір результуючого масиву обмежиться розміром масиву з меншим числом елементів. Те саме стосується й логічних, і рядкових операцій. Вони також можуть працювати з масивами, що містять логічні й рядкові елементи.

Кластери – один із найпотужніших засобів програмування в **LabVIEW**, що дозволяє розв'язувати безліч проблем, пов'язаних з передачею даних.

Якщо масив являє собою сукупність однотипних елементів, то кластер може містити в собі елементи різних типів, у тому числі масиви й інші кластери. Механізм дії кластера можна зобразити як з'єднані вихідні провідники окремих різнотипних реальних приладів у загальний джгут (саме його імітує кластер), по якому інформація передається в будь-який інший пристрій, де цей джгут розділяється на окремі провідники, а вже з них зчитується інформація. На лицьовій панелі всі елементи, зібрані в кластер, як і елементи масиву, розташовуються в рамці. Рамку кластера можна викликати з інструментального набору **Controls**. Далі послідовність створення кластера така ж сама, як при створенні масиву.

Математичні операції, які є поліморфними, можуть працювати з кластерами так само, як із цифровими змінними, причому кластери на входах операцій повинні містити тільки цифрові елементи й бути однотипними. Їх можна додавати, віднімати, множити і ділити, узявши елементи з однаковими індексами або окремими цифровими змінними. Створювати кластери можна не тільки на передній панелі, але й на діаграмі. У функціональному меню є набір операцій з кластерами (рис. 9.2).

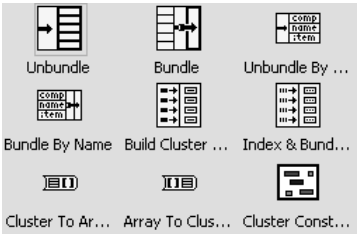


Рис. 9.2 – Операції з кластерами

Приклад виконання завдання

1. Відкрийте нову панель і створіть ВП, показаний на рис. 9.3. Спочатку створіть два масиви, перш за все вибравши шаблон масиву в підпалітрі масив і кластер палітри функції. Потім помістіть числовий елемент відображення в вікно шаблона. Для того щоб побачити кілька елементів масиву, ви повинні пересунути кут вікна відображення елементів за допомогою сіткового курсору інструменту переміщення. Ви можете зробити видимими одночасно велику кількість елементів одновимірного масиву, витягаючи кордон вікна в горизонтальному або у вертикальному напрямку.

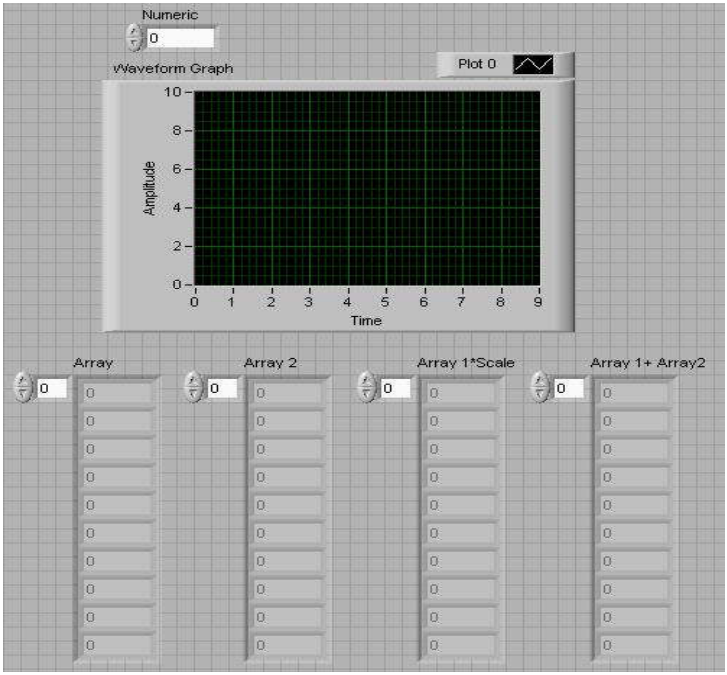


Рис. 9.3 – Приклад загального вигляду масивів

Всі чотири масиву в цій вправі містять елементи відображення. Дайте їм особливі ярлики, щоб надалі їх не переплутати.

Якщо ви все-таки забудете, який об'єкт лицьової панелі відповідає якому терміналу блок-діаграми, то просто клацніть правою кнопкою миші по будь-якому з них і виберіть опцію «Знайти термінал» **Find Terminal** або «Знайти індикатор» **Find Indicator**, і **LabVIEW** виділить відповідний об'єкт (рис. 9.4).

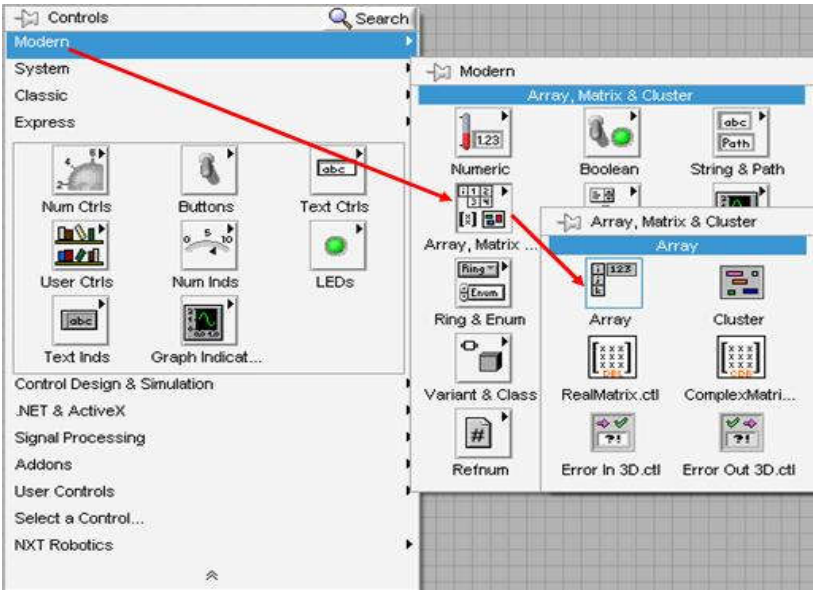


Рис. 9.4 – Піктограма функції «Масив»

2. Після того, як ви створили масиви, виберіть функцію «Графік осцилограми» підпалітрі **Graph** палітри «Елементи управління».

3. Не забудьте створити елемент управління масштабом.

4. Побудуйте блок-діаграму, як показано на рис. 9.5.

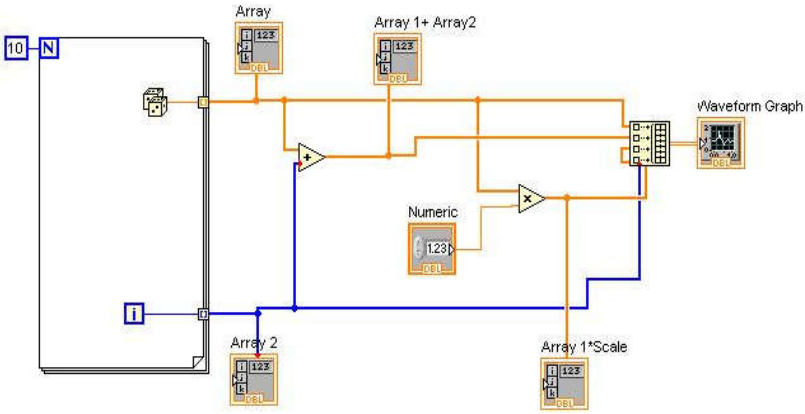


Рис. 9.5 – Готова блок-діаграма віртуального приладу для заповнення масивів

Будьте уважні: з'єднання елементів може бути досить заплутаним. У циклі з фіксованим числом ітерацій автоіндексація задіяна за замовчуванням, тому масиви будуть створюватися автоматично.

5. Функції **Add**, **Multiply** і **Random Num (0-1)** знаходяться в палітрі **Numeric**. (рис. 9.6).

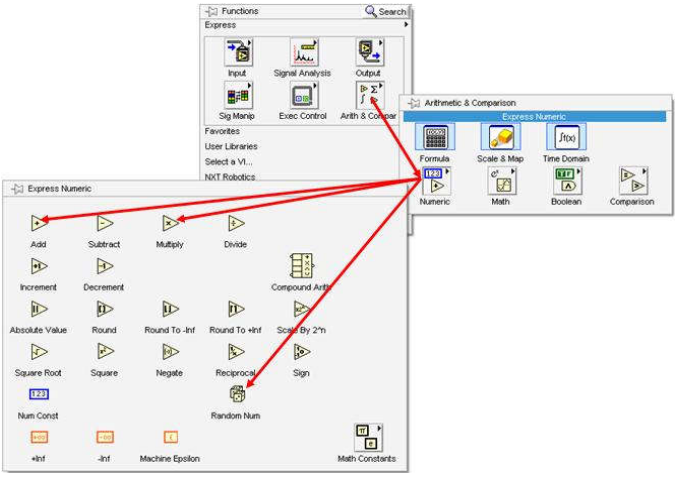


Рис. 9.6 – Піктограми функцій додавання, множення та випадкове число

6. Використовуйте функцію «Створити масив» з палітри **Array**. (рис. 9.7).

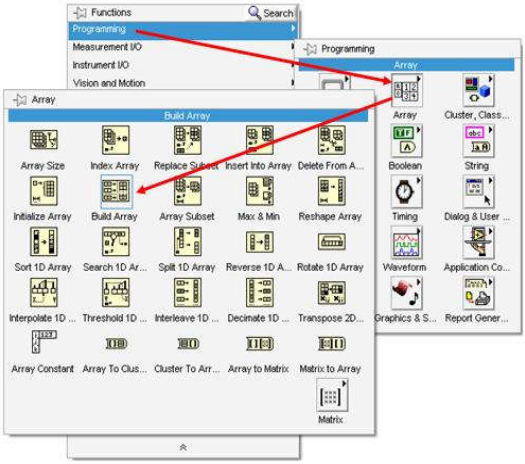


Рис. 9.7 – Піктограма функції Array

Збільште її за допомогою інструменту переміщення таким чином, щоб на виході функції «Створити масив» був двовимірний масив. Кожен вхідний масив стає рядком, так що вихідний двовимірний масив складається з чотирьох рядків і десяти стовпців.

7. Запустіть ВП. На графіку кожен елемент масиву розташовується навпроти свого індексу одночасно для всіх чотирьох масивів: «Дані масиву 1», «Дані масиву 2», «Дані масиву1×Масштаб» і «Масив1+Масив2». Ці результати демонструють кілька видів використання поліморфізму в **LabVIEW**. Наприклад, «Масив1» і «Масив2» можуть бути вхідними осцилограмами, які потрібно масштабувати.

8. Збережіть ВП. Закрийте віртуальний прилад.

Завдання до лабораторної роботи

Завдання 9.1. Створити програму, для роботи з масивами і кластерами відповідно до варіанта (табл. 9.1).

Таблиця 9.1 – Варіанти завдань

Варіант	Завдання
1	У кластері два одновимірні масиви. Скласти два масиви, знайти максимальний і мінімальний елементи результуючого масиву. Відсортувати за зростанням
2	Скласти масив з числом *X*. Знайти індекс максимального елемента і, починаючи з нього, виділити підмасив. Записати до кластера отриманий масив і його довжину
3	З двох одновимірних масивів, відсортованих за зростанням, створити двовимірний. Перетворити рядки в стовпці. Записати до кластера отриманий масив і його розмірність
4	Засвітити індикатор критичного значення якщо значення регулятора перевищує задане число *X*. Створити одновимірний масив і вставити число *X* на п'яту позицію
5	Дано два масиви. Перший відсортувати за зростанням і знайти максимальний елемент. Другий – за спаданню і знайти мінімальний елемент. Скласти два масиви в один і знайти кількість його елементів. Результати записати до кластера
6	Кластер складається з трьох масивів. Для першого визначити максимальний і мінімальний елементи та поміняти їх місцями; знайти суму елементів другого і добуток елементів третього масиву
7	Дано масив. Знайти максимальний і мінімальний елементи. Виділити два підмасиви: 1) починаючи з максимального до кінця, відсортувати за зростанням; 2) від початку до мінімального елемента, відсортувати за спаданням. Записати результуючі масиви до кластера

Продовження таблиці 9.1

Варіант	Завдання
8	У кластері двовимірний масив. Визначити суму елементів для кожного рядка масиву і засвітити індикатор для рядка з більшою сумою
9	Дано два масиви. Виділити підмасив до мінімального елемента й вставити в кінець другого. Для отриманого масиву знайти довжину, максимальний і мінімальний елементи, суму елементів і записати дані до кластера
10	Знайти різницю двох масивів. Поміняти місцями перший і останній елементи. Записати до кластера масив, максимальний, мінімальний елементи і добуток елементів

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу та його блок-діаграма.
3. Опис побудови віртуального приладу (які елементи та функції використані).
4. Висновки з роботи.

Контрольні питання

1. Що таке масив?
2. Як побудувати масив у **LabVIEW**?
3. Які є функції для роботи з масивами?
4. Що таке кластер?
5. Як побудувати кластер у **LabVIEW**?
6. Які є функції для роботи з кластерами?
7. Що таке поліморфізм?

***Література*:** [3, 4, 6]

Лабораторна робота 10

Найпростіші графічні індикатори. Використання циклів і масивів при побудові віртуальних приладів

***Мета роботи*:** навчитися використовувати цикли та масиви при роботі з функціями для отримання необхідних графічних залежностей.

Порядок виконання

1. Відкрити програмний пакет **LabVIEW**. Створити новий документ **New VI**. Вивчити елементи й функції математичного опису.
2. Відповідно до варіанта створити віртуальний прилад для побудови графічних залежностей з використанням циклів та масивів.
3. Оформити звіт з виконаної роботи.

Завдання до лабораторної роботи

Побудуємо віртуальний прилад, який буде показувати на екрані дві функції «sin» і «cos». Прилад повинен дозволяти змінювати число періодів, які відображаються на екрані, і дозволяти вимірювати значення цих функцій. Для цього нам знадобиться найпростіший графічний індикатор **Waveform Chart**, засіб для формування циклу **For Loop** і засіб для об'єднання масивів даних в кластер.

Завдання 10.1. Графічний індикатор **Waveform Chart** використовують для відображення послідовності даних, що надійшли на його вхід. По осі x послідовність нумерується цілими числами. Запустимо програму **LabVIEW**, створимо новий віртуальний прилад і вивчимо можливості цього індикатора. Помістимо індикатор на лицьову панель **Control→Graph→Waveform Chart**. Викличемо до нього контекстне меню. У цьому в розділі **Visible Item** меню ми будемо використовувати пункти:

– **Label;**
– **Plot Legend;**
– **Cursor Legend;**
– **X - Scale;**
– **Y - Scale.**

Відзначте ці пункти «галочкою». Додайте елементи на лицьову панель так, як показано на рис. 10.1.

Налаштування вузла. У контекстному меню до графічного індикатору викличте останній пункт **Properties**. Будемо коригувати вміст вкладок.

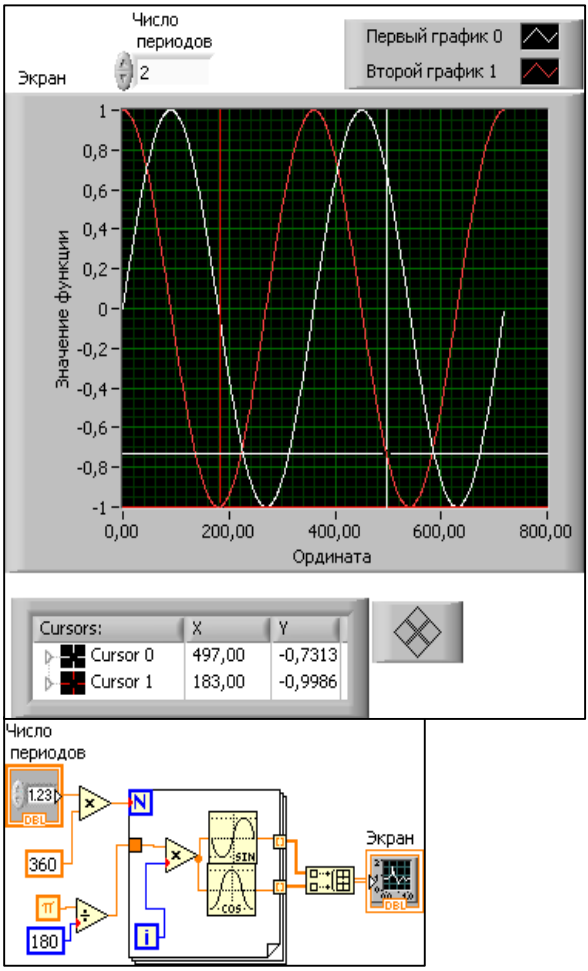


Рис. 10.1 – Лицьова панель та блок-діаграма віртуального приладу

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

2. Вкладка **Format and Precision** в рамці зліва вгорі: ордината (**X-axis**), значення функції (**Y-axis**); параметр **digit** – 6.

3. 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.

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

5. Вкладка **Cursors** – параметр **Show cursor** повинен бути відзначений:

– 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.

Такі налаштування потрібні для нашого віртуального приладу. Надалі, коли прилад буде створений і збережений, спробуйте поміняти ці налаштування і стежте, до чого це призведе.

Завдання 10.2. Прилад для формування циклу **For Loop**. Це прилад призначений для розрахунку **N** значень будь-якої функції, номер реалізації позначений через **i**, яке змінюється від «0» до «**N**-1». Завантажте в блок-діаграму цикл **For Loop**; **Function→Structure→For Loop**. На блок-діаграму буде поміщений порожній квадрат з двома невизначеними параметрами **N** та **i**. Лівіше циклу збереть блок-дігграму (див. рис. 10.1). Її верхня частина формує число значень для циклу (**N**=360*число періодів). Вихід вузла множення подайте на вхід **N** циклу. Нижня частина формує значення в 1 градус, виражене в радіанах. Вихід дільника поки не під'єднуйте.

Тепер заповнимо цикл. Розмістимо в ньому вузли розрахунку синуса і косинуса **Function→Mathematics→Elementary→ Trigonometric→...**, перед ними поставимо вузол множення і його вихід під'єднаємо до входів цих вузлів. На входи вузла множення подамо: **i** – номер реалізації та вихід дільника, який розташований за межами циклу. Таким чином, ми забезпечили розрахунок функцій «sin» та «cos» через 1 градус протягом стількох періодів, число яких задано регулятором «Число періодів». Щоб подати вихідні дані на вхід графічного індикатора, їх потрібно об'єднати в єдиний потік, що можна здійснити, сформувавши масив.

Завдання 10.3. Формування масиву вхідних даних для графічного індикатора **Waveform Chart**. Масив – **array** – це пронумерований, безперервний, необмежений набір однотипних даних. Кожен елемент масиву має набір індексів, відповідний розмірності масиву: одновимірний – 1 індекс, двовимірний – 2 індекси і т.д.

Графічно масив виглядає як прямокутна область, через яку можна проглядати елементи масиву. Поруч з лівим верхнім кутом цій

галузі відображаються 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 великих розмірностей на площині екрану монітора відобразити неможливо, тому вони виглядають як таблиці, що представляють собою зріз за певними індексами.

Масив може містити дані довільного типу. Наприклад, може бути масив тумблерів (дискретні регулятори), або масив цілих чисел, або масив кластерів. Для зміни розмірності масиву можна використовувати спливаюче меню властивостей індексу 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 безперервний набір даних, без пропусків.

Всі елементи масиву мають один і той же тип даних, причому в широкому сенсі. Це означає, що однакові як і власне типи даних, так і їх графічне зображення, кольори, розміри графічних образів кожного елемента. У палітрі функцій є всі необхідні засоби для роботи з масивами.

Побудуємо масив вхідних даних для графічного індикатора **Waveform Chart**. Розмірність масиву визначається числом відображуваних кривих, а число елементів в рядку масиву – числом точок на одному графіку. Для побудови масиву на блок діаграмі скористаємось функцією **Build Array; Function→Array→Build Array**. За цією командою на екран буде виведено зображення для одновимірного масиву. Виберемо цей елемент і розтягнемо його так, щось у нього з'явилося два входи. Верхній вхід відповідає стовпцю 0 (перша крива на індикаторі), а нижній – колонки 1 (друга крива на індикаторі). Подамо на верхній вхід вихід блоку «sin», який розташований всередині циклу, а на нижній – вихід блоку «cos», який розташовується там же. Вихід масиву подамо на вхід графічного індикатора «Екран».

Увімкніть віртуальний прилад і перевірте його роботу.

Завдання 10.4. У попередньому пункті ми виводили інформацію на екран. Тепер отримаємо крім графіка ще й таблицю значень функцій «sin» та «cos», для цього скористаємося масивами.

На рисунку представлений модернізований прилад, який не тільки виводить графік функцій, але і їх значення.

Створимо 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 виведення від 0 до 10 значень функцій в інтервалі від 0 до 90 градусів через градус, причому початкове значення вибирається за допомогою регулятора.

Для цього викличемо контекстне меню до лицьової панелі і за допомогою команди: **Control→Array, Matrix...→Array** розмістимо дві заготовки масиву на передній панелі. Для кожного з них встановимо числовий тип даних, завантаживши в обидва масиви число **Control→ Numeric→Numeric Control**. За допомогою контекстного меню визначимо для цих чисел формат **SGL; properties→Data Range→Representation→SGL**. Розмістимо ці масиви так, як показано на малюнку і зробимо видимими їх мітки (контекстне меню до масиву **→Visible Item→Lable**). Додамо два регулятора з мітками «Початковий елемент (початок)» і «Число елементів (число)». Розмістимо їх на лицьовій панелі (рис. 10.2). Встановимо видимість значень регуляторів (контекстне меню до регулятору **→Visible Item→Digital Display**). Нижче другого масиву введіть текст «Кут першого елемента в градусах» і перетягніть числове значення першого індикатора під цей напис.

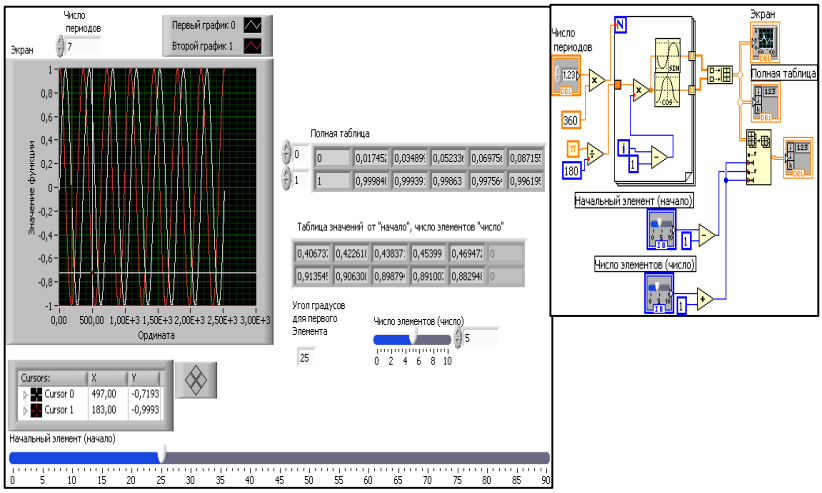


Рис. 10.2 – Лицьова панель та блок-діаграма віртуального приладу

Перейдемо до блок-діаграми (на рисунку праворуч), розмістимо масив «Повна таблиця» під графічним індикатором і під'єднаємо його вхід. Нижче розташуємо вузол для вибору даних з масиву **Control→Array, Matrix...→Array Subset**. Під'єднаємо його вхід **array** до шини даних і значок зміниться. Розмістимо регулятори «Початковий елемент (початок)» і «Число елементів (число)». Від значення першого з них віднімемо, а до значення другого додамо 1 і під'єднаємо результат до входів елемента **Array Subset**. Залишилася остання правка. Щоб почати

виведення елементів з нульового значення в циклі потрібно зменшити поточний номер на 1. Проробимо це.

Перевіримо правильність роботи блоку, наприклад, використовуючи значення для регуляторів, приведених на рисунку і порівнюючи результати. Якщо все працює правильно, то змінимо значок приладу. Виберемо зразок контактної панелі з двома входами і двома виходами. Входи – регулятори «Початковий елемент (початок)» і «Число елементів (число)», а виходи масиви – «Повна таблиця» і «Таблиця значень від (початок)», «Число елементів (число)».

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу (які елементи й функції використані).
4. Висновки з роботи.

Контрольні питання

1. Які у **LabVIEW** є елементи графічного відображення інформації?
2. Що являє собою розгорнення осцилограми?
3. Які є режими відображення розгорнення осцилограми?
4. У чому полягають функціональні відмінності розгорнень і графіків осцилограм?
5. Що таке цикл?
6. Як працює цикл з фіксованим числом ітерацій **For Loop**?
7. Що таке масив?
8. Як побудувати масив у **LabVIEW**?
9. Які є функції для роботи з масивами?

***Література*:** [3–6]

Лабораторна робота 11

Використання вузлів вибору при побудові віртуальних приладів

***Мета роботи*:** вивчити використання операцій логічного розгалуження програми, вузол **Select** і **Case**-структури.

Порядок виконання

1. Ознайомитись з теоретичними відомостями.
2. Відкрити програмний пакет **LabVIEW**. Створити новий документ – **New VI**. Вивчити елементи та функції математичного опису.
3. Відповідно до варіанта створити віртуальний прилад для використання операцій логічного розгалуження програми, вузол **Select** і **Case**-структури.
4. Оформити звіт з виконаної роботи.

Теоретичні відомості

Реалізація навіть елементарних алгоритмів, як правило, не обходиться без операцій логічного розгалуження програми в залежності від певних умов. Для цих цілей використовується вузол **Select** і **Case**-структури. Такі вузли дозволяють здійснювати вибір за умовою або за значенням параметра-селектора і переходити на виконання відповідних дій. Вузол **Select** призначений для вибору одного з двох варіантів, а вузол **Case** – одного з декількох. Тому керувати вузлом **Select** можна, наприклад, перемикачем. Для керування вузлом **Case** використовують спеціальний регулятор **Menu Ring**, хоча це і не обов'язково. Можна обійтися цілочисельними перемикачами.

Завдання до лабораторної роботи

Завдання 11.1. Побудуємо віртуальний прилад 1, який складається з двох перемикачів, кожен з яких має грубе і плавне регулювання. Обидва перемикача плавно змінюють величину від 0 до 1, а грубе перемикання відбувається по-різному. У першому перемикачі додається ціле число, а в другому – значення плавного регулятора множиться на 10 в цілому степені. Пристрій вибору подає на вихід один з варіантів залежно від положення перемикача «Вибір».

На рис. 11.1 поміщена лицьова панель цього віртуального приладу та його блок-діаграма. Для «Перемикача 1» установка значення

«грубо» здійснюється елементом **Dial; Controls→Numeric→Dial**, а точно елементом **Controls→Numeric→Horizontal Point**. Після складання сигнал надходить на вузол **Select**. Для «Перемикача 2» використовуються ті ж регулятори, але значення перемикача «грубо» використовується як показник ступеня, в яку зводиться 10, а потім сигнали перемножуються і теж надходять на вузол **Select**. Для вибору номера перемикача використовується елемент **Vertical Toggle Switch; Controls→ Boolean→ Vertical Toggle Switch**.

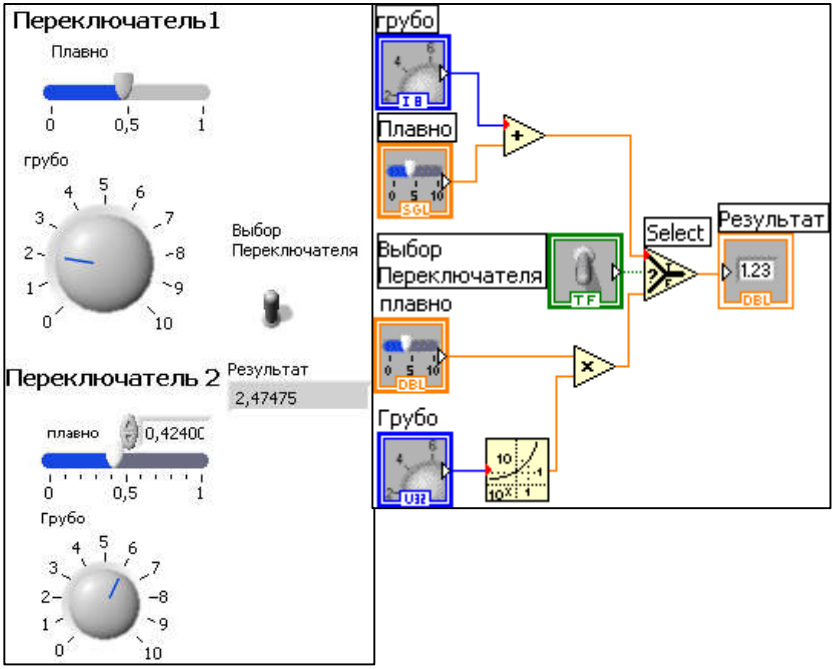


Рис. 11.1 – Лицьова панель віртуального приладу та його блок-діаграма

Створіть лицьову панель приладу і його блок-діаграму. Для того, щоб перемикачі виконували свої функції слід встановити формат даних для регуляторів. Регулятори «грубо» повинні приймати тільки цілочисельні значення, можливо негативні. Тому для них **Data Range** – **U8**. Регулятори «плавно» повинні видавати натуральні числа. Тому для них формат **SQL.**

Завдання 11.2. Побудуємо віртуальний прилад 2 – найпростіший калькулятор, який виконує 4 дії арифметики (див. рис. 11.2). На ри-

сунку зліва приведена лицьова панель, нижче – блок-діаграма, а праворуч вміст сайту **Case** при різному значенні керуючого сигналу, який формується вузлом **Menu Ring** (контекстне меню до лицьової панелі **Modern→Ring&Enum→Menu Ring**). Додайте на лицьовій панелі необхідні регулятори, індикатори та елемент **Menu Ring**). Останньому надайте мітку «Виберіть дію» і зробіть її видимою на екрані. Мітки для регуляторів і індикаторів встановіть відповідно до рисунку на лицьовій панелі. Дайте формати регуляторам (в контекстному меню **Data Range**) відповідно до форматів на блок-діаграмі (їх можна відрізнити за кольором регуляторів і проводів).

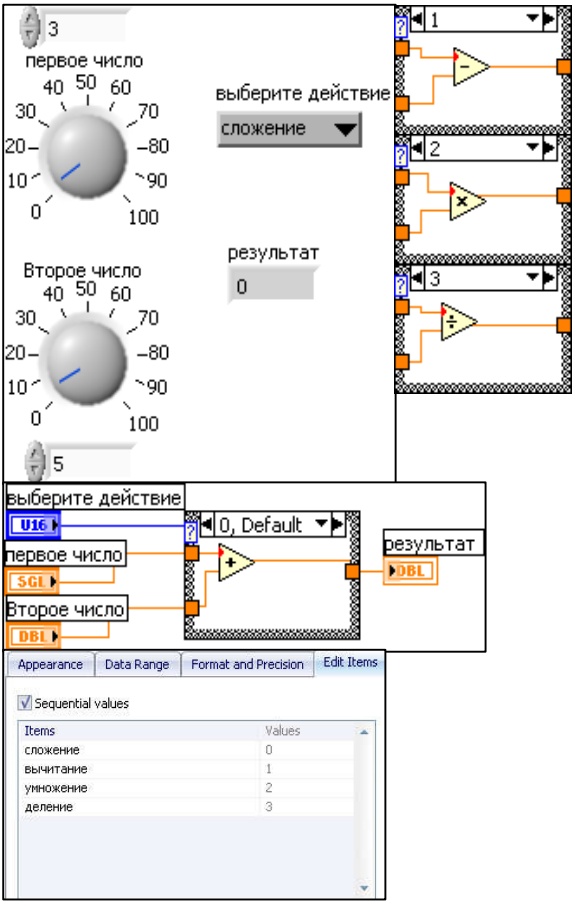


Рис. 11.2 – Лицьова панель віртуального приладу та його блок-діаграма

Встановимо властивості вузла **Menu Ring**. Для цього викличемо для нього контекстне меню, а в ньому пункт **Properties**. У діалоговому вікні виберемо вкладку **Edit Items**. Відредагуємо її відповідно до рисунку, розташованому під блок-діаграмою. Керуючий елемент готовий.

Перейдемо на блок-діаграму. Вставимо вузол вибору **Case** (контекстне меню до блок-діаграми **Function→Structure→Case Structure**). Підведемо керуючий сигнал від регулятора «виберіть дії». В меню вузла **Case** виберемо випадок 1, потім в контекстному меню вузла команду **Add Case After** два рази. Тепер у вас чотири варіанти роботи вузла – 0, 1, 2, 3. Заповніть ці варіанти відповідно до рисунку. Підключіть вузол до регуляторів і індикаторами, перевірте правильність форматів регуляторів і індикаторів. Запустіть віртуальний прилад і перевірте правильність його роботи. Відредагуйте значок, створіть конектор і призначте вхідним і вихідним вузлів контакти конектора. Конектор містить три вхідних сигнали: перше число, виберіть дію і друге число; один вихідний сигнал: результат.

Завдання 11.3. Створіть віртуальний прилад, який виконує наступні функції:

1. На вхід приладу надходить три числа: n_1, n_2 і n_3 в діапазоні значень від 0 до 2π.

2. Число n_2 використовується для обчислення однієї з функцій: $\sin(x)$, $\cos(x)$, x^2, $x^{1/2}$ за нашим вибором.

3. Результат розрахунку надходить на блок, який формує сигнал:

– 0, якщо результат за модулем менше 0.25;

– 1, якщо результат за модулем більше або дорівнює 0.25, але менше 0.5;

– 2, якщо результат за модулем більше або дорівнює 0.5, але менше 0.75;

– 3, якщо результат за модулем більше або дорівнює 1.

4. Цей результат використовується для управління калькулятором, на входи якого надходять числа n_2 і n_3. Калькулятор використовується готовий, зроблений в попередньому пункті.

Зміст звіту

1. Тема та мета лабораторної роботи.

2. Вигляд лицьових панелей віртуальних приладів і їх блок-діаграм.

3. Опис побудови віртуального приладу (які елементи та функції використані).

4. Висновки з роботи.

Контрольні питання

1. Що таке операції логічного розгалуження?
2. Для яких цілей використовується вузол **Select**?
3. Для яких цілей використовується **Case**-структура?
4. Як відбувається керування вузлом **Select** та **Case**-структурою?

***Література*:** [3–6]

Лабораторна робота 12

Використання графічного індикатора Waveform Graph для отримання графічних залежностей

***Мета роботи*:** вивчити використання графічного індикатора **Waveform Graph** для отримання графічних залежностей з набору даних різних форматів.

Порядок виконання

1. Відкрити програмний пакет **LabVIEW**. Створити новий документ – **New VI**. Вивчити елементи й функції математичного опису.
2. Відповідно до варіанта створити віртуальний прилад для отримання графічних залежностей з набору даних різних форматів.
3. Оформити звіт з виконаної роботи.

Завдання до лабораторної роботи

Завдання 12.1. Скористаємося для відображення іншим графічним індикатором **Waveform Graph**, для якого потрібно задати початкове значення ординати (поздовжньої осі або осі x), приріст по цій осі Δx і масив даних для відображення по осі ординат (поперечної осі або осі y). Таким чином, на графічний індикатор доведеться подавати набір даних різного формату: поодинокі дані і масиви даних. Для цього передбачено спеціальний набір даних – кластер.

Кластер – це кінцевий набір даних різних типів. Графічно кластер виглядає як прямокутна область, всередині якої знаходяться різнотипні дані. У кластері дані розташовуються не довільно, а в певному порядку.

Порядок проходження елементів в кластері важливий для використання його даних. У кластері, використовуваному для відображення даних на графічному індикаторі **Waveform Graph** послідовність даних така (зверху вниз): спочатку вводиться x_0, потім Δx, а потім масив даних для відображення. Для роботи з кластерами в **LabVIEW** є відповідна палітра функцій **Function→Cluster&Variant**. Функції дозволяють розібрати кластер на окремі елементи (**Unbundle**, **Unbundle by name**), або зібрати різні дані в кластер (**Bundle**, **Bundle by name**). При цьому в разі складання одного кластера можна використовувати «успадкування» типів і значень даних з іншого кластера. Це дозволяє, по-перше, уникнути помилок з розбіжністю типів, по-друге, змінити значення тільки необхідних складових кластера, успадкувавши інші.

Сформуємо кластер для послідовного виведення отриманих значень «sin» і «cos» (отриманих в лабораторній роботі 10) на екран графічного індикатор **Waveform Graph**. Позначимо n_1 – число відображуваних періодів; n_3 – кількість точок в періоді, що відображаються на графіку; n_2 – задане число відображуваних негативних періодів; n– – кількість негативних періодів; N – число повторень циклу; x_0 – початкове значення ординати; dx – інтервал по ординаті між сусідніми відліками. Якщо $n_2<n_1$, то n–=n_2, якщо $n_2>n_1$, то n–=0. Тоді необхідні для роботи графічного індикатора величини можна розрахувати так:

$$dx = \frac{2\pi}{n3};\ N = n_1 \cdot n_3;\ n- = \begin{cases} n_2, & \text{при} \quad n_2 \le n_1 \\ 0, & \text{при} \quad n_2 > n_1 \end{cases};\ x_0 = -n_2 \cdot n - dx\ .$$

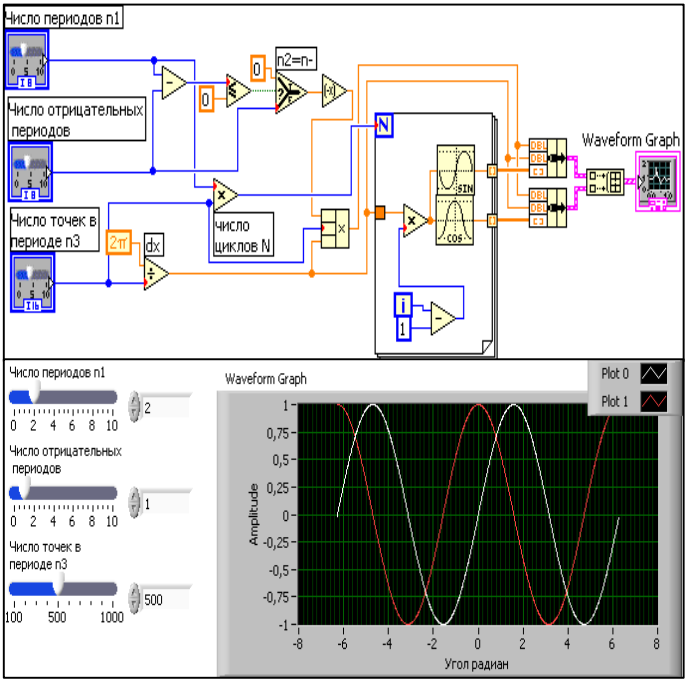


Рис. 12.1 – Блок-діаграма та лицьова панель приладу 1

Частина блок-діаграми, розташована лівіше циклу формує змінні відповідно до наведених виразів і не вимагає пояснень. Цикл для розрахунку функцій той же, що і в блок-діаграмі приладу з лабораторної роботи 10. Його можна просто скопіювати звідти.

Правіше циклу розташовані два кластери (**Function→Cluster &Variant→Bundle**). Виділіть ці два кластери і розтягніть так, щоб у кожного з них було три входи. Підключіть їх входи так, як показано на блок-діаграмі. Виходи кластерів потрібно під'єднати на входи масиву (**Function→Array→Build Array**) так само, як ви робили використовуючи інший графічний редактор. Розтягніть масив так, щоб у нього з'явилося два входи і підключіть їх до виходів кластерів. Тепер помістимо на передню панель графічний індикатор **Waveform Graph** (**Controls→ Graph→Waveform Graph**). Розмістимо і позначимо елементи на лицьовій панелі, а на блок-діаграмі під'єднаємо вихід масиву до входу графічного індикатора. Прилад готовий. Перевірте його роботу. Встановіть для регуляторів значення за замовчуванням, для цього, після того, як буде встановлено потрібне значення на регуляторах, викличте контекстне меню послідовно до кожного регулятору і виберіть команду: **Data Operation→Make Current Value Default**.

Користуючись отриманими результатами, організуйте вивід на екран значення «sin» і «cos» для заданого кута в градусах з розрахованих значень (додаткове завдання).

Завдання 12.2. Скористаємося для відображення ще одним графічним індикатором **XY Graph**, для якого потрібно задати початкове значення ординати (поздовжньої осі або осі x), приріст по цій осі Δx і масив даних для відображення по осі ординат (поперечної осі або осі y).

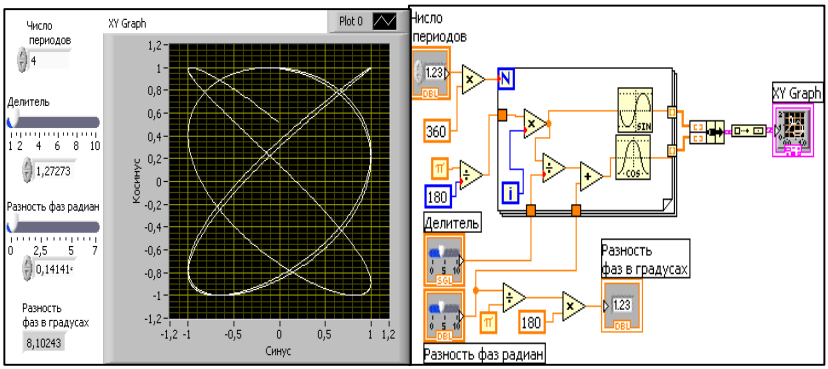


Рис. 12.2 – Блок-діаграма та лицьова панель приладу 2

Почнемо створювати новий віртуальний прилад. Запустимо програму і виконаємо команду **File→New VI**. На передній панелі розмістимо регулятори: **Numeric Control** для числа періодів і два регу-

лятора **Horizontal Point Slide** для подільника частоти та установки різниці фаз сигналів. Додамо індикатор «Різниця фаз в градусах» і перейдемо на блок-діаграму. Користуючись вузлами множення, ділення і числовими константами з розділу **Function→Numeric**, побудуємо схему для задання параметрів циклу (див. блок-діаграми зліва і нижче циклу) і організуємо його. Завантажимо заготовку циклу **Function→ Structures→ For Loop**, і заповнимо вміст циклу (див. попередні лабораторні роботи).

Тепер можна створювати систему відображення на графічному індикаторі **XY Graph**. Ця система містить кластер, який об'єднує дані для осі X і Y (**Function→Cluster&Variant→Bundle**). Ці дані надходять з виходів вузлів «sin» і «cos», що знаходяться всередині циклу. Вихідні дані кластера надходять на вхід масиву (**Function→Array→Build Array**).

Перейдіть на лицьову панель. У палітрі **Control→Graph** виберіть **XY Graph** і розмістіть на лицьовій панелі. Зручно розмістіть регулятори та відредагуйте їх мітки і видимість міток відповідно до рисунку.

Поверніться на блок-діаграму, приведіть в порядок розташування елементів і під'єднайте вихід масиву до графічного індикатора.

Тепер приведемо до ладу параметри елементів. На лицьовій панелі відредагуйте назву горизонтальної та вертикальної осі. У властивостях для графічного індикатора виберіть вкладку **Scale**, приберіть галочку для параметра **Autoscale**, і введіть мінімум –1,2, максимум 1,2.

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Лицьові панелі віртуальних приладів і їх блок-діаграми.
3. Опис побудови віртуального приладу (які елементи й функції використані).
4. Висновки з роботи.

Контрольні питання

1. Які у **LabVIEW** є елементи графічного відображення інформації?
2. Що являє собою розгорнення осцилограми?
3. Які є режими відображення розгорнення осцилограми?
4. Що таке цикл?
5. Як працює цикл із фіксованим числом ітерацій **For Loop**?
6. Що таке операції логічного розгалуження?
7. Для яких цілей використовується вузол **Select**?
8. Для яких цілей використовується **Case**-структура?
9. Як відбувається керування вузлом **Select** та **Case**-структурою?

***Література*:** [2, 4, 5]

Лабораторна робота 13

Використання генераторів сигналів при проектуванні віртуальних приладів

***Мета роботи*:** вивчити використання генераторів сигналів при проектуванні віртуальних приладів і типи фільтрів, які використовуються.

Порядок виконання

1. Ознайомитись з теоретичними відомостями.
2. Відкрити програмний пакет **LabVIEW**. Створити новий документ **New VI**. Вивчити елементи та функції математичного опису.
3. Відповідно до варіанта створити віртуальний прилад для генерації сигналів необхідної форми при проектуванні віртуальних приладів та типи фільтрів, які використовуються.
4. Оформити звіт з виконаної роботи.

Теоретичні відомості

У середовищі **LabVIEW** розрізняють цифрові й «аналогові» сигнали, цифрові можуть приймати тільки два значення 0 або 1. Такі сигнали характеризують стан входів або вихідний цифровий лінії, або декількох ліній (регістру). До цифрових сигналів відносяться також послідовності імпульсів. Цифровим сигналам відповідають логічні **Boolean** джерела/приймачі даних. «Аналогові» сигнали приймають безліч значень, що допускаються їх числовим представленням **Numeric**, і можуть бути функціями часу, частоти, просторових координат і т.д. Нижче ми будемо мати справу тільки з «аналоговими» сигналами. Лапки в даному випадку означають, що реальний аналоговий сигнал представлений в доступній комп'ютеру цифровій формі, тобто – кінцевим числом відліків, проквантованих за рівнем. Без прив'язки до часу (або інший незалежній перемінній) такий сигнал є просто числовим масивом. У літературі з **LabVIEW** для таких сигналів використовується також термін **Pattern**, що перекладається як послідовність (відліків сигналу). Оцифрований аналоговий сигнал може бути отриманий з аналого-цифрового перетворювача (АЦП) під час вимірювань і також є масивом. Кілька сигналів, що надійшли через АЦП з однотипних датчиків (каналів), утворюють двовимірний масив. Отже, можна говорити про двовимірний сигнал.

У LabVIEW існує спеціальний формат (кластер) даних – **Waveform** (переклад – осцилограма), в якому зберігається інформація про абсо-

лютну тимчасову прив'язку сигналу: час першого відліку і крок між наступними. ВП з бібліотеки **DAQmx** допускають читання і запис як масивів, так і осцилограм. В останньому випадку піклуватися про зазначення частоти подачі даних на цифро-аналоговий перетворювач (ЦАП) немає необхідності: якщо це допускається його технічними характеристиками, на виході вийде аналоговий сигнал з правильним тимчасовим масштабом.

Цифрова фільтрація **Signal Processing→Filters**.

Фільтрація – це процес, за допомогою якого змінюється частотний спектр сигналу. Це одна з найбільш широко використовуваних процедур обробки сигналів в науковому експерименті, радіо- і гідролокації, техніці зв'язку, автоматичному регулюванні, медичній діагностичній апаратурі, аудіо- та відеоапаратурі і т.д. У зв'язку з усе більш широким проникненням в ці області цифрового представлення сигналів аналогова фільтрація витісняється цифровою. Теорії, програмної та апаратної реалізації цифрових фільтрів (ЦФ) присвячена велика кількість літератури, тому викласти навіть основи цього спеціального напряму в рамках методичних вказівок неможливо. Тому припустимо, що вивчаючи цей курс, ви вже освоїли курси вищої математики і теоретичних основ електротехніки та електроніки (теорії кіл та сигналів, мікропроцесорна техніка) та нагадаємо основні поняття і визначення, які необхідні для розуміння прикладів і виконання вправ.

1. Фільтри змінюють або видаляють непотрібні частоти. Залежно від частотного діапазону, який фільтри або пропускають, або послаблюють, вони можуть бути класифіковані на наступні типи:

– низькочастотний фільтр – пропускає низькі частоти, але послаблює високі;

– високочастотний фільтр – пропускає високі частоти, але послаблює низькі;

– смуговий фільтр – пропускає певну смугу частот;

– фільтр, що загороджує – послаблює певну смугу частот.

На рис. 13.1 схематично показані частотні характеристики перерахованих фільтрів. Позначення: ПП – смуга пропускання, ПР – смуга режекції (не пропускання), ПО – перехідна область, fc – частота зрізу.

2. Імпульсна характеристика ЦФ h[k] – це його відгук на одиничний імпульс, тобто сигнал, заданий послідовністю: x[0]=1 і x[i]=0 при $i \neq 0$.

3. За видом імпульсної характеристики ЦФ поділяються на дві групи: КІХ-фільтри (**FIR-filters**) – фільтри з імпульсною характеристикою, яка має кінцеву тривалість і НІХ-фільтри (**IIR-filters**), у яких імпульсна характеристика триває нескінченно довго.

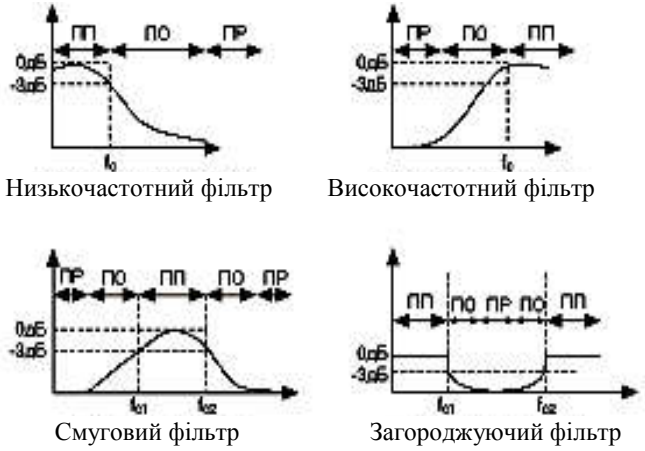


Рис. 13.1 – Схематичний вигляд частотних характеристик різних фільтрів

Завдання до лабораторної роботи

Для формування різноманітних сигналів будемо користуватися розділом контекстного меню до блок-діаграми **Programming→ Waveform**. У лабораторній роботі потрібно скористатись базовим функціональним генератором **Basic Function Generator** і вузлом для перетворення формату виходу генератора **Get Waveform Components**.

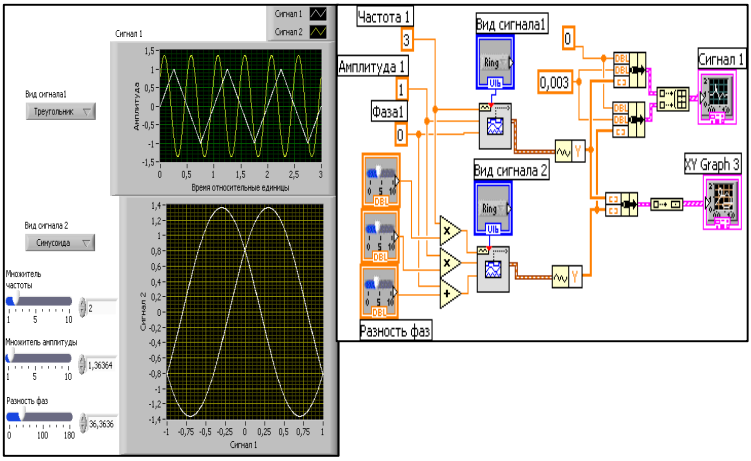


Рис. 13.2 – Блок-діаграма та лицьова панель приладу 1

Створимо віртуальний прилад, який містить два формувача сигналу. Сигнал – синусоїда, або трикутний сигнал, або прямокутний сигнал. Вихід одного з генераторів використовується для розгортки по осі *X*, а іншого подається на вісь *Y* індикатора **XY Graph.** Вихідні сигнали генераторів, крім того, подамо на графічний індикатор **Waveform Graph**, щоб спостерігати їх форму.

Запустимо програму і створимо новий віртуальний прилад. Почнемо роботу з лицьової панелі. Розмістимо два вузла вибору режиму генератора (**Controls→Ring&Enum→Menu Ring**) і змінимо їх мітки відповідно до рисунку. Розмістимо також три регулятора **Horizontal Pointer Slide** і у них змінимо мітки відповідно до рисунку. Виведемо числові значення регуляторів (контекстне меню до регулятору **→Visible Item→Digital Display**). Розмістимо два графічних індикатора **Waveform Graph** (зверху) і **XY Graph** і перейдемо на блок-діаграму.

На блок-діаграмі зручно розмістимо регулятори, додамо три константи 3, 1 і 0, розмістимо два вузла множення та один акумулятор. Проведемо з'єднання відповідно до рисунку.

Розмістимо на схемі два функціональних генератори (**Programming→Waveform→Analog Wfm→WaveformGeneration→Basic Function Generator**). Під'єднаємо виходи вузлів **Menu Ring** до входів генератора **signal type**. Зліва до генераторів підходять провідники до контактів (зверху вниз): **frequency**; **amplitude**; **phase**.

Займемося системою відображення. Правіше генераторів сигналів розмістимо два блоки, що вибирають потрібну інформацію (**Programming→Waveform→Get Waveform Components**). Розмістимо три формувачі кластера, два для індикатора **Waveform Graph** і один для **XY Graph**. Два верхніх розтягнемо на три позиції, а нижній на 2. До нижнього під'єднаємо виходи обох вузлів **Get Waveform Components**. Для верхніх, створимо дві константи 0 (початок розгортки) і 0,03 (це крок по осі X). Під'єднаємо нижні входи цих формувачів до виходів вузлів **Get Waveform Components**.

Тепер сформуємо масиви даних для графічних індикаторів. Для цього розмістимо два вузла **Build Array** правіше формувачів кластерів. Верхній розтягнемо на два входи. Під'єднаємо виходи формувачів кластерів до входів вузлів **Build Array** так, як показано на схемі. Виходи вузлів подамо на графічні індикатори – схема готова. Займемося її налаштуванням.

На лицьовій панелі послідовно викличемо властивості обох вузлів **Menu Ring** і введемо **Items** відповідно до рисунку зліва. Встановимо межі зміни у регуляторів. Виставимо значення на регуляторах, наприклад, такі, як на рисунку, і зробимо їх значеннями за замовчу-

ванням. Для цього викличте контекстне меню послідовно до кожного регулятору і виберіть команду: **DataOperation→Make Current Value Default**.

У графічних індикаторів перейменуємо осі. Викличемо властивості для індикатора **Waveform Graph** і на вкладці **Plots** встановимо кольори променів: для одного білий, а для іншого жовтий. На вкладці **Scale** для обох променів відзначимо **Autoscale**. Елемент **Plot Legend** (справа вгорі над індикатором) розтягнемо на 2 позиції і перейменуємо їх відповідно до рисунку передній панелі. У другого індикатора **XY Graph** у властивостях досить на вкладці **Scale** для обох сигналів встановити режим **Autoscale**.

Тепер прилад готовий до роботи. Включимо його і подивимося на роботу віртуального приладу в різних режимах.

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.
3. Опис побудови віртуального приладу (які елементи й функції використані).
4. Висновки з роботи.

Контрольні питання

1. Які бувають типи сигналів?
2. Що таке фільтрація сигналу?
3. Які існують типи фільтрів?
4. Які у **LabVIEW** є елементи графічного відображення інформації?
5. Що являє собою розгорнення осцилограми?
6. Які є режими відображення розгорнення осцилограми?

***Література*:** [2, 4, 5]

Лабораторна робота 14

Моделювання фізичних процесів у середовищі LabVIEW

Мета роботи: навчитись використовувати програмне середовище **LabVIEW** для математичного моделювання різних фізичних процесів та явищ, створення підпрограм обробки даних.

Порядок виконання

1. Ознайомитись з теоретичними відомостями.
2. Відкрити програмний пакет **LabVIEW**. Створити новий документ **New VI**.
3. Відповідно до варіанта створити віртуальний прилад для математичного моделювання різних фізичних процесів та явищ, створення підпрограм обробки даних.
4. Оформити звіт з виконаної роботи.

Теоретичні відомості

Розглянемо можливості інженерного середовища графічного програмування для швидкого створення професійного інтерфейсу і обробки даних для моделювання термодинамічних процесів ідеального газу.

Процесом називається будь-яка зміна параметрів стану середовища. Зазвичай змінюються всі три параметри, пов'язані між собою рівнянням стану. Для ідеальним газів, до яких відноситься повітря, рівняння стану має вигляд $P \cdot V = R \cdot T$.

Існує ряд процесів, протягом яких зберігається постійне відношення виконаної роботи і кількості тепла, що бере участь у теплообміні із зовнішнім середовищем. Такі процеси називаються політропними. Для них виконується додаткове співвідношення $P \cdot V \cdot n = \text{const}$, де n – показник політропи.

Якщо в політропному процесі повітря, яке є ідеальним газом, стискається дуже швидко, то при зменшенні об’єму в 15 разів, температура його підвищується до 650 °С.

При цьому степені стиснення, яке виконується досить повільно, температура залишається без змін. Це пов’язане з тим, що в повільному процесі теплова енергія, яка утворюється при стисненні газу, встигає розсіятись у навколишньому середовищі. Таким чином, характер зміни параметрів стану фактично залежить від швидкості процесу. В першому випадку показник політропи дорівнює коефіцієнтові

адіабати $n = 1,4$ (адіабатний процес), у другому випадку $n = 1$ (ізотермічний процес).

Завдання до лабораторної роботи

Процес відбувається в циліндрі об'ємом $V0 = 1$ л з початковим тиском $P0 = 100$ кПа і температурою $T0 = 300$ К при степені стиснення $\lambda = V0/VK = 6$.

Результати обчислень необхідно відобразити у вигляді індикаторів традиційних приладів, використаних для вимірювання V, P, T, графіків їх зміни за часом та PV діаграм процесу, що досліджується.

Якщо у стиснене повітря впорснути дизельне паливо – воно самозапалиться. Таким способом може бути реалізований один із процесів термодинамічного циклу Дизеля автомобільного двигуна.

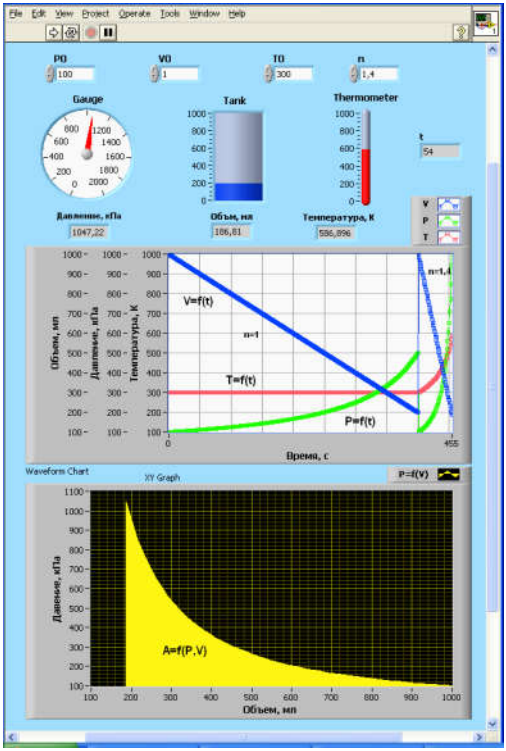


Рис. 14.1 – Лицьова панель моделювання процесів стиснення

Лицьова панель користувача, з якою здійснюється управління процесом моделювання, показана на рис. 14.1. У її верхній частині знаходиться чотири цифрових елементи керування для введення вихідних даних задачі: *V*0, *P*0, *T*0, *n*.

Для відображення поточних значень *V*, *P*, *T* на лицьовій панелі поміщені чотири віртуальних прилади – мірна ємність, стрілочний манометр, термометр і цифровий секундомір з верхніми межами показань їх шкал, відповідними діапазонами вимірювань об'єму – 1000 мл, тиску – 2000 кПа, температури 1000 К і 1000 секунд віртуального часу. Показання лінійних шкал цих приладів дублювані цифровими індикаторами, що дозволяють виконувати більш точний відлік контрольованих параметрів.

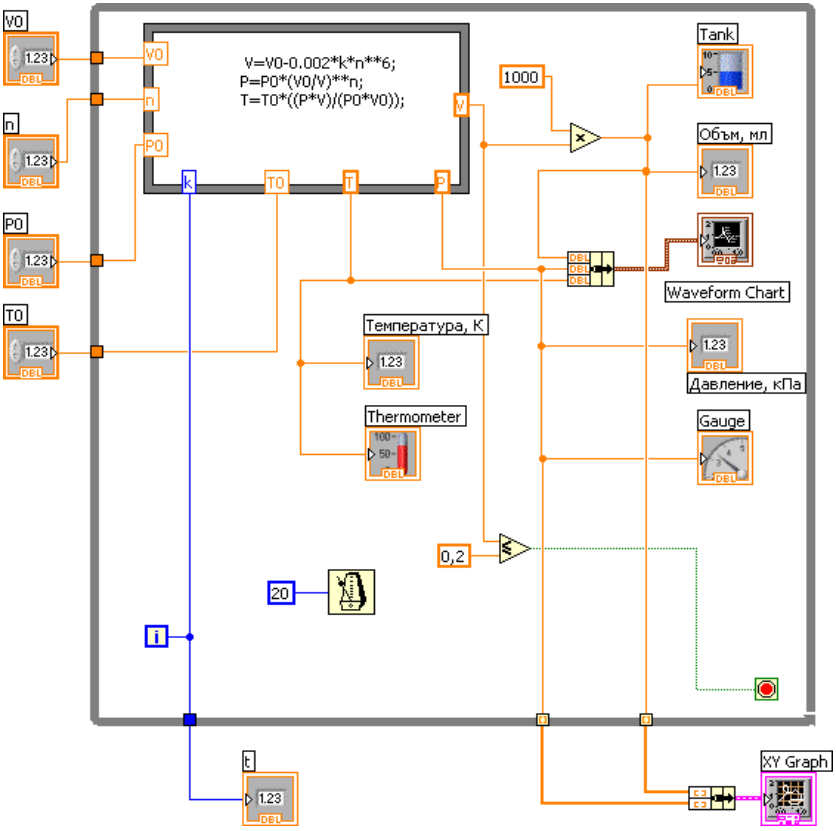


Рис. 14.2 – Блок-діаграма програми моделювання

Для спостереження за ходом процесу, що моделюється, на лицьовій панелі знаходяться віртуальний трипроменевий запам'ятовуючий осцилограф і *XY*-самописець для побудови *P–V*-діаграми процесу. Графічний код програми моделювання представлений на рис. 14.2.

У центрі блок-діаграми, яка для зручності розробки програми присутня на екрані комп'ютера одночасно з лицьовою панеллю, знаходиться графічна піктограма циклу за умовою у вигляді зовнішнього прямокутника, який включає всі оператори, що виконуються всередині циклу (політропне стиснення повітря). Наступною центральної структурою програми є вузол формул, в який у звичайному записі внесені основні співвідношення математичної моделі, що визначають зміни параметрів стану газу залежно від швидкості його стиснення і віртуального часу *k*. У нашому випадку визначальним процесом «Політропного стиснення повітря» є рух поршня і відповідна зміна об'єму стисненого повітря. Зв'яжемо швидкість цього процесу з показником політропи наступним чином:

$$V = V0 - 0,002 \cdot k \cdot n^6, \tag{14.1}$$

де $k = i$.

Це означає, що за кожен цикл моделювання об'єм стиснутого газу лінійно зменшується на ($2n^6$) мл. Відповідну зміну тиску в циліндрі описуємо формулою:

$$P = P0 \cdot \left(V0/V\right) \cdot n, \tag{14.2}$$

а температури:

$$T = T0 \cdot \left(\left(P \cdot V\right)/\left(P0 \cdot V0\right)\right). \tag{14.3}$$

Для забезпечення роботи операторів формульного вузла необхідно ввести через термінали входу значення всіх змінних, присутніх в правій частині записаних рівнянь. Обчислені значення параметрів (ліві частини формул) виводяться через термінали виходу, виділені більш жирним контуром.

Програма автоматично припиняє роботу, коли за умовою завдання ступінь стиснення повітря в циліндрі $\lambda = V0/VK$ стає рівною або більшою 6. Для цього в програмі спеціально створюється графічний ланцюжок, що формує умови завершення циклу. Він складається з логічного перемикача «істина» або «брехня» з подачею на два його входи поточного значення об'єму і кінцевого об'єму при стис-

ненні $VK \le 0,2$ л, а також вимикача програми, що спрацьовує при значенні «істина».

Для спостереження за процесом стиснення та розтягування його за часом в центрі циклу поміщений камертон з часом затримки такту 20 мс, так як за замовчуванням швидкість виконання циклу складає всього 1 мс.

Поточні значення вихідних величин V, P, T з терміналів формульного вузла подаються на входи приладів, що знаходяться на лицьовій панелі і піктограми яких присутні в циклі. Показники V, P дублюються і зберігаються на кордонах циклу у вигляді масивів. По завершенні програми вони виводяться на самописець для побудови P–V-діаграми процесу. Результати моделювання за різної швидкості стиснення повітря (різних значеннях показника політропи) наведено в таблиці 14.1.

Таблиця 14.1 – Результати моделювання політропного процесу

№ з/п	Параметр				
	n	T, мс	V, л	P, кПа	T, К
1	1				
2	1,2				
3	1,4				

Отримані значення параметрів стану використані для автоматичної побудови P–V-діаграми процесу. При цьому площа області, що лежить під кривою $P(V)$, чисельно дорівнює роботі, затраченої для стиснення повітря:

$$A = \sum Pdv .$$

Можна бачити, що в ізотермічному процесі ($n = 1$) ступінь підвищення тиску дорівнює ступеню стиснення повітря. При швидкому адіабатному стисненні ($n = 1,4$) тиск повітря в кілька разів перевищує ізотермічний та збільшується більш, ніж в 10 разів. Температура при цьому сягає 560 К. Це пов'язано з тим, що в повільному процесі теплова енергія, яка утворюється при стисненні газу, розсіюється у навколишньому середовищі. Таким чином, характер зміни параметрів стану фактично залежить від швидкості процесу.

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Вигляд лицьової панелі віртуального приладу і його блок-діаграма.

3. Опис побудови віртуального приладу.
4. Висновки щодо роботи.

Контрольні питання

1. Які у **LabVIEW**є елементи графічного відображення інформації?
2. Що являє собою розгорнення осцилограми?
3. Що таке цикл?
4. Як працює цикл із фіксованим числом ітерацій **For Loop**?
5. Як працює структурний вузол **Formula Node**.
6. Охарактеризуйте адіабатичний процес стиснення ідеального газу?
7. Охарактеризуйте ізотермічний процес стиснення ідеального газу?

***Література*:** [2, 3, 5, 6]

Перелік джерел посилань

1. Лупов С. Ю. LabVIEW в примерах и задачах : учеб.-метод. материалы по программе повышения квалификации «Обучение технологиям National Instruments» / С. Ю. Лупов, С. И. Муякшин, В. В. Шарков. – Нижний Новгород, 2007. – 101 с.

2. Тревис Д. LabVIEW для всех / под ред. В. В. Шаркова, В. А. Гурьева. – 4-е изд., перераб. – М. : ПриборКомплект, 2011. – 912 с.

3. Суранов А. Я. LabVIEW 8.20 : справочник по функциям / А. Я. Суранов. – М. : ДМК Пресс, 2007. – 536 с.

4. Баторвин В. К. LabVIEW: практикум по электронике и микропроцессорной технике : учеб. пособ. для вузов / В. К. Баторвин, А. С. Бессонов, В. В. Мошкин. – М. : ДМК Пресс, 2005. – 182 с.

5. Бутырин П. А. Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций) / под ред. П. А. Бутырина. – М. : ДМК Пресс, 2005. – 265 с.

6. Баторвин В. К. LabVIEW: практикум по основам измерительных технологий : учеб. пособ. для вузов / В. К. Баторвин, А. С. Бессонов, В. В. Мошкин, В. Ф. Папуловский. – М. : ДМК Пресс, 2005. – 208 с.

Зміст