МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„ КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. ІГОРЯ СІКОРСЬКОГО ”

ЕКСПЛУАТАЦІЯ КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЧНИХ КОМПЛЕКСІВ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за спеціальністю
151 «Автоматизація та комп'ютерно-інтегровані технології»

Київ
КПІ ім. Ігоря Сікорського
2023

Експлуатація комп'ютерно-інтегрованих технологічних комплексів: комп'ютерний практикум [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського ; уклад.: Я. Ю. Жураковський, М. С. Піргач. – Електронні текстові данні (1 файл: 5 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2023. – 83 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 5 від 23.02.2023 р.)
за поданням Вченої ради інженерно-хімічного факультету (протокол № 1 від 30.01.2023 р.)

Електронне мережне навчальне видання

ЕКСПЛУАТАЦІЯ КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЧНИХ КОМПЛЕКСІВ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Укладачі: *Жураковський Ярослав Юрійович*
Піргач Микола Соловейович, канд. техн. наук, доц.

Відповідальний редактор *Ковалюк Д. О.*, канд. техн. наук, доц.

Рецензент *Гавриленко О. В.,* канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних систем і технологій КПІ ім. Ігоря Сікорського

Навчальний посібник містить матеріали для виконання комп'ютерного практикуму з Експлуатації комп'ютерно-інтегрованих технологічних комплексів. У посібнику розглянуто методику побудови програмного забезпечення для імітаційного моделювання системи керування та обробки даних в середовищі LabVIEW, що використовується для побудови систем керування у хіміко-технологічних процесах. Наведено практичні приклади побудови елементів програм, питання для самоперевірки та практичні завдання.

Посібник призначений для здобувачів ступеня бакалавра за спеціальністю 151 «Автоматизація та комп’ютерно-інтегровані технології» усіх форм навчання.

ЗМІСТ

ВСТУП

Програмне середовище LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) надає можливість перетворити персональний комп'ютер у сучасну SCADA систему. При наявності відповідного обладнання для зв’язку із об’єктом керування це програмне забезпечення дозволить збирати інформацію про об’єкт та здійснювати керування ним.

Існує також можливість за відсутності спеціального обладнання вводу-виводу створювати імітаційні системи керування із програмними моделями об’єктів та регуляторів та досліджувати їх взаємодію у реальному часі. Саме цій задачі присвячений комп’ютерний практикум з дисципліни «Експлуатація комп'ютерно-інтегрованих технологічних комплексів».

1. СТВОРЕННЯ ДИСПЛЕЙНОЇ МНЕМОСХЕМИ

Мета роботи: Знайомство із концепцією та засобами створення мнемосхем в середовищі LabVIEW, отримання вміннь створення мнемосхем.

Як правило при створенні мнемосхеми виділяють частину процесу (декілька установок, агрегатів) та декілька найбільш важливих технологічних параметрів.

На рис.3.1 наведено приклади оживленної дисплейної мнемосхеми.

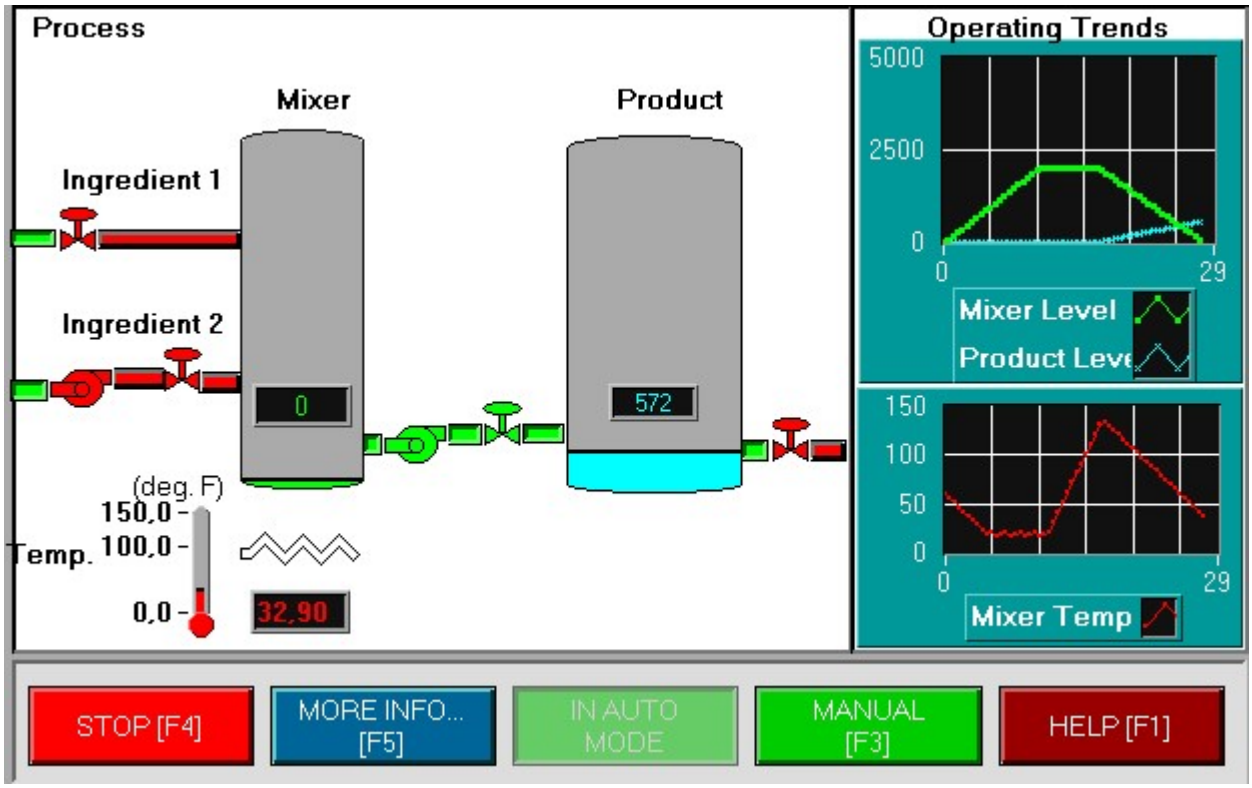


Рис.3.1а

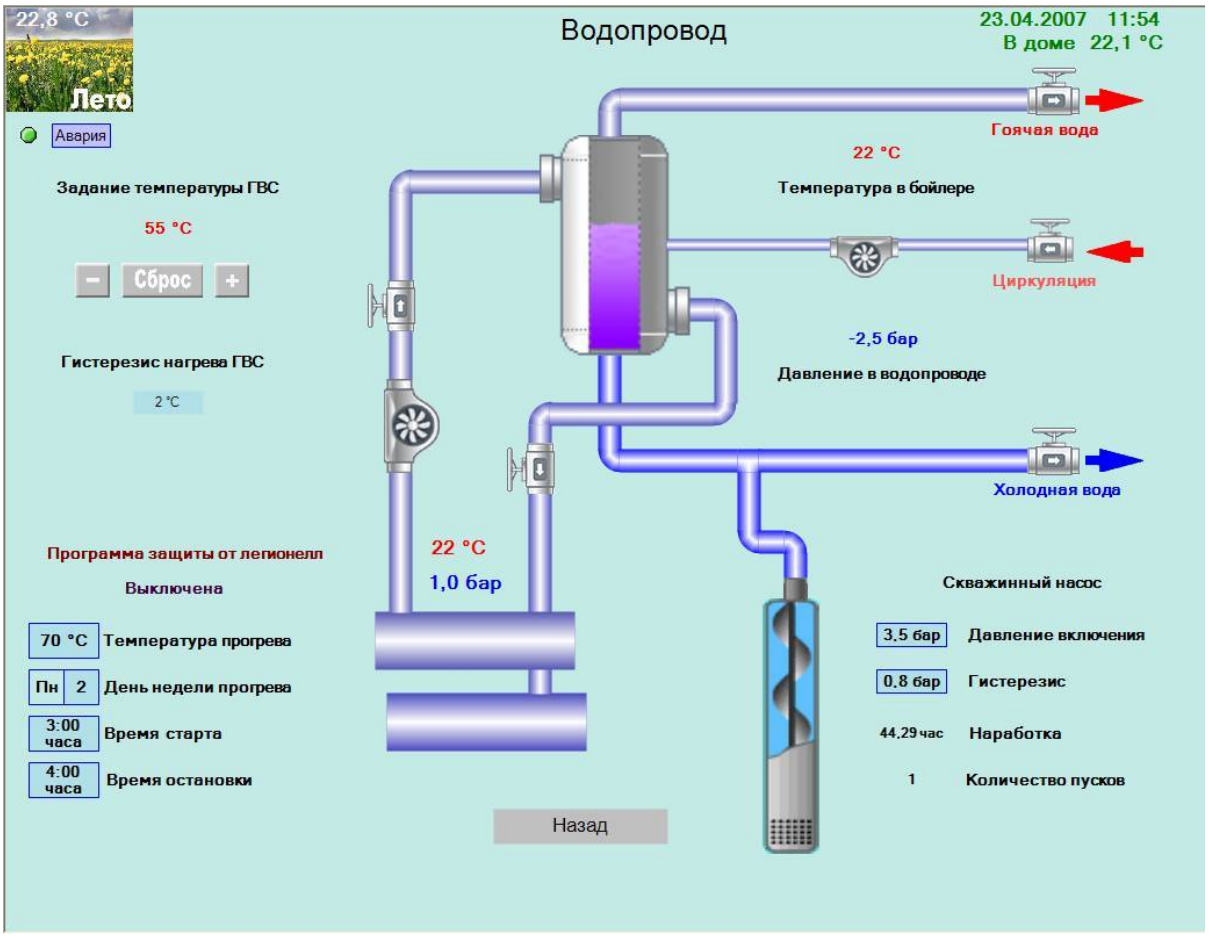

22,8 °C
Лето
Водопровод
23.04.2007 11:54
В доме 22,1 °C
Авария
Задание температуры ГВС
55 °C
Сброс
Гистерезис нагрева ГВС
2 °C
Программа защиты от легионелл
Выключена
70 °C
Температура прогрева
Пн
2
День недели прогрева
3:00 часа
Время старта
4:00 часа
Время остановки
Горячая вода
22 °C
Температура в бойлере
Циркуляция
-2,5 бар
Давление в водопроводе
Холодная вода
22 °C
1,0 бар
Скважинный насос
3,5 бар
Давление включения
0,8 бар
Гистерезис
44,29 час
Наработка
1
Количество пусков
Назад

Рис.3.1б

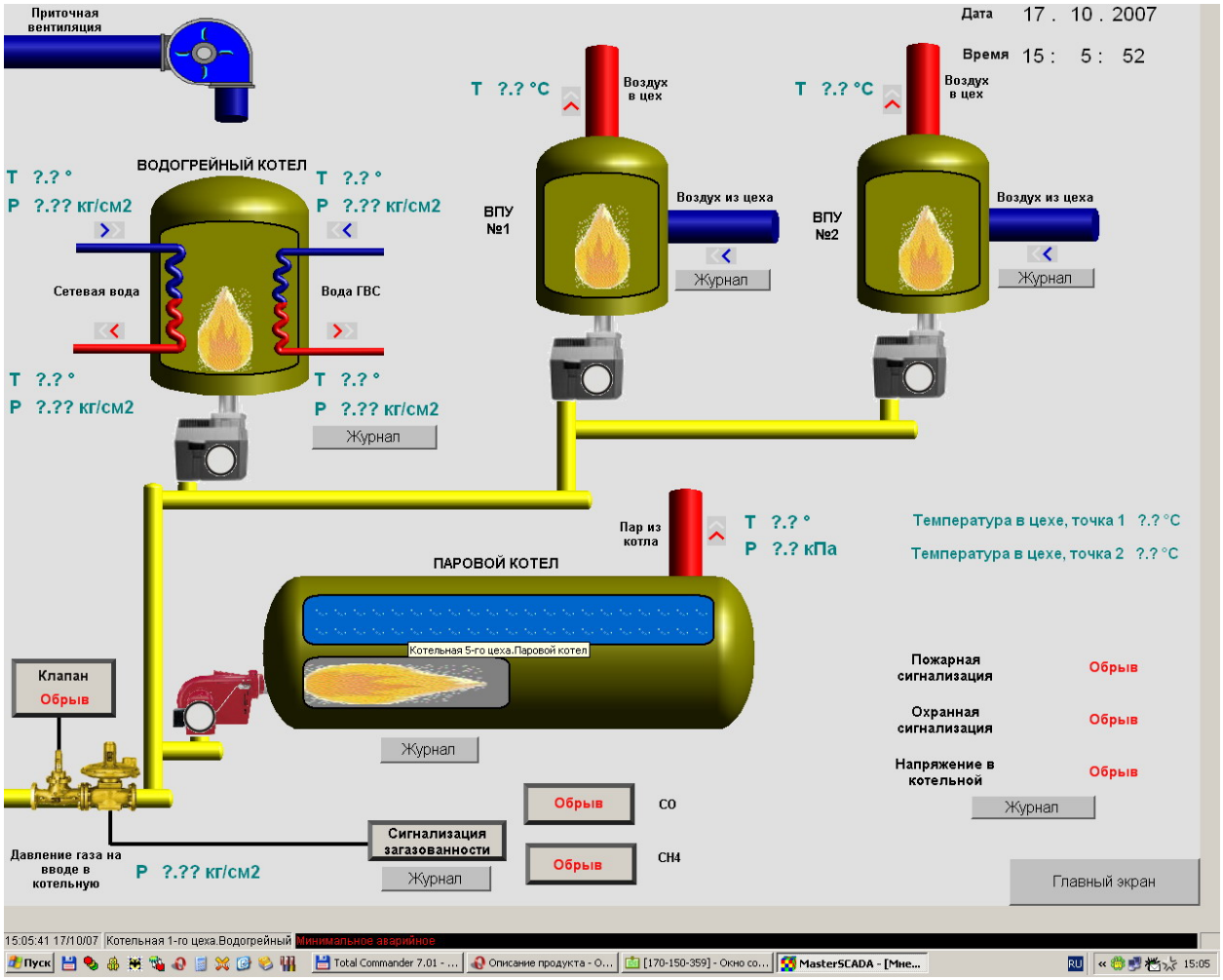

Приточная вентиляция
Дата 17 . 10 . 2007
Время 15 : 5 : 52
Т ?.? °С
Воздух в цех
ВПУ №1
Воздух из цеха
Журнал
Т ?.? °С
Воздух в цех
ВПУ №2
Воздух из цеха
Журнал
ВОДОГРЕЙНЫЙ КОТЕЛ
Т ?.? °
Р ?.?? кг/см2
Сетевая вода
Вода ГВС
Т ?.? °
Р ?.?? кг/см2
Журнал
Пар из котла
Т ?.? °
Р ?.? кПа
ПАРОВОЙ КОТЕЛ
Котельная 5-го цеха.Паровой котел
Журнал
Температура в цехе, точка 1 ?.? °С
Температура в цехе, точка 2 ?.? °С
Клапан
Обрыв
Пожарная сигнализация
Обрыв
Охранная сигнализация
Обрыв
Напряжение в котельной
Обрыв
Журнал
Обрыв
CO
Обрыв
CH4
Сигнализация загазованности
Журнал
Давление газа на вводе в котельную
Р ?.?? кг/см2
Главный экран
15:05:41 17/10/07
Котельная 1-го цеха.Водогрейный
Минимальное аварийное
Пуск
Total Commander 7.01 - ...
Описание продукта - О...
[170-150-359] - Окно со...
MasterSCADA - [Мне...
RU
15:05

Рис.3.1в

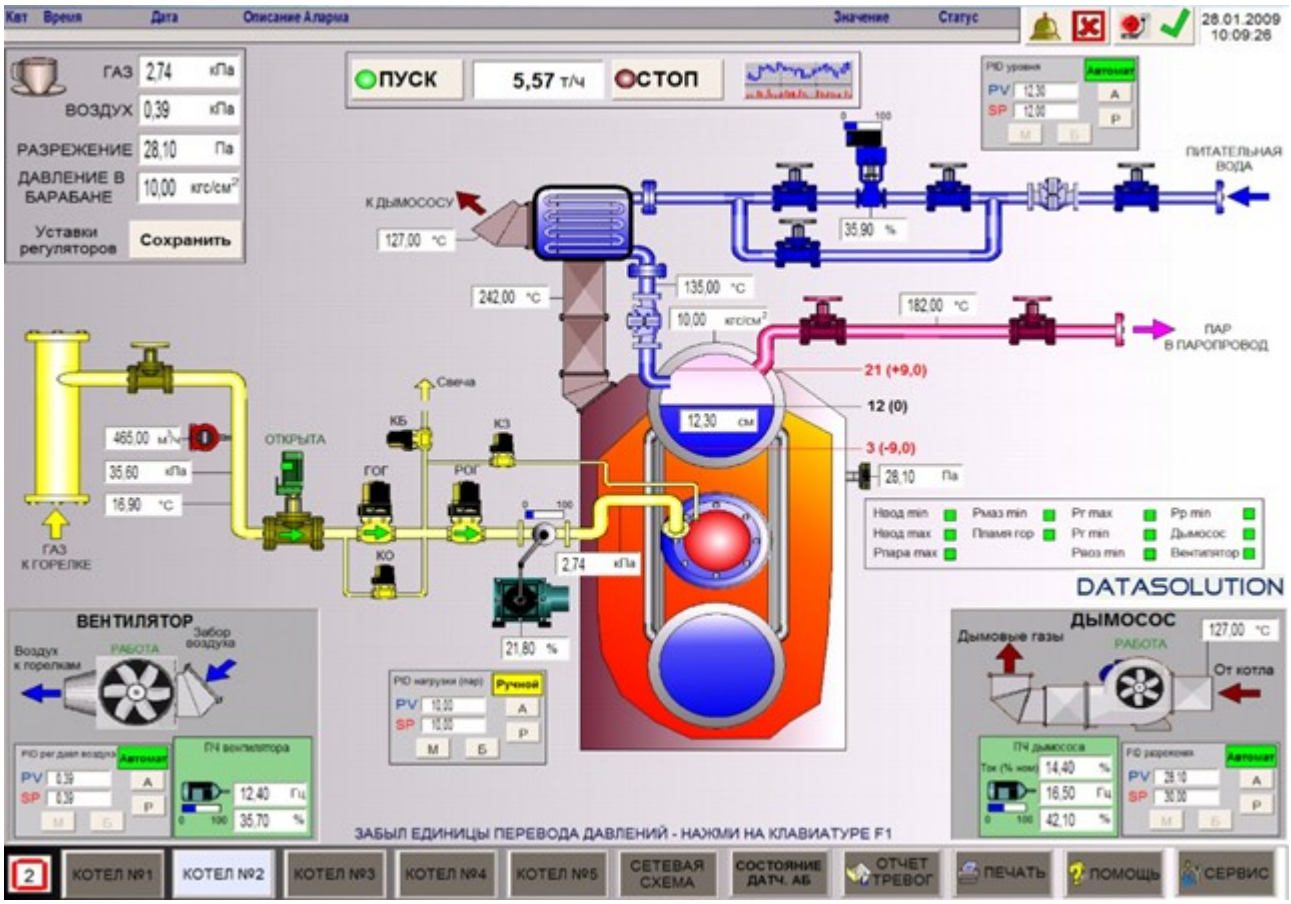


Рис.3.1г

Як правило при створенні мнемосхеми виділяють частину процесу (декілька установок, агрегатів) та декілька найбільш важливих технологічних параметрів.

Якщо технологічний процес складається з декількох апаратів, в цьому випадку крім загальної мнемосхеми процесу складають детальні мнемосхеми окремих апаратів та установок у вигляді ієрархічної системи екранів (рис. 3.2а). переключення між екранами можна здійснювати за допомогою кнопок (рис. 3.1г) що розташовані внизу екрана. В LabVIEW переключення між екранами можна реалізувати за допомогою Controls\Array&Cluster\Tab Control у версії 6і або Layout\Tab Control у версії 2019 (рис. 3.2б).

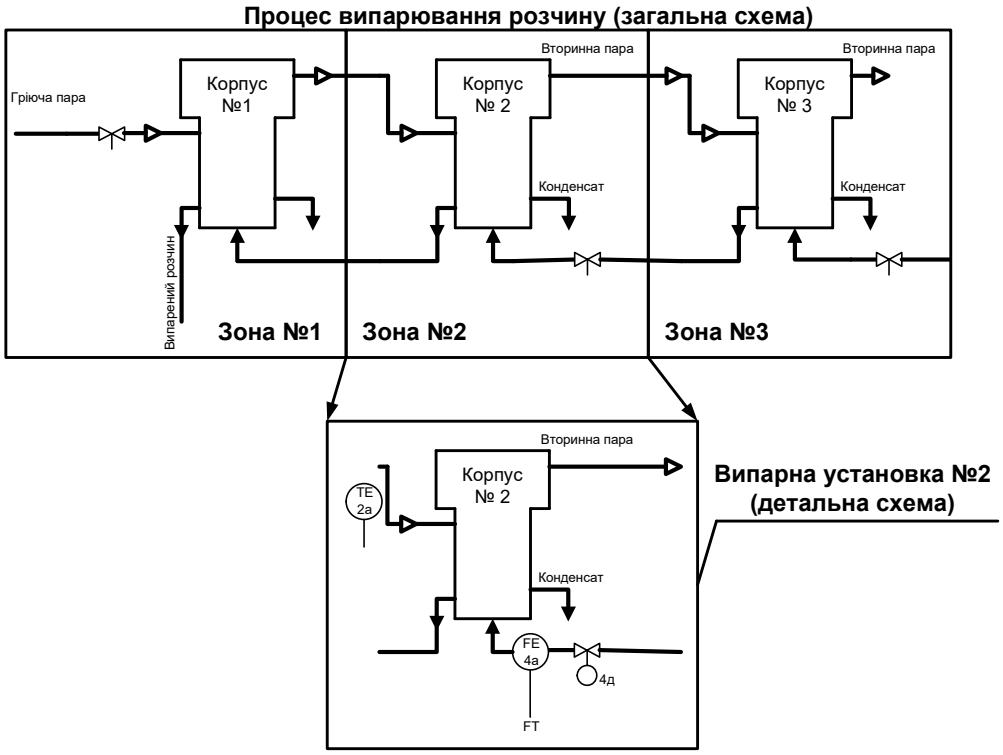


Рис. 3.2а

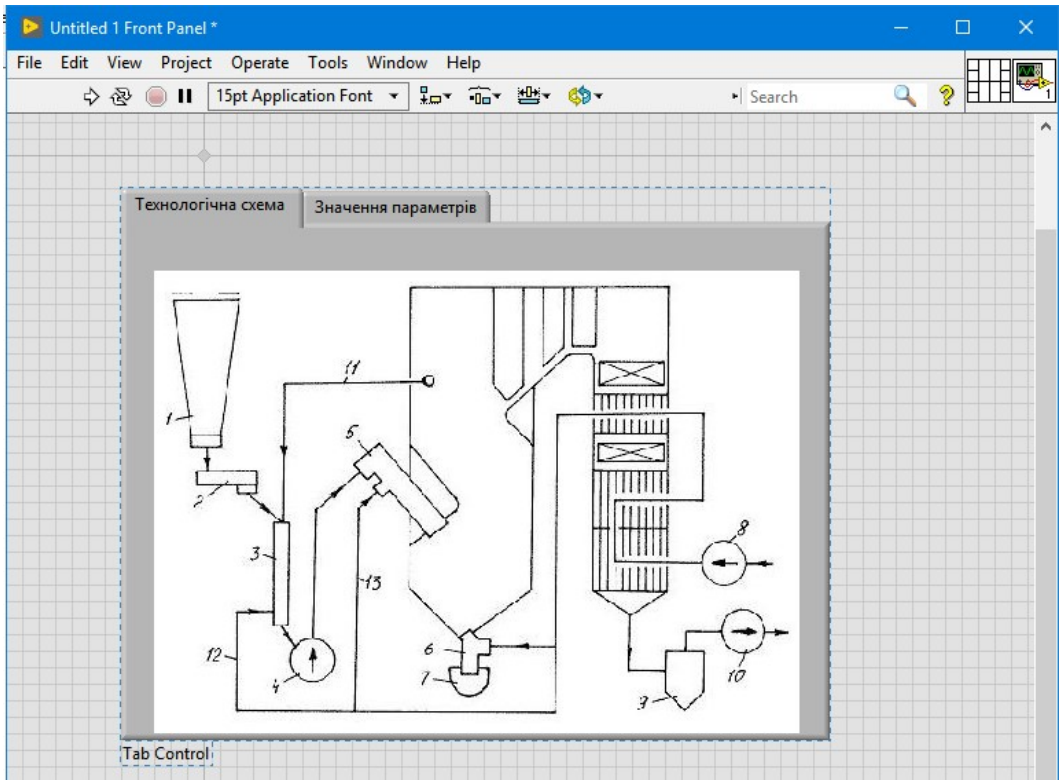


Рис. 3.2б

Будь-яка мнемосхема складається з простих елементів. Розглянемо докладно **створення зображення клапана**.

При створенні оживленої графічної мнемосхеми технологічного процесу часто буває недостатньо тих інструментів вводу / виводу, які надаються системою LabView. Розберемо приклад створення зображення регулюючого органу з виконавчим механізмом. Для цього виберемо в інструментах управління елемент Controls \ Classic_Controls \ Boolean \ Flat_Square_Button, як найбільш відповідний для розміщення потрібного нам зображення клапана. Сам клапан намалюємо в графічному редакторі Paint (рис. 3.3). Розміри зображення: 23 x 33 пікселя.

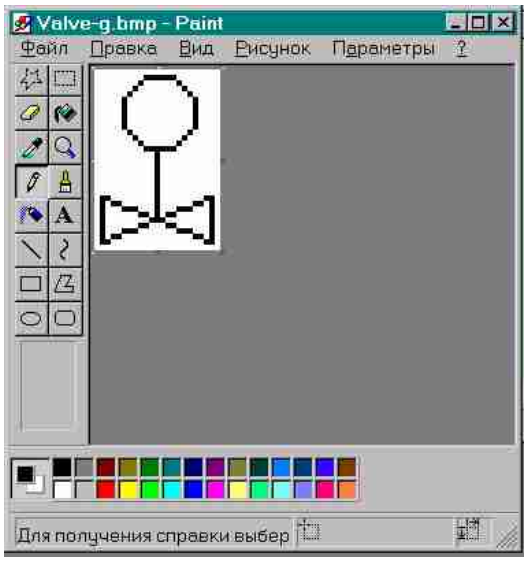


Рис. 3.3

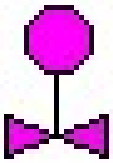


Рис. 3.4

Дане зображення будемо використовувати для позначення клапана в закритому стані. Для відображення включеного стану клапана (відкритий) виберемо заливку фіолетовим кольором (Рис. 3.4) . Збережемо обидва зображення у файлах з розширенням bmp : «valve-g.bmp» і «valve-g1.bmp» відповідно.

У версії програми LabVIEW 2019 графічне зображення може мати інший графічний формат, наприклад, jpeg.

Повернемося в LabVIEW. Для того, щоб помістити створені нами зображення у кнопку, необхідно завантажити їх з файлів. Для цього в контекстному меню інструменту Flat_Square_Button виберемо пункт "Advanced\Customise ...". Відкриється вікно редагування інструменту - кнопки. За допомогою команди меню "Edit\Import_Picture_from_file..." (або "Edit\Import_Picture_to_Clipboard..." в LabVIEW 2019) завантажимо картинку з «valve-g.bmp» і вибравши в контекстному меню кнопки пункт Import_Picture\False (або Import_Picture_from_Clipboard\False у LabVIEW 2019) встановимо зображення для закритого клапана (False). Аналогічно з файлу "valve - g1.bmp" встановимо зображення для True.

Закриємо вікно редагування інструменту, попередньо зберігши його у файлі з розширенням «*.Ctr». Надалі Ви можете використовувати створений інструмент через пункт " Controls \ Select_Control ..." або " Controls \ Select_a_Control ...".

У відповідь на запит про заміну оригінального інструменту на створений вами можемо відповісти ствердно. У підсумку отримуємо (рис. 3.5).

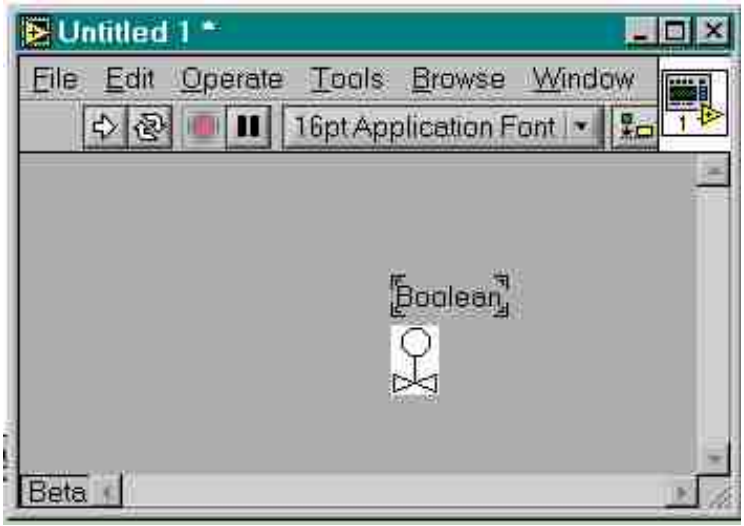


Рис. 3.5

Аналогічно можна створити зображення вертикально розташованого клапана, відрізків трубопроводу, насосів та іншого технологічного обладнання.

Для статичних зображень (не змінних в процесі роботи) можна просто вставляти зображення через пункти меню "Edit \ Import_Picture_from_file ..." і "Edit \ Paste" або просто перетягнувши назву файлу з папки у вікно передньої панелі LabVIEW. (Для версії LabVIEW 6i зображення повинно бути у форматі bmp).

Завдання:

Використовуючи отримані навички програмування в середовищі LabView, побудувати дисплейну мнемосхему технологічного процесу. Всі завдання можна розділити на такі підзадачі:

1. Створення графічного фонового зображення технологічного процесу (обладнання);

2. Вибір параметрів для індикації, способів індикації, елементів керування.

Контрольні питання:

1. Як розмістити зображення на передній панелі програми?

2. Як замінити зображення індикаторної лампи на інше?

3. Як змінити колір елементу передньої панелі?

2. СТВОРЕННЯ ІМІТАЦІЙНОЇ МОДЕЛІ

Мета роботы: Знайомство засобами створення імітаційних моделей в середовищі LabVIEW, отримання вміннь створення імітаційної моделі

Створення імітаційної моделі обєкту, що заданий аперіодичною ланкою 1-го порядку.

Представлення змішувача диференціальним рівнянням

Принципову схему змішувача як обєкту керування концентрацією зображено на рис. 4.1.

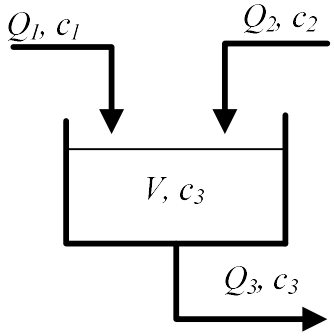


Рис. 4.1 – Принципова схема змішувача як об’єкт керування концентрацією волокнинної суспензії

Рівняння матеріального балансу змішувача має вигляд:

$$V\frac{dc_3}{dt} + Q_3c_3 = Q_1c_1 + Q_2c_2\,, \qquad (4.1)$$

де V – об’єм змішувача;

Q_1, Q_2 і Q_3 – витрати вхідних компонентів та суміші, м^3/с;

c_1, c_2 і c_3 – концентрація вхідних компонентів та суміші, %.

Диференціальне рівняння (4.1) можна подати і так:

$$\frac{V}{Q_3}\cdot\frac{dc_3}{dt}+c_3=\frac{c_2}{Q_3}Q_2+\frac{c_1}{Q_3}Q_1$$

або як аперіодичну ланку першого порядку

$$T_{11}\frac{dy}{dt}+y(t)=k_{11}u(t)+k_{12}f(t), \qquad (4.2)$$

де $T_{11}=\frac{V}{Q_3}$ – стала часу змішувача, с;

dt – прирощення часу за один крок розрахунку;

$k_{11}=\frac{c_2}{Q_3}$ – коефіцієнт підсилення каналу керування 11, $\frac{\%}{м^3/с}$;

$k_{12}=\frac{c_1}{Q_3}$ – коефіцієнт підсилення каналу збурення 12, $\frac{\%}{м^3/с}$;

$y=c_3$ – вихідна змінна змішувача;

$u=Q_2$ – керувальне діяння (вхідна змінна);

$f=c_1$ – збурювальне діяння.

Запишемо (4.2) в дискретній формі, тобто замість dt використаємо прирощення Δt; dy є прирощенням параметру y, тобто

$$dy=\Delta y=y_i-y_{i-1},$$

де i та i–1 є індексами параметру y в i-й та i–1-й момент часу. Таким чином рівняння (4.2) можна подати в такий дискретній формі

$$T_{11}\frac{y_i - y_{i-1}}{\Delta t} + y_i = k_{11}u + k_{12}f \tag{4.3}$$

Зробимо перетворення для того, щоб виразити y_i , тобто значення параметру обєкта на поточний момент часу, через інші змінні:

$$T_{11}\left(y_i - y_{i-1}\right) + y_i\Delta t = k_{11}u\Delta t + k_{12}f\Delta t$$

$$y_i\left(T_{11} + \Delta t\right) = k_{11}u\Delta t + k_{12}f\Delta t + y_{i-1}T_{11}$$

$$y_i = \frac{k_{11}u\Delta t + k_{12}f\Delta t + y_{i-1}T_{11}}{T_{11} + \Delta t} \tag{4.4}$$

Функцію (4.4) можна запрограмувати в системі LabVIEW. Для цього використаємо структуру While і Formula Node. Для передачі даних про стан параметру y на попередній ітерації, тобто значення y_{i-1}, використано зсувовий регістр Shift register (рис. 4.2). Значенню параметру y_i відповідає змінна «yi», значенню y_{i-1} відповідає змінна «yi1». Початкове значення параметру y_i задано елементом керування «yi_0».

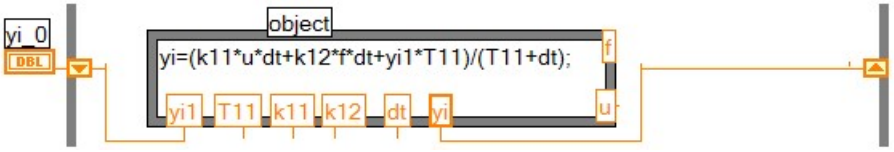


Рис. 4.2 – Фрагмент програми із моделлю обєкта

Розглянемо рівняння ПІД-регулятора

$$U(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\,\frac{de(t)}{dt} \right]$$

де $U(t)$ – регулювальний вплив; $e(t)$ – відхилення регулювальної величини від заданого значення; T_i, T_d, – сталі часу інтегрування та диференціювання.

Замінив $e(t)$ на $e(nT)$, а $U(t)$ на $U(nT)$, де T – період дискретизації, n=0, 1, 2 ..., перейдемо до гратчастої функції

$$U(nT) = K_p\left[e(nT) + \frac{T}{T_i}\sum_{i=1}^{n}\frac{e(iT)+e((i-1)T)}{2} + T_d\left[\frac{e(nT)-e((n-1)T)}{T}\right]\right].$$

Віднімемо з обох частин рівняння величину $U((n-1)T)$ та отримаємо рекурентне рівняння, що є більш зручним для програмування:

$$U(nT) = U((n-1)T) + K_p\left[\begin{array}{l}\{e(nT)-e((n-1)T)\}+\frac{T}{2T_i}\{e(nT)+e((n-1)T)\}+\\ +\frac{T_d}{T}\left[e(nT)-2e((n-1)T)+e((n-2)T)\right]\end{array}\right],$$

або після перетворень

$$U(nT) = U((n-1)T) + K_p\left(1 + \frac{T}{2T_i} + \frac{T_d}{T}\right)e(nT) +$$
$$+ K_p\left(-1 + \frac{T}{2T_i} - \frac{2T_d}{T}\right)e((n-1)T) +$$
$$+ \frac{K_p T_d}{T}e((n-2)T).$$

Відхилення e(nT) обчислюється як

$$e(nT) = SP - y(nT),$$

де SP – задане значення регулювальної величини «SetPoint»; $y(nT)$ – значення регулювальної величини у момент часу nT або «ProcessVariable» (PV).

Запрограмуємо функцію в LabVIEW (рис. 4.3). Використаємо такі позначення змінних:

Змінна	Позначення
$U(nT)$	un
$U((n-1)T)$	un_1
$e(nT)$	e
$e((n-1)T)$	e1
$e((n-2)T)$	e2

Для отримання значень відхилень $e((n–1)T)$ та $e((n–2)T)$ використовуємо зсувові регістри на два та три кроки назад. Початкові значення відхилень задаємо нульовими.

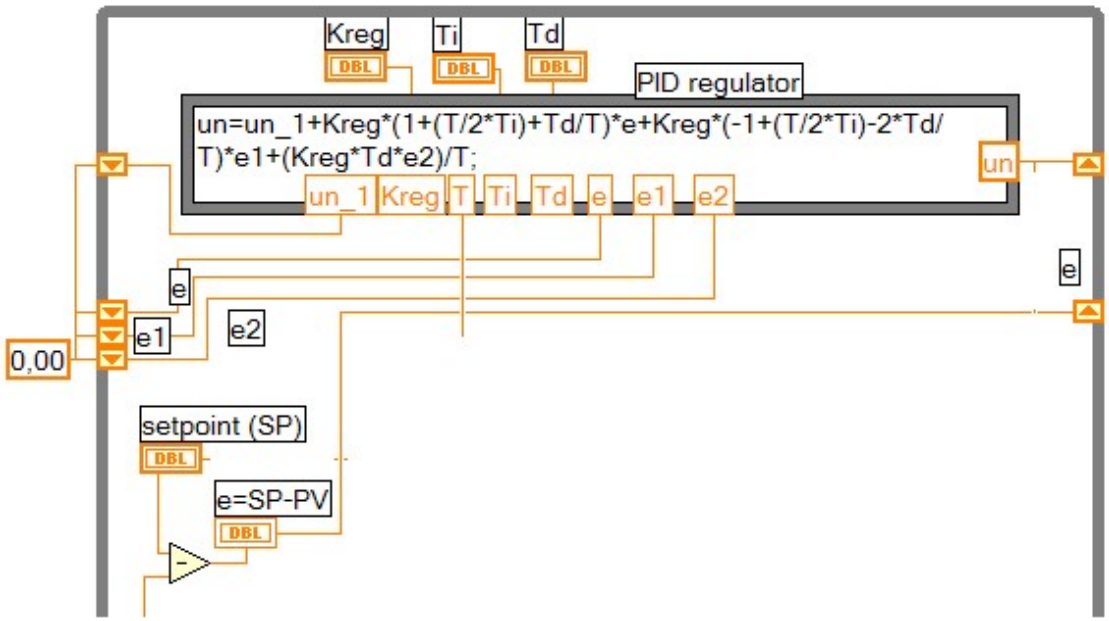


Рис. 4.3 – ПІД-регулятор

Структура системи керування

Для моделювання оберемо одноконтурну дискретну замкнену систему із дискретним аналогом ПІД-регулятора (рис. 4.4).

Вихідна змінна системи $y(t)$ змінюється від дії на об’єкт керування (ОК) збурення $f(nT)$ через передавальну функцію $F(z)$. З метою ліквідації дії цього збурення вихідна змінна системи порівнюється із її задавальним діянням SP, і одержаний сигнал відхилу $e(nT)$ після квантування у часі подається на дискретний регулятор $D(z)$, який своїм керівним діянням $u(nT)$ впливає на ОК з передавальною функцією $W(z)$. Також регулятор відпрацьовує зміну задавального діяння SP.

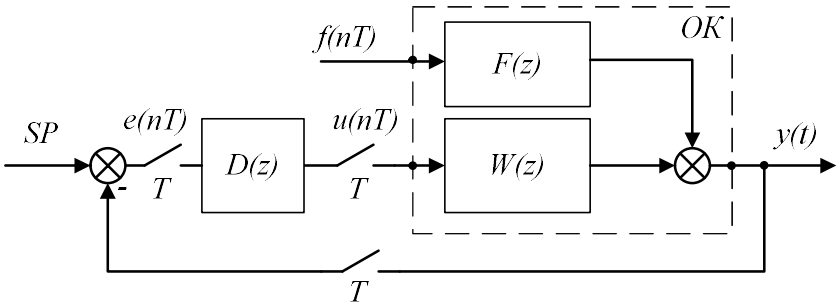
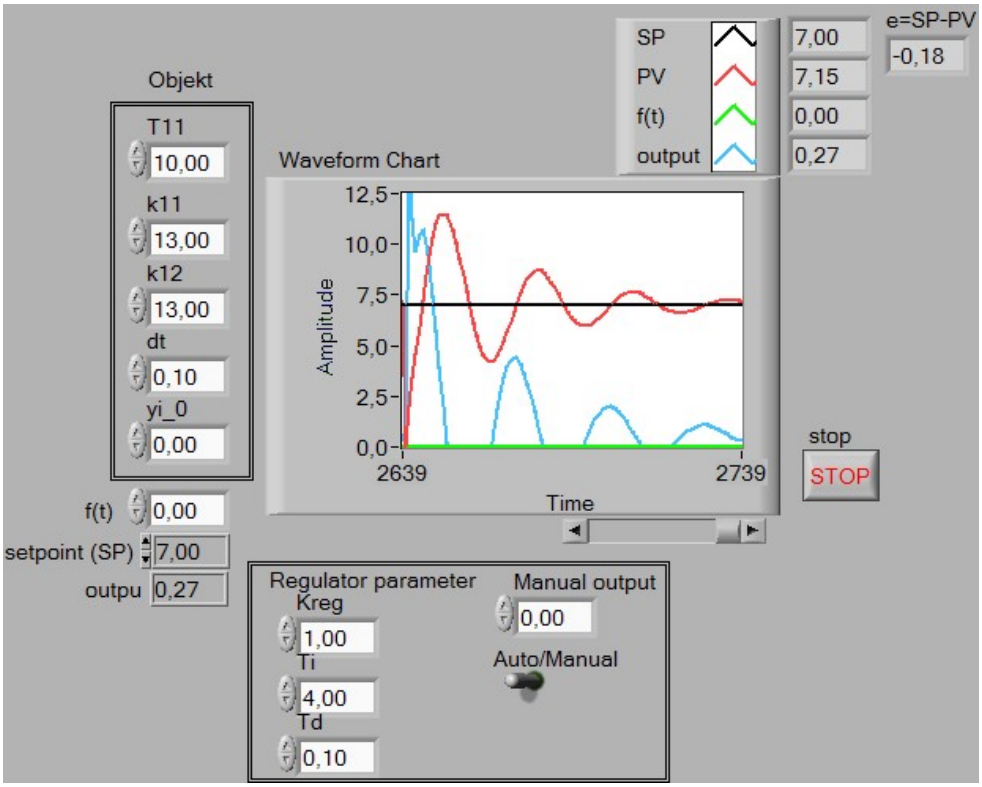


Рис. 4.4 – Структурна схема системи керування

За обраною структурою (рис. 4.4) зберемо схему, що показано на рис. 4.5. До схеми з рис. 4.2 додано ПІД-регулятор (рис. 4.3). На на вхід ПІД-регулятору подано значення відхилення «e», що обчислено як різницю між вихідною змінною «yi_1» та задавальним діянням SP, що задається задатчиком «setpoint (*SP*)». Керувальне діяння «un» з виходу регулятора подається на обєкт як змінна «u» і на передній панелі позначено як «output». У схемі передбачена можливість вибору режиму ручного керування, коли регулятор відключається перемикачем «Auto/Manual» і керувальне діяння задається задатчиком «Manual output». Збурення f задається задатчиком «f(t)». Параметри регулятора «Kreg», «Ti», «Td» та параметри обєкту «T11», «k11», «k12», «dt», «yi_0» можна змінювати з передньої панелі.

Результати обчислень відображаються на графіку «Waveform Chart» і формуються за допомогою «Bundle».

Front Panel



Block Diagram

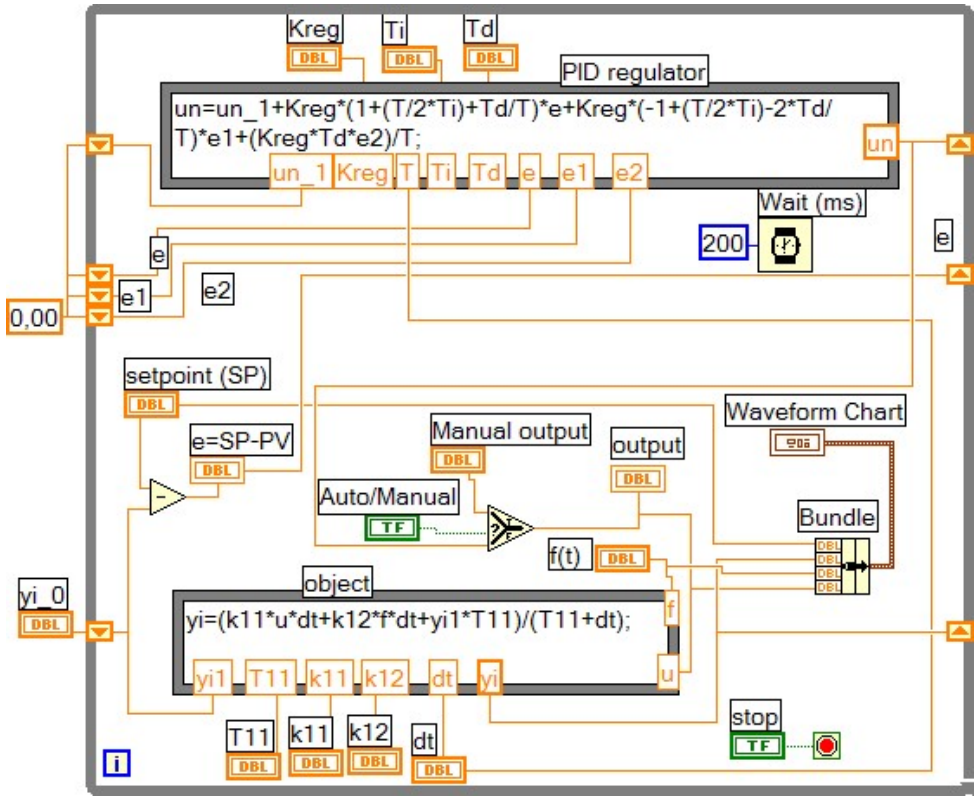


Рис.4.5

Завдання:

1. Створити імітаційну модель процесу (математичного опису зміни параметрів у часі). Можливе використання спрощеної математичної моделі, однак слід враховувати зв'язок між параметрами, наприклад, між витратою пари в теплообміннику та температурою речовини на виході теплообмінника тощо.

2. Створити імітаційну модель декілької регуляторів (П, ПІ, позиційного).

Контрольні питання:

1. Як здійснити передачу даних в циклі між ітераціями?

2. Як задати початкові значення змінних?

3. Як здійснити переключення між режимами роботи «ручний» та «автоматичний»?

3. СТВОРЕННЯ ТЕХНОЛОГІЧНОЇ СИГНАЛІЗАЦІЇ

Мета роботи: Знайомство із концепцією та засобами створення технологічної сигналізації в *LabVIEW,* отримання вміннь програмування та налаштування технологічної сигналізації

Створення візуальної сигналізації

Сигналізації, призначені для сповіщення обслуговуючого персоналу про стан контрольованих об'єктів.

Аварійно-попереджувальна сигналізація служить дли подачі сигналу про ненормальну роботу установок, переведених на автоматичне управління.

Сигнали зазвичай бувають звукові і світлові. Звукові сигнали привертають увагу чергового. Світлові сигнали тією чи іншою мірою розшифровують характер несправності та вказують на об'єкт, де виникла ця несправність.

Схеми сигналізації за призначенням можна класифікувати таким чином:

1) сігналізація положення сповіщає про стан контрольованих об'єктів (включені або відключені магнітні пускачі, контактори, вимикачі, відкриті чи закриті засувки, заслонки та інші подібні пристрої);

2) командна сигналізація призначена для передачі заздалегідь визначених команд з одного приміщення в інше за допомогою різних світлових та звукових сигналів;

3) сигналізації дії та захисту автоматики – це інформація про роботу того чи іншого виду автоматичного захисту або блокування;

4) технологічна сигналізація призначена для сповіщення про порушення нормального ходу технологічних процесів (про відхилення від встановленого значення температури, тиску, рівня і т. д.). При цьому технологічна сигналізація може бути двох видів:

а) попереджувальна сигналізація ненормальних, але поки ще допустимих значень контрольованих або регульованих параметрів. Поява попереджувальних сигналів вказує обслуговуючому персоналу на необхідність прийняття певних заходів для усунення несправностей, що виникли;

б) аварійна сигналізація неприпустимих значень контрольованих або регульованих параметрів, аварійний стан на окремих ділянках технологічного процесу або аварійні відключення контрольованих об'єктів. Поява аварійних сигналів часто супроводжується дією пристроїв автоматичного захисту і блокування. Аварійна сигналізація вимагає негайного втручання персоналу.

За принципом дії схеми сигналізації поділяють на:

1) схеми сигналізації з індивідуальним зніманням звукового сигналу;

2) схеми сигналізації з центральним (загальним) зніманням звукового сигналу; ці схеми оснащені єдиним пристроєм, що дозволяє відключати звуковий сигнал, зберігаючи індивідуальний світловий сигнал;

3) схеми сигналізації з центральним зніманням звукового сигналу і повторністю дії; вони відрізняються від попередніх схем здатністю повторно подавати звуковий сигнал при спрацьовуванні.

Найважливішим завданням системи моніторингу є сигналізація про наступні події:

- Вихід параметрів середовища за допустимі межі;
- Обрив ліній підключення датчиків;
- Обрив зв'язку в мережі обміну даними;

- Непрацездатність окремих вимірювальних приладів;

- Збої в роботі системи.

У зазначених випадках система видає світлові і звукові сигнали, а також реєструє їх у протоколі подій. Протокол подій заноситься в архів.

Залежно від типу подій в системі може бути передбачено *квитування* подій. Це означає, що черговий оператор повинен підтвердити, що він виявив подію і зробив необхідні дії. Квитування також реєструється в протоколі подій.

Робота попереджувальної і аварійної сигналізації при виході параметрів середовища за допустимі межі показані на рис. 5.1. На вимогу замовника може бути встановлена як світлова, так і звукова сигналізація.

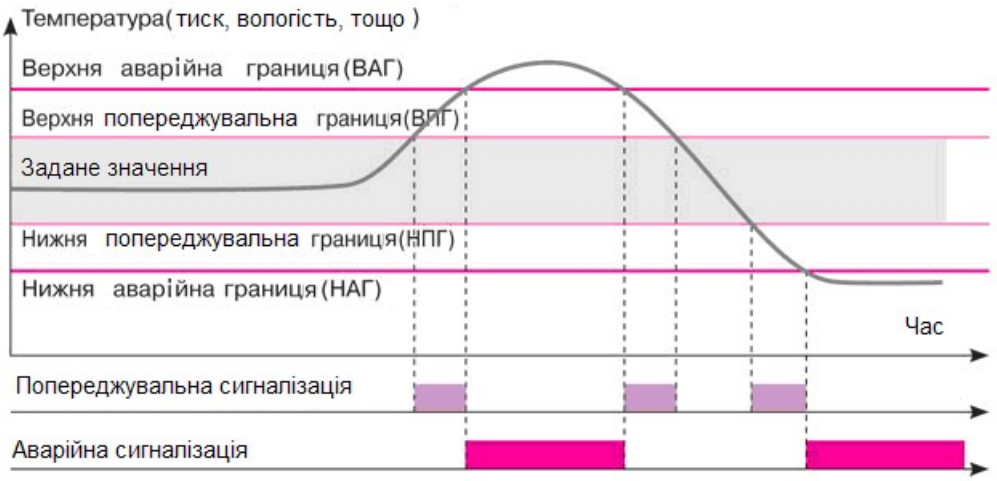


Рис. 5.1. – Робота попереджувальної і аварійної сигналізації

Рівні сигналізації

Сигналізації доступні різноманітних типів, з кількома рівнями тривоги. У всіх процесах відбуваються порушення, які можуть суттєво змінити процес експлуатації від нормального режиму. За допомогою систем управління, процеси мають потрапляти в діапазон допустимих

нормальних робочих меж. Коли процес відхиляється за нормальні межі, повинна спрацювати сигналізація.

Для більшості процесів, для безпечної роботи є, як мінімум, два рівня тривоги: попереджувальна (warning) і критична сигналізація (critical alarm). Попереджувальна сигналізація говорить операторам, що процес відхилився за межі допустимого діапазону і дає їм час і можливість прийняти коригуючі заходи для попередження впливу на якість продукції, навколишнє середовищє і правила техніки безпеки. Якщо правильні дії не будуть здійснені або здійснені не достатньо швидко щоб виправити ситуацію, може спрацювати критична сигналізація. Критична сигналізація повідомляє операторам, що технологічні параметри є у небезпечній близькості до порушення дозволених меж значень параметрів. У багатьох випадках критична сигналізація вимагає системних дій із зупинки процесу, доки проблема не буде розв’язана.

Умови, при яких спрацьовує попереджувальна і критична сигналізація є тими умовами, що визначені для процесу як такі, що перевищують дозволені межі. Невизначеність вимірювання завжди повинна бути врахована, тому що всі пристрої в системі управління мають якусь припустиму помилку. Установка сигналізації точно на прийнятному діапазоні для процесу може дозволити виміряному значенню змінюватися в межах цього діапазону, хоча фактичне його значення буде лежить поза діапазоном. Виконуючи аналіз помилок і статистичну теорію розподілу, межі спрацювання сигналізації за необхідністю можна регулювати. Інформація про обрані рівні сигналізації повинна бути добре документована.

На рис. 5.2 подано візуальне представлення діапазонів сигналізації.

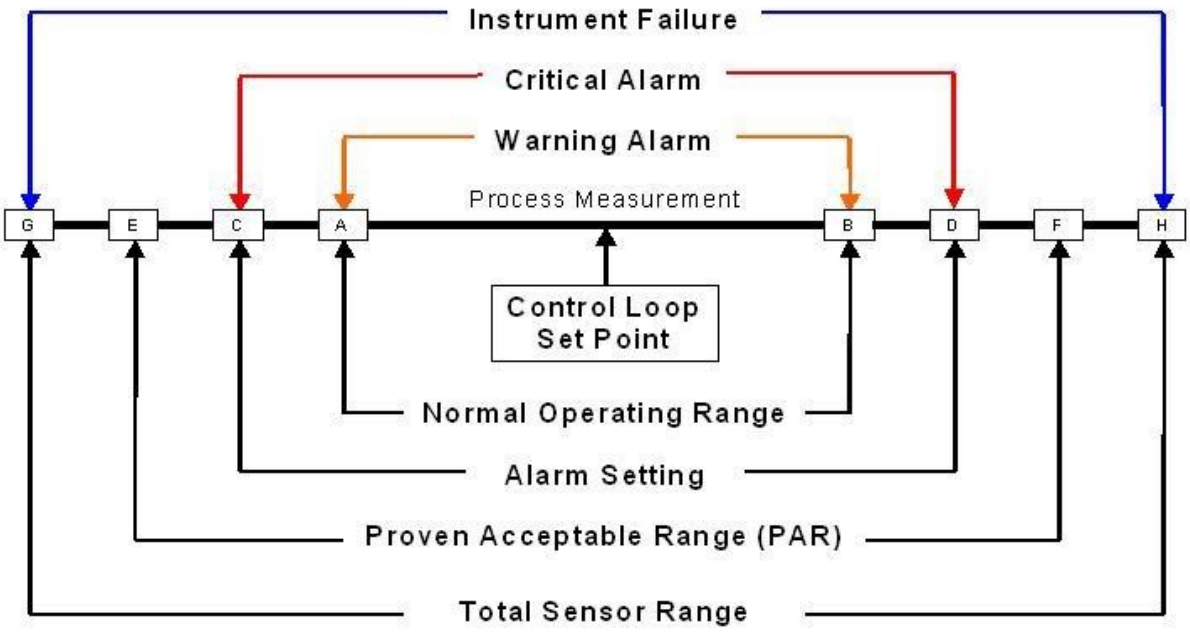


Рис. 5.2

Як видно в центрі на рис. 5.2, уставка контуру управління (Set Point) є оптимальним значенням керованого параметру для процесу (наприклад, оптимальна температура і концентрації реагентів для реакції). Неможливо підтримувати процес точно на цьому значенні, тож є цілий діапазон «нормальної» експлуатації (Normal Operating Range), всередині якого процес вважається таким, що працює прийнятним чином. Попереджувальна сигналізація буде задіяна, коли значення параметру виходить поза межі цього діапазону (менше, ніж A або більше, ніж B), що дає час для дій, які необхідні для повернення параметру у діапазон «нормальної» експлуатації. Критична сигналізація буде задіяна, якщо процес виходить за межі настройки сигналізації (менше, ніж C або більше, ніж D). Цей параметр визначається діапазоном охорони прийнятного діапазону процесу «Process Acceptable Range» (PAR), видно, що встановлення сигналізації лежить всередині PAR. Проміжки між E і C, а також між D і F вимірюються із невизначеністю. PAR повинен бути

всередині загального діапазону вимірювання датчика, вихід за який означає збій у роботі приладу.

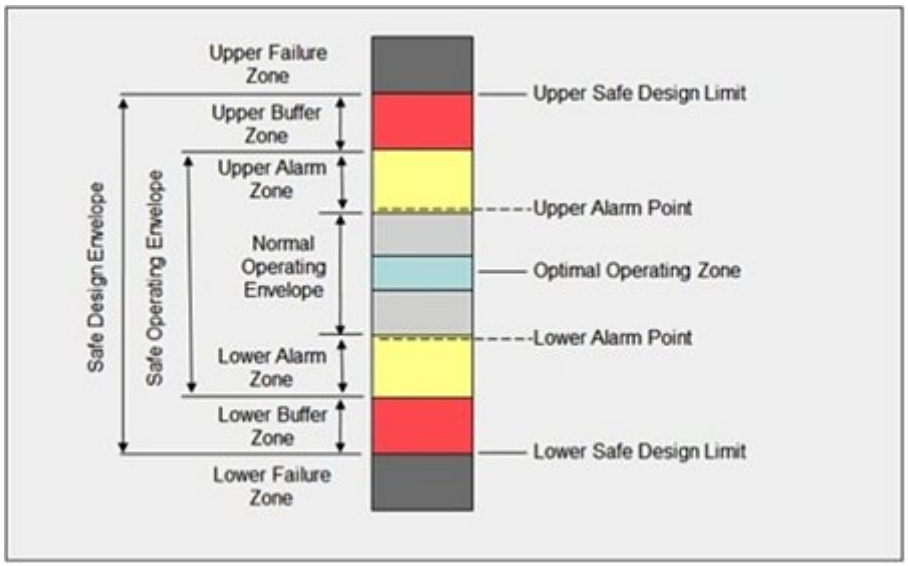


Рис. 5.3

Використання кольорів для світлової сигналізації регламентується стандартами МЭК 60073–96 (табл. 5.1).

Для привернення уваги оператора можна застосовувати миготливий сигнал, наприклад для позначення:

а) вимоги негайної дії (після підтвердження оператором переходить у постійне світіння);

б) відмінності між номінальним і фактичним станом обладнання;

в) зміна стану обладнання (миготіння під час перехідного процесу).

Таблиця 5.1 – Функціональні значення кольорів для кодування

Колір	Смислове значення		
	Безпека людей та обладнання	Стан процесу	Стан обладнання
Червоний	Небезпека	Критичний стан	Несправність
Жовтий	Увага	Перехідний (зміна умов або стан, що передує зміні умов)	Перехідний (зміна умов або стан, що передує зміні умов)
Зелений	Безпека	Нормальний	Нормальний
Синій	Спеціальне (може мати будь яке значення, крім функціонального для червоного, жовтого та зеленого)		
Білий, сірий	Не мають спеціального значення		

Кольори, які використовуються для відображення інформації на відеодисплеї та їх смислові значення повинні відповідати таблиці 5.1. Кольори повинні контрастувати з суміжними кольорами і з фоном дисплея. Смислове значення кожного кольору повинно застосовуватися узгоджено при використанні декількох дисплеїв і інших пов'язаних приладів. Для безпеки кольори повинні бути яскравими, насиченими і контрастними. Кольори для інформації низького пріоритету можуть бути тьмяними і ненасиченими.

Кодування графічними символами та / або положенням

Візуальні коди, що використовують в якості засобів кодування форму і (або) положення, можуть застосовуватися таким чином:

а) як основний код;

б) як додатковий до основного коду, наприклад графічні символи на додаток до квітам, щоб виключити помилки, що мають місце у людей з дефектним колірним сприйняттям.

Смислові значення графічних символів представлені в таблиці 5.2.

Таблиця 5.2 - Значення графічних символів для кодування

Графічне позначення	Смислове значення		
	Безпека людей або обладнання	Стан процесу	Стан обладнання
	Небезпека	Критичний	Несправний
	Увага	Перехідний	Перехідний
	Безпека	Нормальне	Нормальне
	Спеціальне		
	Не має спеціального значення		

Звукова апаратура розрізняється за тембром звуку:

а) дзвінки;

б) гудки;

в) сирени.

Загальні правила для позначення звукових сигналів при кодуванні інформації наведено в табл. 5.3.

Неперервний звуковий сигнал має застосовуватися тільки у деяких, строго обмежених випадках (наприклад при зміні небезпечного або перехідного стану на безпечний).

Таблиця 5.3 – Смислові значення звукових сигналів

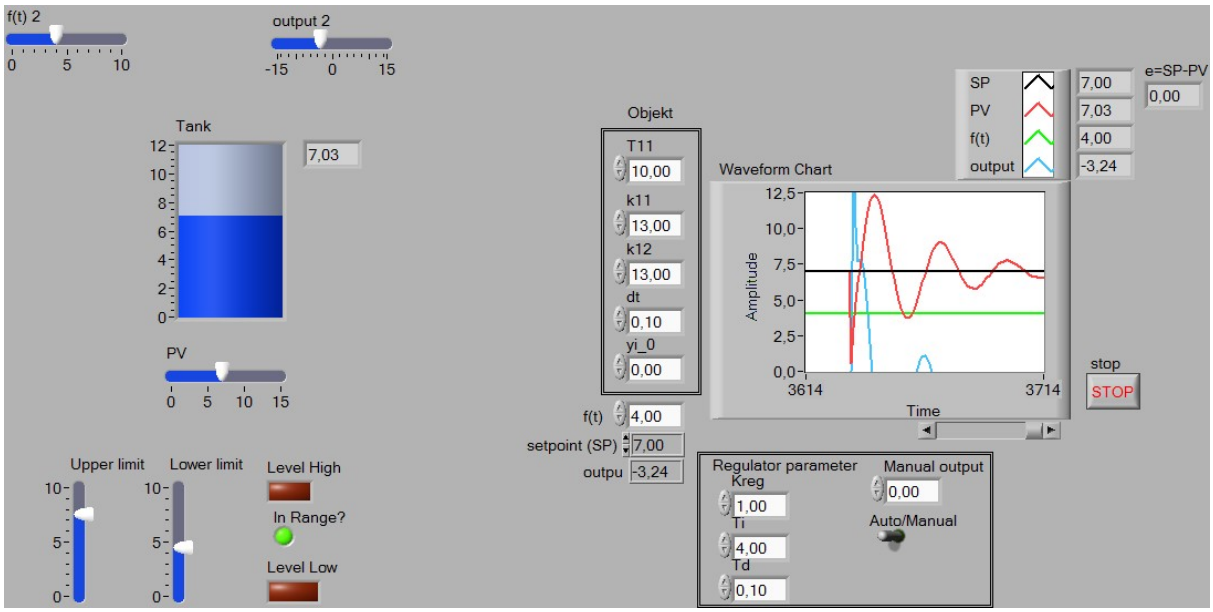
Вид звуку	Смислове значення		
	Безпека людей та обладнання	Стан процесу	Стан обладнання
Протяжний, різкий, що посилюється	Небезпека	Критичний стан	Несправність
Переривчастий із постійним інтервалом	Увага	Перехідний	Перехідний
Неперервний із постійним рівнем	Безпека	Нормальний	Нормальний
Попеременні звуки	Спеціальне		
Інші звуки	Не мають спеціального значення		

Звукові сигнали не повинні застосовуватися при безпечному стані (тиша).

В прикладі на рис. 5.4 реалізована сигналізація виходу значення парамеру за межі допустимих значень, що визначаються задатчиками «Upper limit» (Верхня межа) та «Lower limit» (Нижня межа). Сигналізація реалізована за допомогою ламп які вмикаються за результатом простого порівняння поточного значення параметру із заданими значеннями границь діапазону. В якості альтернативи можна використовувати інструмент Function\Comparison\In Range and Coerce, який визначає чи

знаходиться параметр в межах заданого діапазону і дає відповідь у вигляді булевої змінної.

Front Panel



Block Diagram

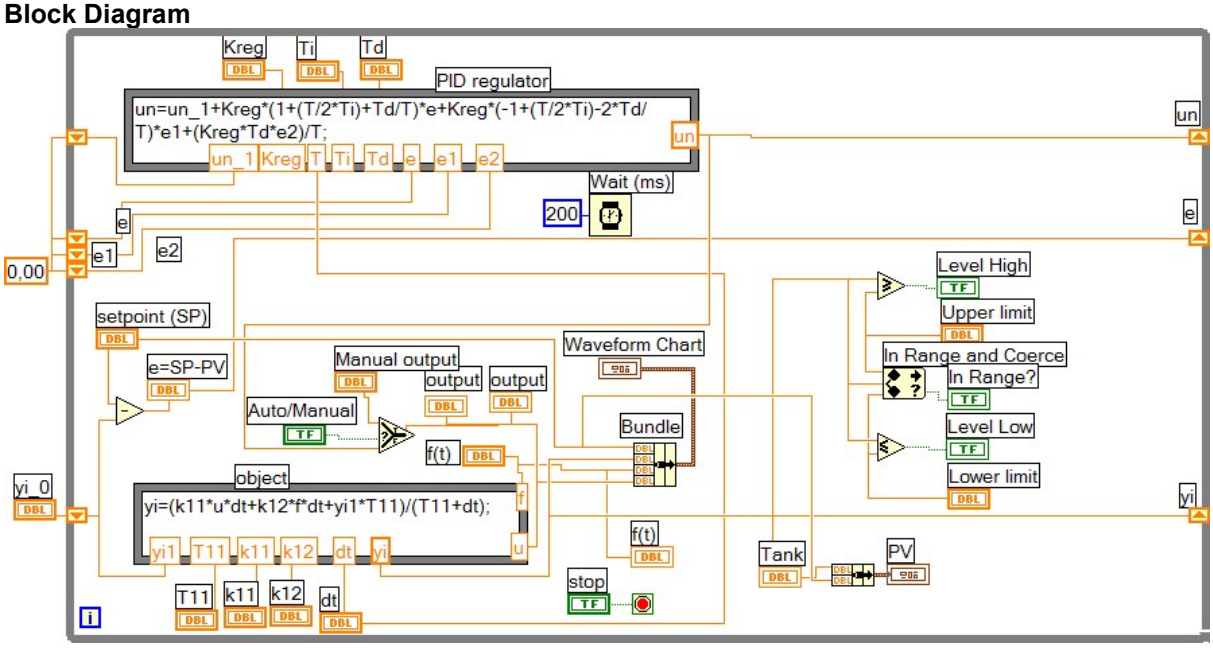


Рис.5.4

Завдання:

1. Вибрати параметри для сигналізації, визначити межі попереджувальної та аварійної сигналізації.

2. Створити візуальну попереджувальну та аварійну сигналізації.

3. Створити звукову аварійну сигналізацію.

Контрольні питання:

1. Що таке діапазони сигналізації?

2. Якими кольорами кодуються режими роботи?

3. Які звукові сигнали використовуються для попереджувальної та аварійної сигналізації?

4. СТВОРЕННЯ ЗВІТІВ

Мета роботи: Знайомство із концепцією та засобами створення звітів в *LabVIEW,* отримання вміннь програмування створення звітів

Під час проведення експериментів, розрахунків чи роботи системи керування виникає потреба автоматизованого документування результатів роботи, що реалізується набором інструментів «Створення звітів» (Report Generation).

Звіт являє собою документ, що може бути роздрукований на принтері (*Standard Report*) або збережений на диск (*HTML*-документ). Звіт формується послідовним додаванням необхідних текстових, числових та графічних даних із заданням параметрів форматування (шрифт, відступи, вирівнювання тощо).

Для створення простого текстового звіту використовують блок «Function\Report_Generation\Easy_Text_Report», але цей блок не дозволяє вставляти в звіт числові дані, таблиці та графіки. Отже, скористаємося повноцінною процедурою створення звіту.

Для початку створення звіту скористаймося блоком «Function\Report Generation\New_Report».

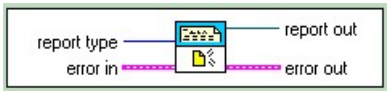


Рис. 6.1 – Блок «New_Report»

Вхідними даними для блоку «New_Report» є «report type» – тип звіту, що може приймати значення:

а) Standard Report – звіт для друку на принтері;

б) HTML – звіт у вигляді HTML-документа.

За допомогою інструменту "котушка"+права кнопка миші на місці входу «report type» створити список, з якого вибрати тип звіту «HTML».

Вихідними даними з блоку «New_Report» є «report out», що є саме звітом, що формується і до якого будемо додавати елементи, та «error out» – інформація про помилки, які виникли під час роботи блоку. Ці обидва виходи є обов'язковими входами для наступних блоків, що формують звіт.

Наступним блоком для формування звіту буде використано блок «Set Report Font», який дозволяє задавати параметри шрифта для звіту. Входами «Set Report Font» є, знову ж таки, «report in» до якого підключаємо вже розпочатий звіт з виходу «report out» блоку «New_Report»; та вхід «error in», куди подаються «error out» дані про помилки з попереднього блоку «New_Report».

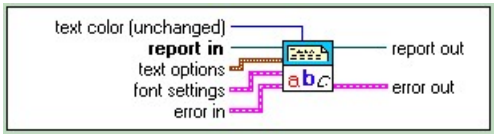


Рис. 6.2 – Блок «Set Report Font»

Для задання параметрів шрифту, треба допомогою інструменту "котушка" + права кнопка миші створити відповідно до входів блоку «Set Report Font» кластери «text options» та «font settings», які містять списки можливих значень параметрів тексту. Цей блок має що вхід «text color» що дозволяє змінювати колір тексту.

Текст, який буде додано до звіту після блоку «Set Report Font» буде мати задані параметри. При необхідності вставити текст із іншими параметрами, буде потрібно ще раз використати блок «Set Report Font».

Далі перейдемо до додавання до звіту текстових даних. Для цього використовуємо блок «Append Report Text», функція якого додавати текстовий рядок до звіту.

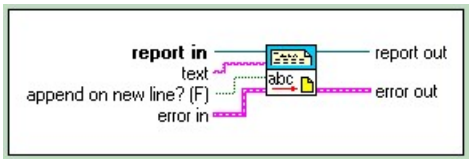


Рис. 6.3 – Блок «Append Report Text»

Текстовий рядок повинен бути типу string і може бути заданий як рядкова константа або введений з елементу керування рядкової змінної. Рядок може містити елементи керування та спеціальні символи («Carriage Return» – переведення каретки, «Line Feed» – переведення рядка, «Tab» – табуляція тощо).

У прикладі показано як скласти рядок із текстової константи та з тексту, що вводиться через елемент керування. Для цього використано блок «Concatenate String» який складає один рядок з декількох рядків або одновимірних рядкових масивів, що подаються на його входи.

Створений таким чином рядок подається на вхід «text» блоку «Append Report Text».

Додамо до звіту дані у вигляді таблиці. Для цього використаємо блок «Append Numeric Table to Report». Коннектори блоку показані на рис. 6.4.

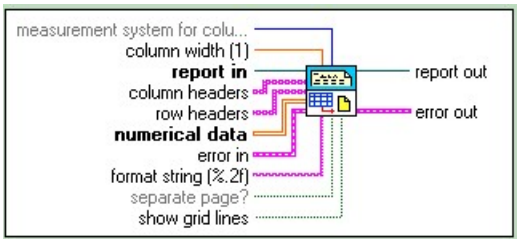


Рис. 6.4 – Блок «Append Numeric Table to Report»

Числові дані у вигляді двовимірного масиву подають на вхід «numerical data». Двовимірний масив даних сформуємо за допомогою циклу «For loop» (рис. 6.5) у якому обчислюється значення сінусу від аргументу, що заданий лічильником циклу «i» домноженим на коефіцієнт 0,1. Таким чином формуються два одновимірних масива: масив аргументів (i*0,1) та значень сінусу від аргументу. За допомогою блоку «Build Array» два одновимірних масива складаються у одни двовимірний «Output Array».

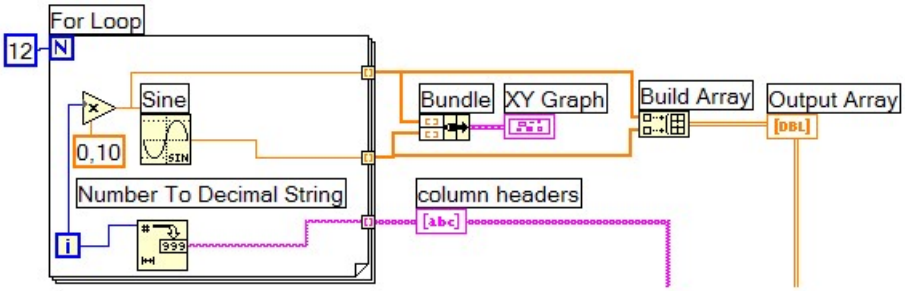


Рис. 6.5 –Формування двовимірного масиву числових даних

Також у циклі формується одновимірний масив рядкових змінних, який складається зі значень, що приймає лічильник циклу «i», тобто 0, 1, 2, 3, ... N. Цей масив рядкових змінних буде заголовком колонок таблиці і подається на вхід «column headers». Заголовок рядків таблиці формується як одновимірний масив «row headers», у прикладі заголовки будуть «Row X» та «Row Y» (рис. 6.6).

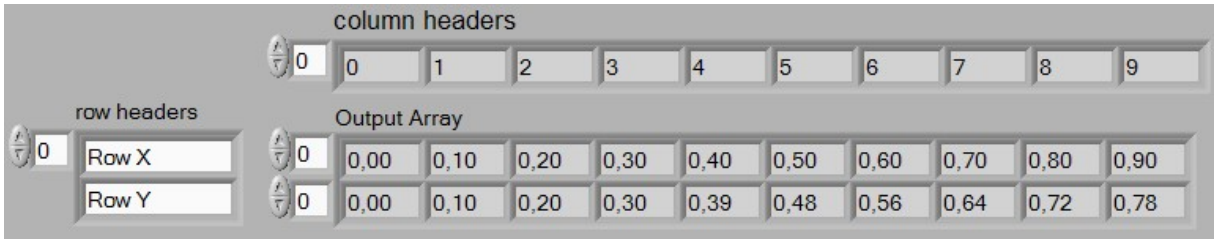


Рис. 6.6 –Формування заголовків таблиці

Також треба задати ширину колонок таблиці «column width» та одиниці вимірювання для ширини колонок «measurement system for column width», які можуть мати значення «US» для дюймів та «Metric» для сатиметрів.

Для того, щоб лінії сітки таблиці були видимі, треба подати на вхід «show grid lines» булеву константу «True».

Наступним елементом, що буде доданий до звіту, є графік «XY Graph», що побудований за значеннями, що отримані у циклі. «XY Graph» отримує значення у вигляді кластера із двох одновимірних масивів, тому кластер формуємо блоком «Functions\Cluster\Bundle».

Для додавання графіку у звіт використовуємо блок «Append Control Image to Report.vi», що знаходиться у бібліотеці «NIReport.llb» за адресою «D:\Program_Files\National_Instruments\LabVIEW6i\LabVIEW_6\vi.lib\Utility\NIReport.llb\Append Control Image to Report.vi» та додається на діаграму з пункту палітри «Functions\Select a VI...».

Графік для цього блоку вказується шляхом використання «посилання» (Reference). Спочатку треба створити посилання на графік «XY Graph», дл чого на діаграмі правою кнопкою миші викликаємо меню випадаюче меню графіку «XY Graph» і обираємо пункт «Create\Reference». На діаграмі зявляється прямокутний обєкт із назвою «XY Graph» та написом всередині «WaveformGraph» і стрілочкою, що є позначкою ярлика. Ось це створене посилання підєднуємо до входу «Ctrl Reference» блоку «Append Control Image to Report.vi».

Додамо дату та час створення до звіту. Це можна зробити блоком «Append Report Text» (дивись опис вище), при цьому рядок, що додається до звіту буде містити дату та час. Дату та час отримуємо з блоку «Get Date/Time String» (рис. 6.7).

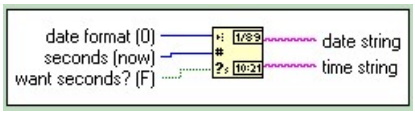


Рис. 6.7 – Блок «Get Date/Time String»

Опціями блоку є формат дати «date format», що може приймати значення:

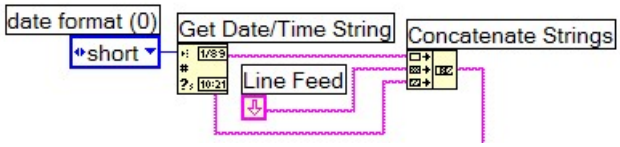
short(0) 12.05.2014;

long(1) 12 травня 2014 р.;

abbreviated(2) 12 Тра 2014 р.

Виходами блоку є рядки дати та часу «date string», «time string».

Для додавання до звіту дата та час складаємо до одного рядка блоком «Concatenate String», додавши між ними константу «Functions\String\Line Feed» – переведення рядка.



Нарешті, сформований звіт слід записати до файлу HTML за вказаним файловим шляхом. Це робиться блоком «Save Report to File», вхідними параметрами якого є, як завжди, звіт «report in» та шлях файлу звіту «report file path».

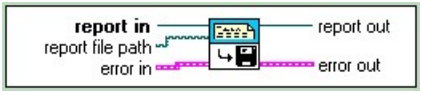


Рис. 6.8 – Блок «Save Report to File»

Шлях є спеціальним типом змінної у LabVIEW, операції над яким можна проводити за допомогою блоків з розділу «Functions\File I/O». Можна кожного разу задавати шлях та ім'я для файлу вручну з елементу керування передньої панелі, але цей спосіб не дуже зручний. У прикладі показано як можна автоматизувати формування шляху та імені файлу. Для цього шлях формується, за умовчуванням, з імені системної папки для тимчасових файлів «Functions\File I/O\File Constant\Temporary Directory» та імені файла, заданого рядковою константою, наприклад «report.html». З цих двох елементів збирається шлях за допомогою блоку «Build Path», де у якості «базового шлях» («base path») буде підєднано «Temporary Directory», а до «ім'я або відносний шлях» («name or relative path») підєднано текстову константу «report.html».

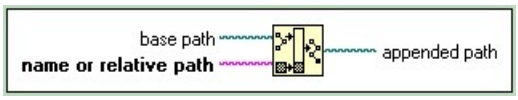


Рис. 6.9 – Блок «Build Path»

На виході блоку «appended path» буде сформований шлях та ім'я за умовчуванням, наприклад, «C:\TEMP\report.html».

Для користувача варто залишити можливість самостійно задавати шлях та ім'я файлу звіту, якщо вибір за умовчуванням не влаштовує. Для цього на передній панелі розташуємо елемент управління «Controls\String&Path\File Path Control» для введення шляху та імені файлу. Для того, щоб програма мала можливість обрати між шляхом за умовчуванням та шляхом користувача, введемо просту перевірку «Functions\Comparison\Empty String/Path?», яка визначає, чи не є рядок шляху порожнім, із булевим виходом.

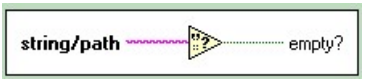


Рис. 6.10 – Блок «Empty String/Path?»

Результат перевірки подаємо на блок «Select». В залежності від результату перевірки, блок «Select» вида на вихід шлях користувача (якщо він не порожній) або шлях за умовчуванням.

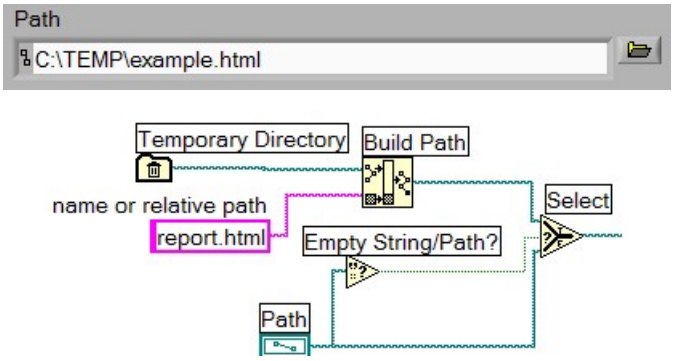


Рис. 6.11 – Формування шляху та імені файлу

Для коректного закінчення формування звіту слід використати блок «Dispose Report» функція якого полягає у закритті звіту та видаленні його з памяті, що дозволяє економити місце у памяті.

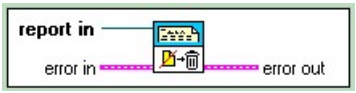
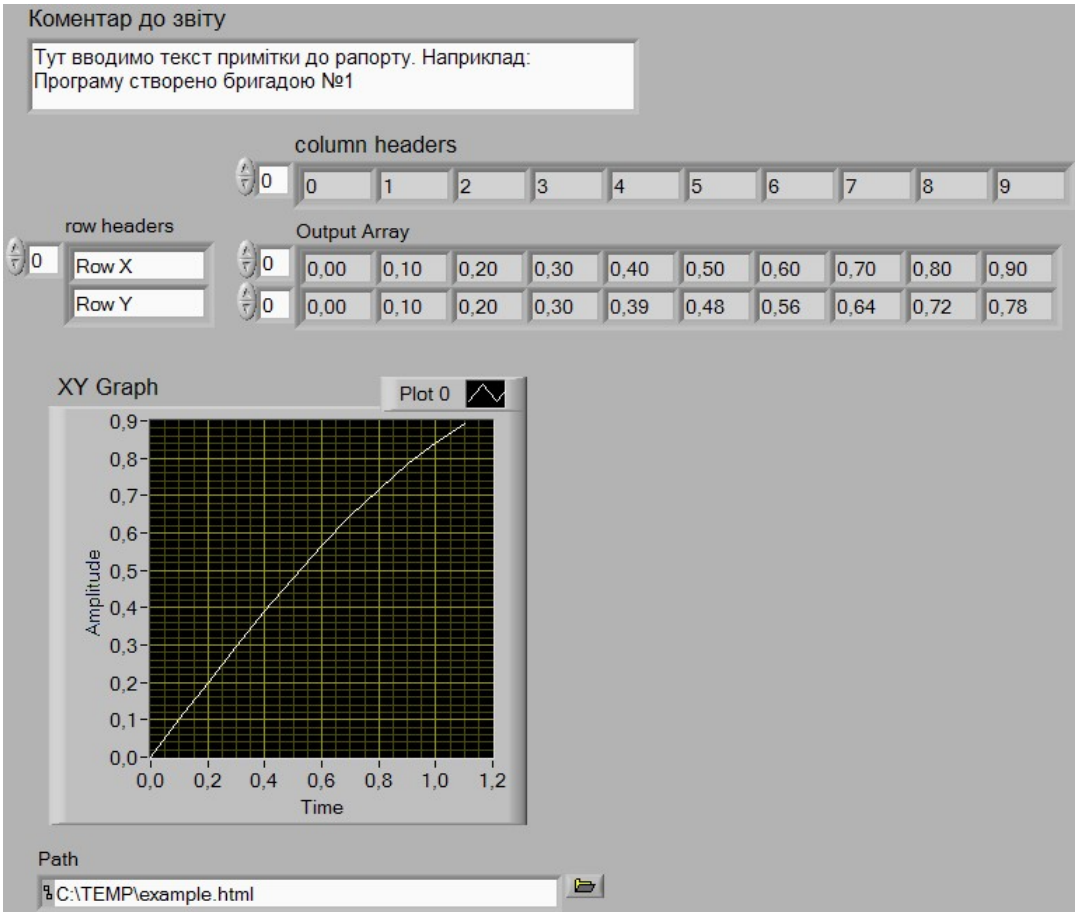


Рис. 6.12 – Блок «Dispose Report»

В результаті маємо отримати програму, що показана на рис. 6.13 та результат роботи програми – звіт рис. 6.14.

Front Panel


Коментар до звіту
Тут вводимо текст примітки до рапорту. Наприклад:
Програму створено бригадою №1
column headers
0 1 2 3 4 5 6 7 8 9
row headers
Row X
Row Y
Output Array
0,00 0,10 0,20 0,30 0,40 0,50 0,60 0,70 0,80 0,90
0,00 0,10 0,20 0,30 0,39 0,48 0,56 0,64 0,72 0,78
XY Graph
Plot 0
Amplitude
Time
0,0 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8 0,9
0,0 0,2 0,4 0,6 0,8 1,0 1,2
Path
C:\TEMP\example.html

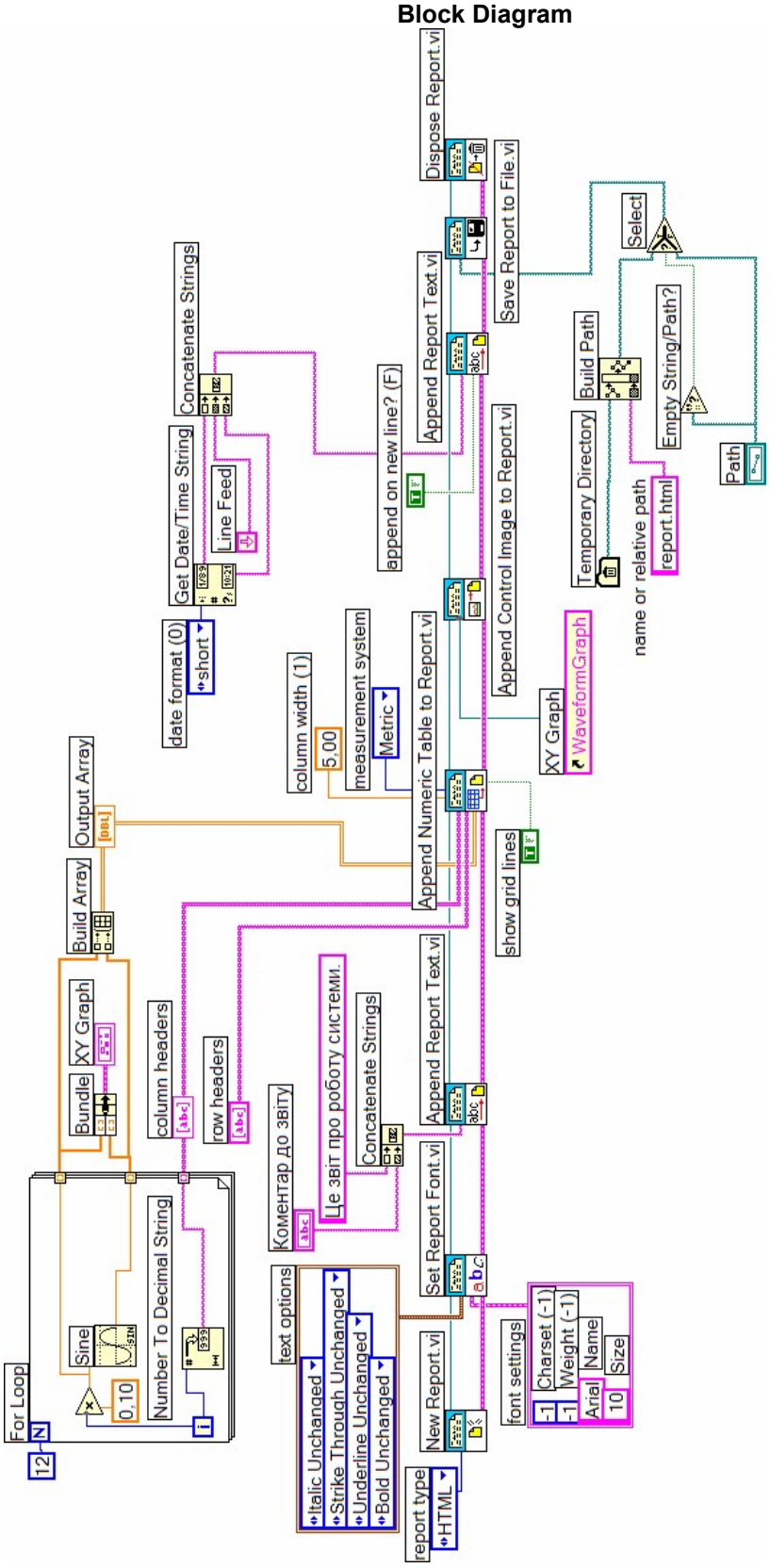


Рис. 6.13 – Програма формування звіту

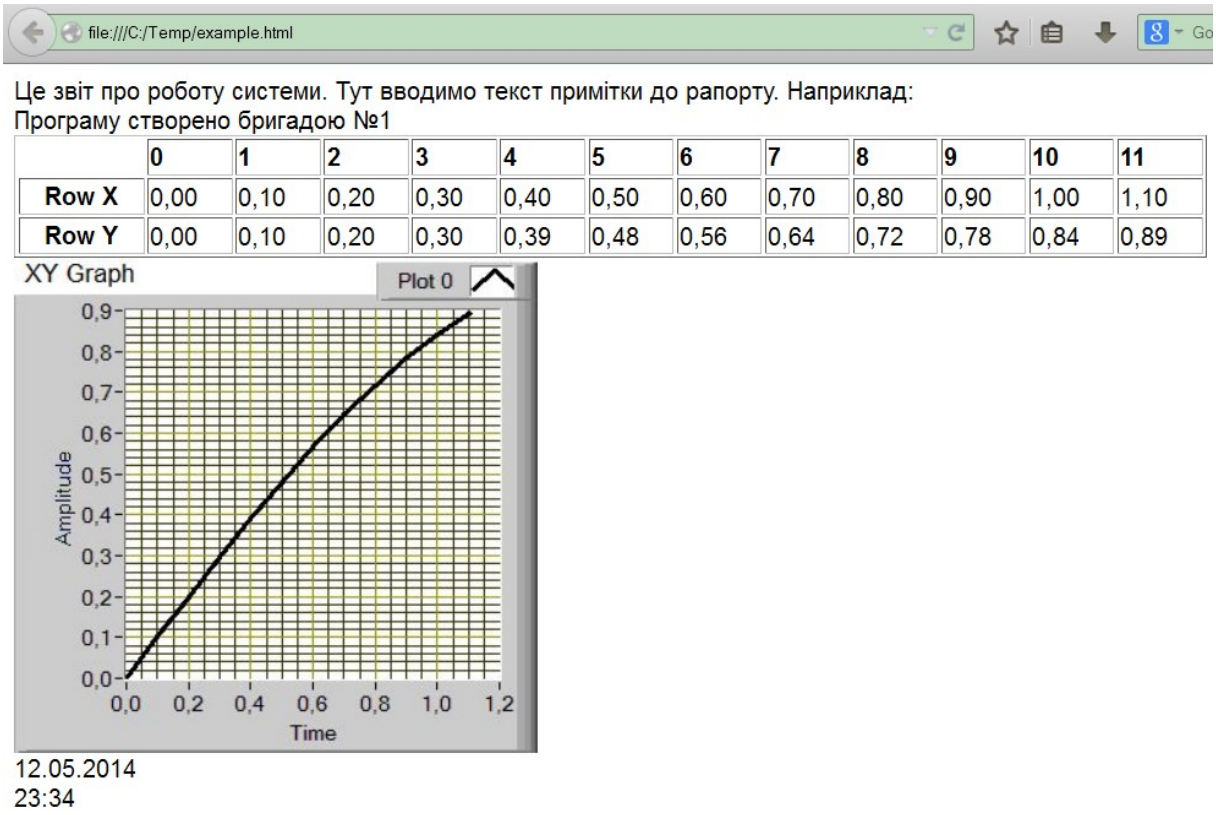


Рис. 6.14 –Звіт, сформований програмою

Завдання:

Сформувати звіт з роботи програми. Звіт має містити:

1) Назву звіту та номер бригади

2) Таблицю вхідних та вихідних даних із заголовками

3) Графік зміни вихідного параметру у часі

4) Дату та час створення звіту

Контрольні питання:

1. Як додати до звіту зображення графіку?

2. Як вставити до тексту звіту параметри, які можуть змінюватися?

3. Як додати до звіту таблицю?

5. СКІНЧЕННИЙ АВТОМАТ

Мета роботи: *Створення та дослідження програми збору даних із використанням скінченного автомату в середовищі LabVIEW.*

Скінченний автомат (англ. *finite-state machine*, машина зі скінченною кількістю станів) — особливий різновид автомата — абстракції, що використовується для опису шляху зміни стану об'єкта в залежності від поточного стану та інформації отриманої ззовні. Його особливістю є скінченність множини станів автомату. Поняття скінченного автомата було запропоновано як математичну модель технічних приладів дискретної дії, оскільки будь-який такий пристрій (в силу скінченності своїх розмірів) може мати тільки скінченну кількість станів.

На рис. 5.1 наведено граф станів скінченного автомату.

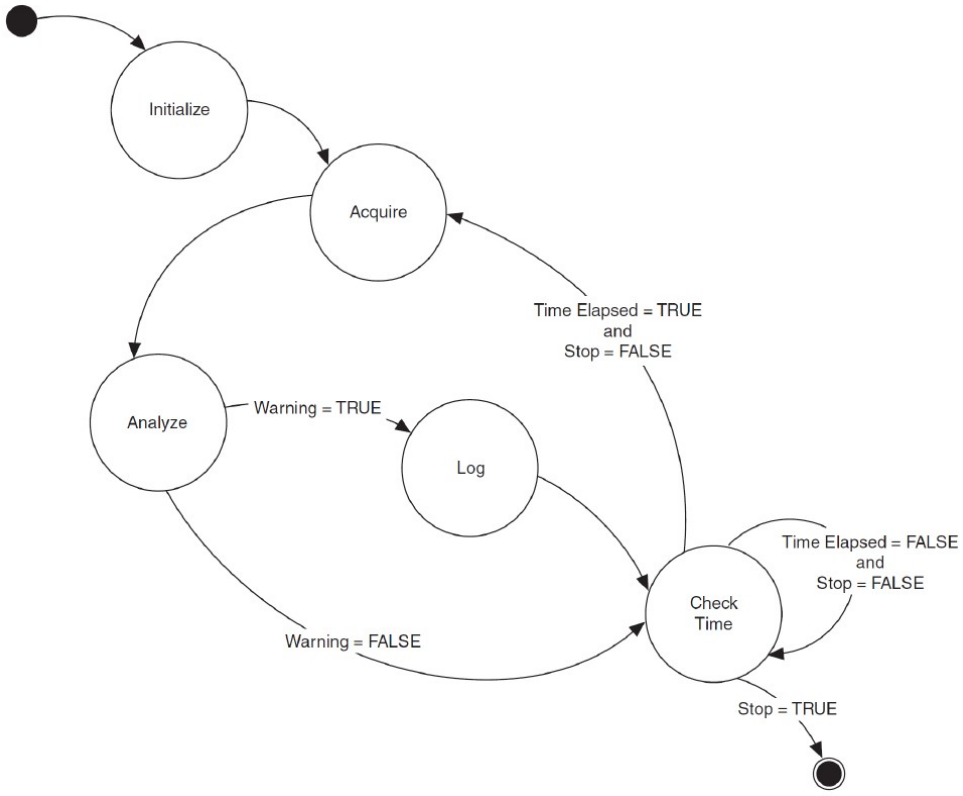


Рис. 5.1 Граф станів скінченного автомату

Граф складається із станів та переходів між станами, які визначають, який стан буде наступним. В таблиці наведено опис станів скінченного автомату наведеного на рис. 5.1.

Стан	Опис	Наступний стан
Acquire (одержання)	Обнулити таймер та одержати дані з датчика температури	“Analyze”
Analyze (аналіз)	Зчитати команди з панелі керування і визначити рівень сигналу попередження	“Log” якщо є попередження “Check Time” якщо немає попередження
Log (реєстрація)	Записати дані у файл	“Check Time”
Check Time (перевірка часу)	Перевірити, чи сплив час (більше або дорівнює 0,5 секунди)	“Acquire” якщо час сплив “Check Time” якщо час не сплив

Програма отримує температуру кожні 0,5 секунди, аналізує кожну температуру, щоб визначити, занадто висока чи занадто низька температура, і попереджає користувача, якщо є небезпека теплового удару або замерзання. Програма реєструє дані, якщо з’являється попередження. Якщо користувач не натиснув кнопку «Stop», весь процес повторюється.

Скінченний автомат дозволяє розширення кількості станів, оскільки інші процеси можуть бути додані в майбутньому.

Розглянемо приклад побудови такої програми.

В якості джерела даних (значень температури) візьмемо генератор випадкових чисел Random Number (0-1). Оскільки він генерує числа в діапазоні від 0 до 1, домножимо згенероване число на 40 щоб отримати

діапазон температур від 0 до 40 градусів.

Реєстрація даних буде відбуватися у разі виникнення попередження шляхом запису у текстовий файл значення температури та тексту попередження.

Програма буде виконуватися циклічно, тож буде використано цикл While.

Кожному стану буде відповідати окреме вікно структури Case.

Почнемо із створення передньої панелі програми (рис.5.2)

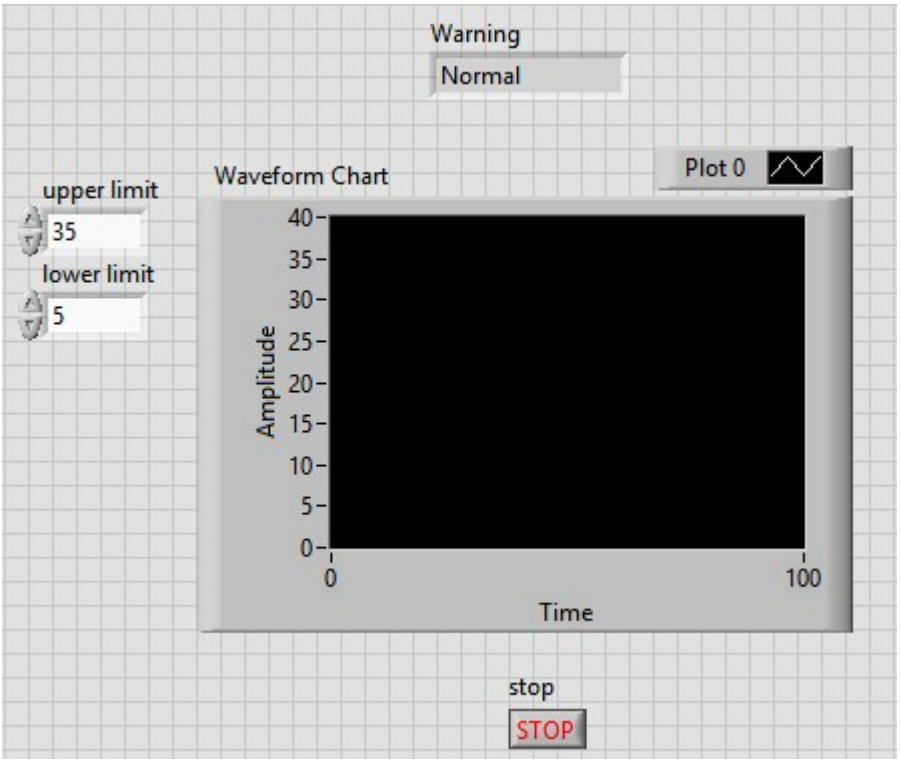
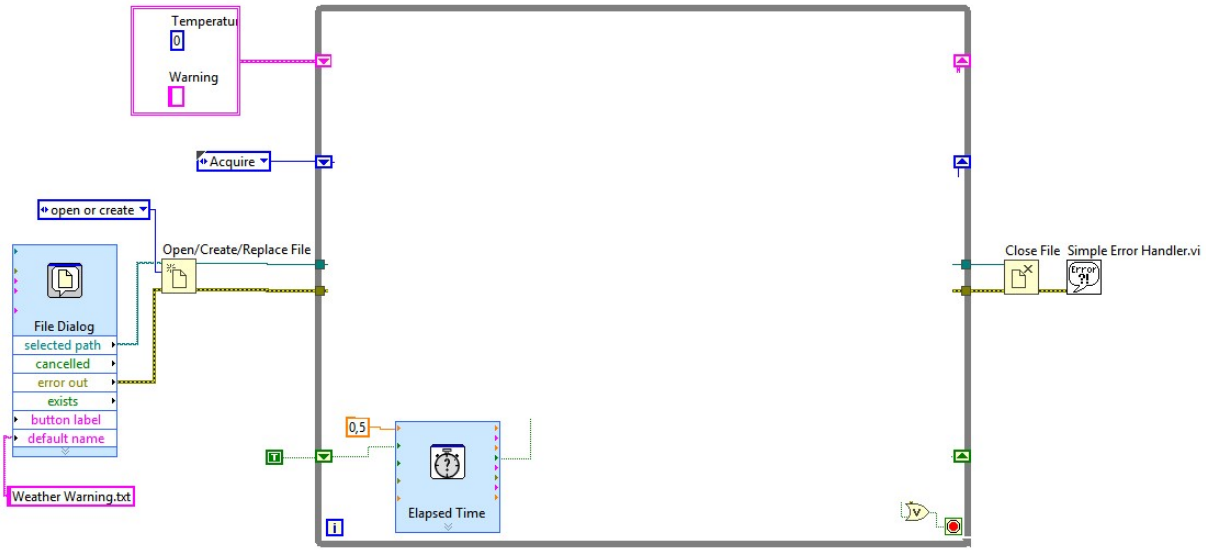


Рис. 5.2

Тут розташовані задатчики верхньої та нижньої межі припустимої температури, вікно Waveform Chart для відображення графіку температури, вікно для виведення текстових попереджень Warning та кнопка зупинки програми Stop.

Розглянемо елементи що знаходяться поза циклом While.



Дані про температуру та попередження будуть передаватися через кластер із елементами Temperature (numeric constant) та Warning (text constant). Кластер підключено до зсувогого регістру на циклі.

Для задання імені та шляху до файлу реєстрації використано блок «File Dialog» із указанням ім’я файлу за умовчанням як текстової константи. Блок «Open/Create/Replace File» відкриває або створює файл за заданим шляхом. Слід задати параметр “operation” створивши для нього control та обрати варіант «open or create» як зручний для даної програми. Термінали «Selected path» та «Error out» проходять скрізь усі блоки роботи із файлами. Для завершення роботи із файлом його слід закрити блоком «Close file» та обробити помилки роботи із файлом блоком «Simple Error Handler».

Для контролю часу виконання ітерації використаємо блок «Elapsed Time» в якому слід підключити до терміналу Time Target константу 0,5 (мінімальний період опитування значень температури в секундах), до терміналу Reset константу True через зсувів регістр (початкове скидання таймера при запуску програми). Із вихідних терміналів буде використано Time has elapsed який набуває значення True коли спливе час 0,5 с.

Ліворуч від циклу створимо Enum Constant (знаходиться в меню Nimeric). Правою кнопкою миші на Enum Constant вибиремо Edit Items і відкриємо вікно Enum Constant Properties. На закладці Edit Items додамо елементи із назвами станів, як показано на рис. 5.3.

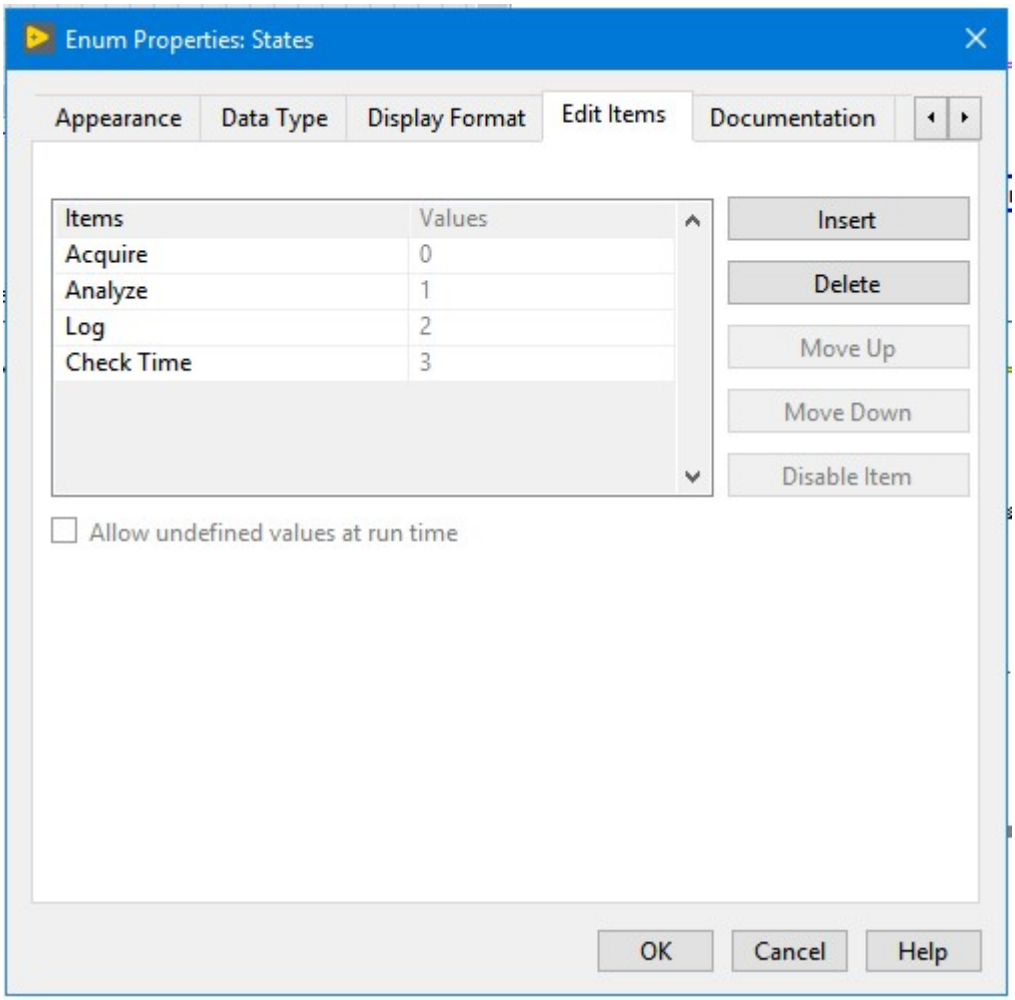


Рис. 5.3.

Правою кнопкою миші в контекстному меню константи на діаграмі оберемо пункт Make Type Def. Це дозволить створити визначення нового типу який можна буде використовувати в проектах. Правою кнопкою миші в контекстному меню константи enum constant оберемо Open Type Def. У вікно що відкрилося перейменуємо enum control у «States». Застосуємо

зміни у визначенні типу (File/Apply Changes). Отримане визначення типу можна зберегти у файлі Weather Station States. Закриємо вікно Type Def.

Цю enum constant підключемо до зсувогово регістру на циклі.

В середину цикла вставимо структуру Case. Підключимо до селектору Case вихід з зсувогово регістру від enum constant (рис. 5.4).

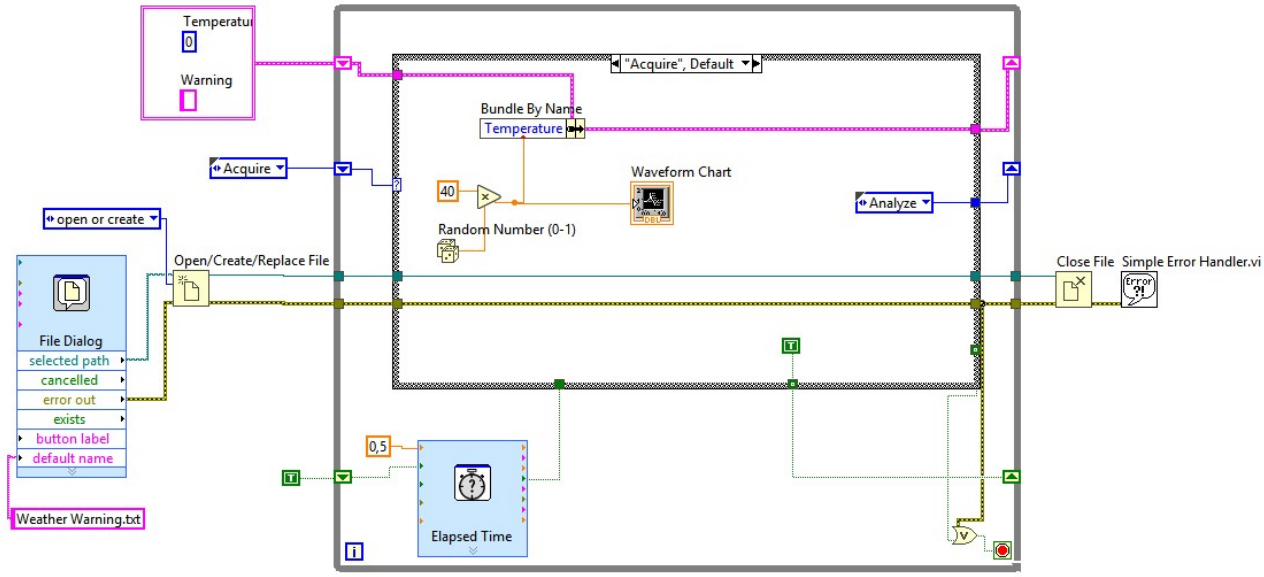


Рис. 5.4

Для того щоб збільшити кількість вікон Case для відповідності кількості станів у enum constant правою кнопкою миші на Case Structure в контекстному меню оберемо «Add Case For Every Value».

Далі заповнюємо вікна Case Structure.

Перше вікно відповідає стану «Acquire». Тут генерується «виміряне» значення температури, виводиться на графік та записується у поле кластера «Temperature». Константу True підключити до виходу до зсувогово регістру скидання таймера. Наступний стан визначається встановленням копії enum constant із обраним значенням «Analyze».

Друге вікно відповідає стану «Analyze» (рис. 5.5).

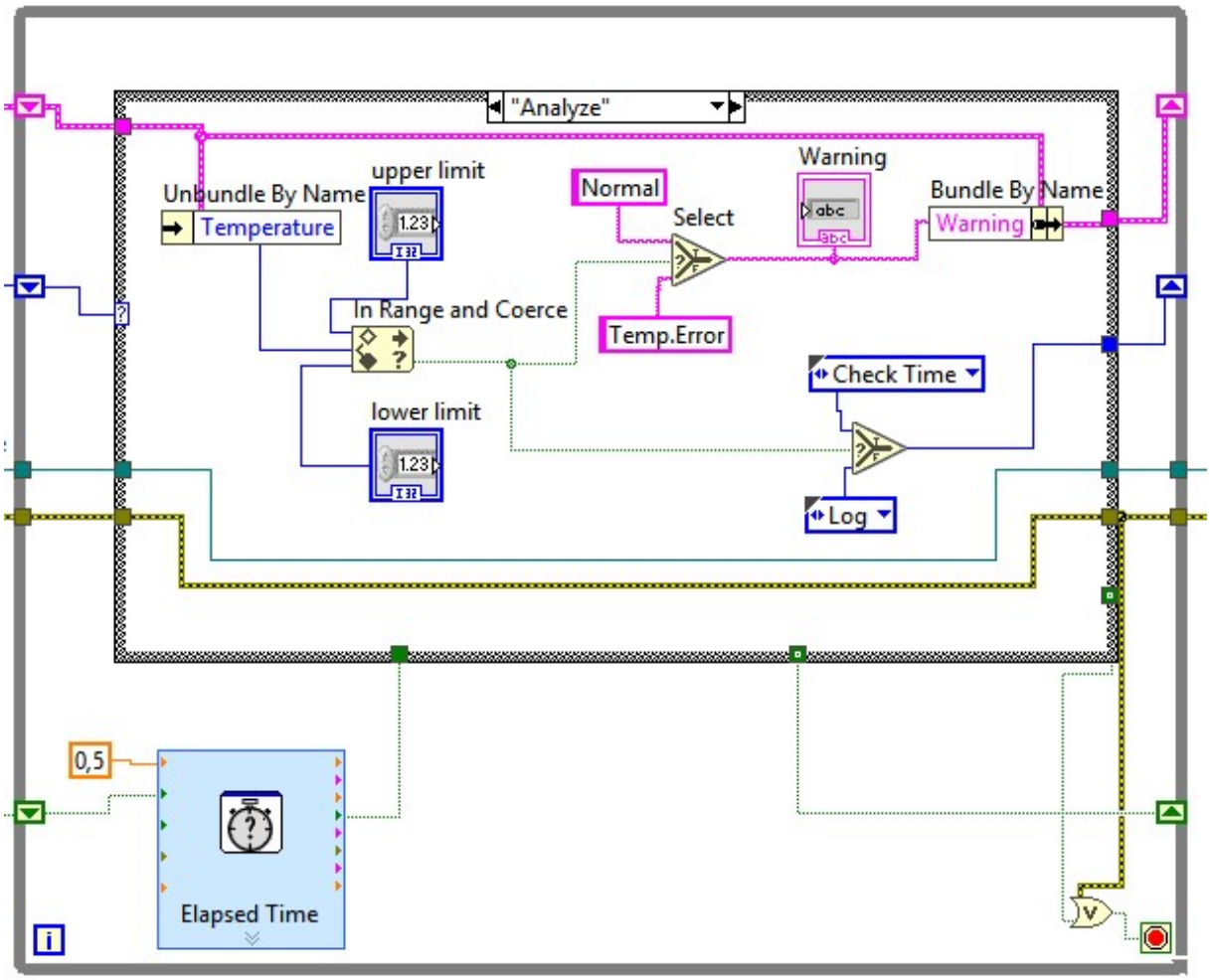


Рис. 5.5

Тут отримане значення температури порівнюється із заданими межами блоком «In Range and Coerce». Якщо значення температури знаходиться у припустимих межах на виході блоку логічна зміння набуває значення True, або при виході за межі – False. Ця змінна використовується як перемикач при виборі тексту попередження блоком Select (“Normal” чи “Temp.Error”). Та сама змінна перемикає блок Select для вибору наступного стану: «Check Time» якщо True (температура знаходиться у припустимих межах) або «Log» (температура поза припустимими межами і треба це зареєструвати).

Третє вікно відповідає стану «Log» (рис. 5.6).

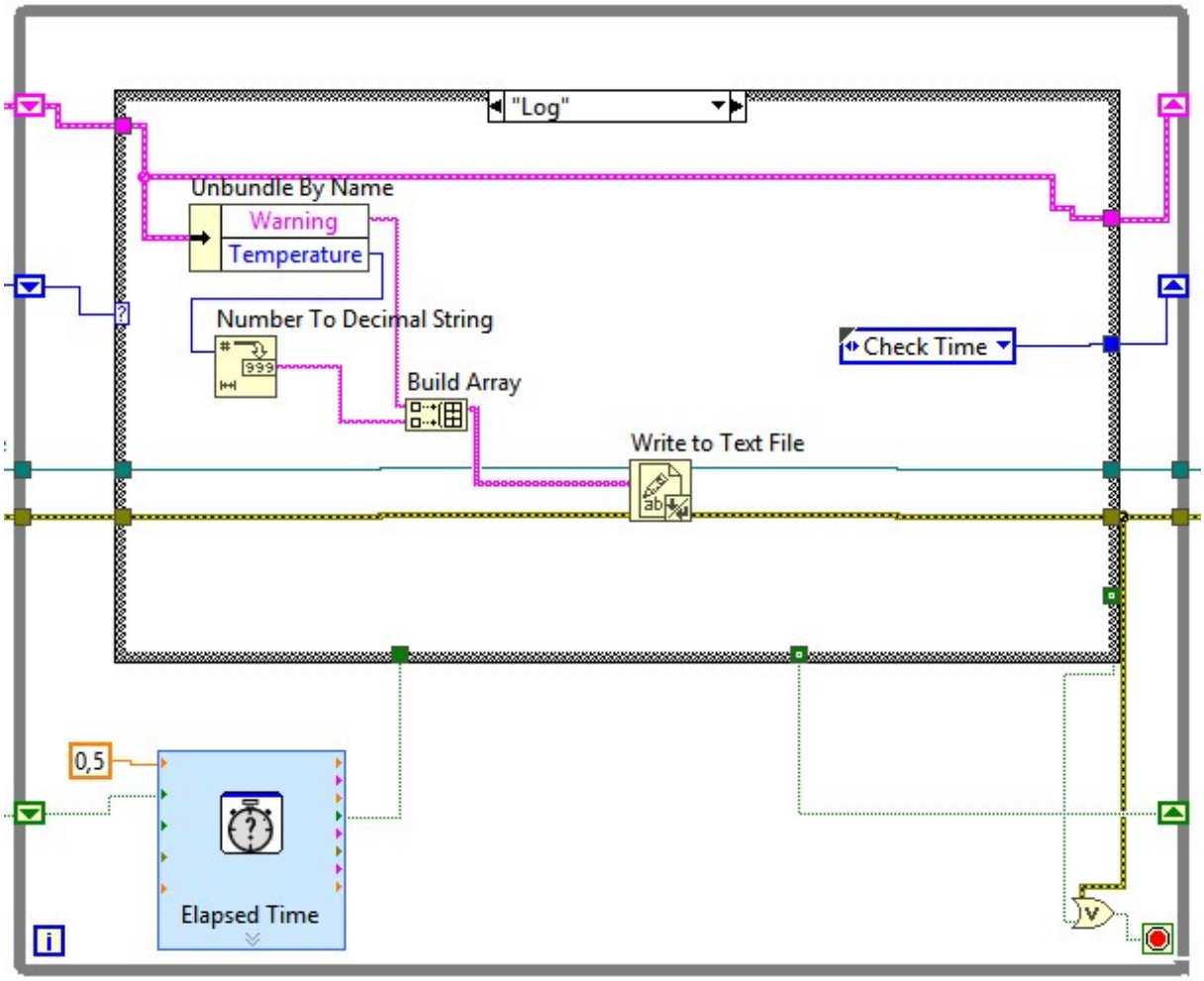


Рис. 5.6

Тут текст попередження та значення температури (перетворене у текст) збирають у масив та передають у блок «Write to text file». Наступний стан задається константою Check Time.

Четверте вікно відповідає стану «Check Time» (рис. 5.7).

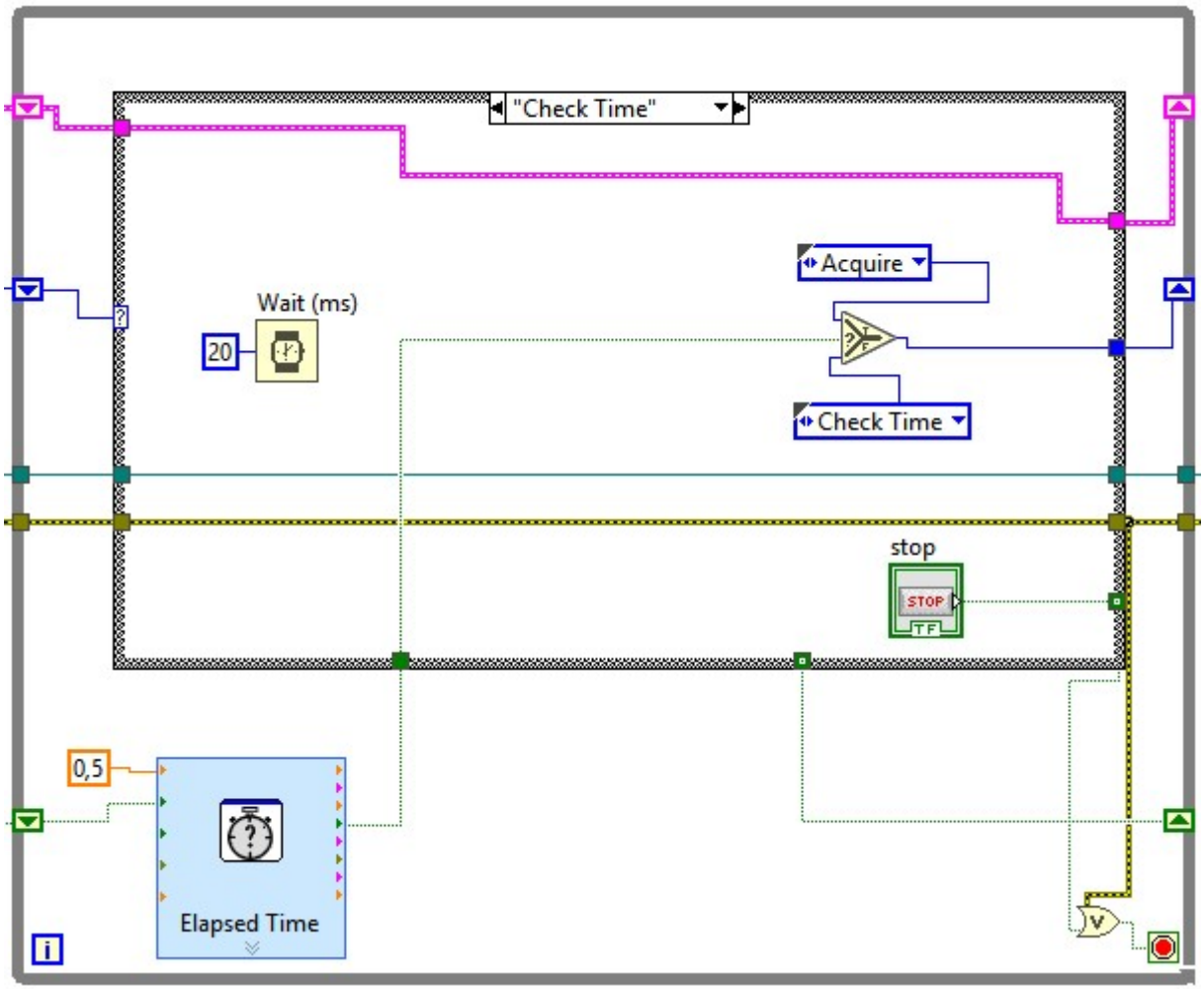


Рис. 5.7

Тут опитується таймер «Elapsed Time»: чи пройшло часу більше 0,5 с? Якщо так, вихід блоку «Elapsed Time» на терміналі «Time has elapsed» набуває значення True і перемикає селектор із наступним станом у «Aquire». Якщо час 0,5 с ще не минув, повторно викликається стан «Check Time». При цьому додатково витримується час 20 мс блоком Wait.

В стані «Check Time» опитується кнопка Stop, яку міг натиснути оператор. В цьому випадку значення True з кнопки іде на логічний блок OR і, далі, на термінал умови зупинки циклу у стані «Stop if True». Іншою умовою зупинки циклу є помилка в роботі із файлом.

На вікні стану «Check Time» лишаються не зарисованими тунелі

логічних зміних у правому нижньому куті Case. Слід правою кнопкою миші викликати на цих тунелях контекстне меню і обрати «Use Default If Unwired». В такому разі вони набудуть значення False якщо не підключені.

Під час роботи програми генерується файл Weather Warning.txt із записами типу:

```
Temp.Error
4
Temp.Error
36
Temp.Error
35
Temp.Error
38
Temp.Error
0
Temp.Error
39
Temp.Error
40
```

Використання структури скінченного автомату для створення програм дозволяє проаналізувати усі можливі стани системи (програми) та логіку переходів між станами. В разі необхідності можна легко додати до вже існуючої програми нові стани чи переходи між станами (рис. 5.8).

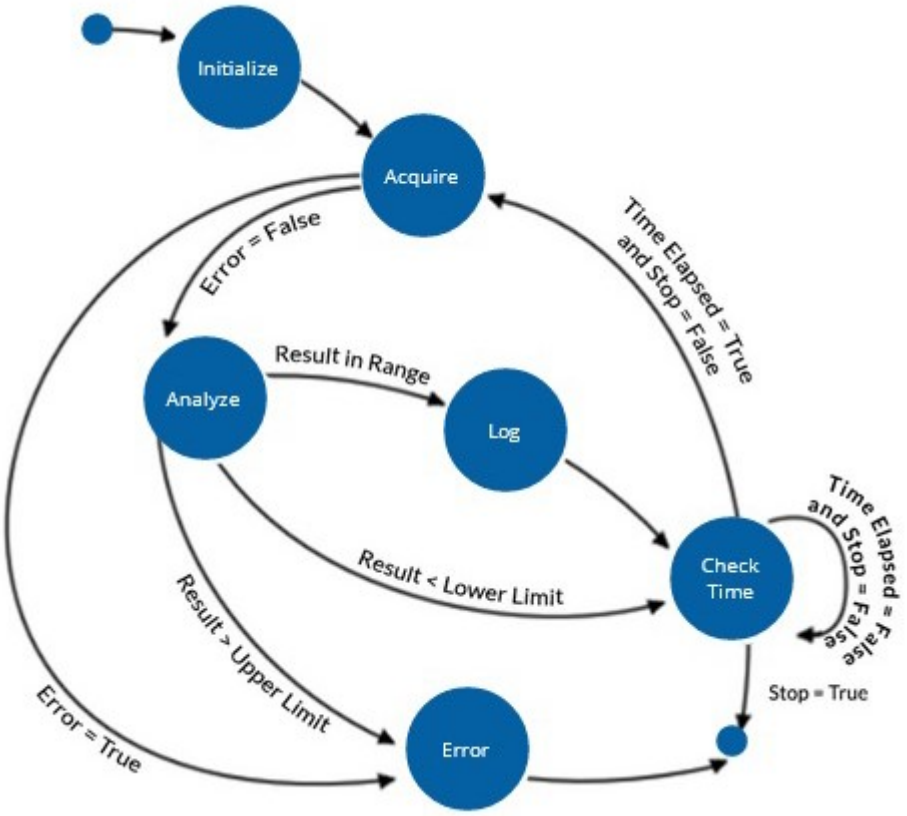


Рис. 5.8.

Завдання:

1. Зібрати описану програму із скінченним автоматом і перевірити її роботу;

2. Змініть програму, додавши ще один стан чи перерехід між станами. Наприклад, як наведено на рис. 5.8.

Контрольні питання:

1. Що таке скінченний автомат?

2. Що таке стан та перерехід скінченному автоматі?

3. Як додати стан до структури Case?

6. ОБРОБКА ШТРИХОВИХ КОДІВ

Мета роботи: *Створення та дослідження програми розпізнання штрихового коду в середовищі LabVIEW.*

В інформаційній системі здійснюється автоматизована обробка інформації із використанням технічного забезпечення, яке включає в себе ЕОМ і засоби зв'язку з джерелами інформації.

Широке проникнення логістики в сферу виробництва та економіки в істотному ступені зобов'язане комп'ютеризації управління матеріальними потоками. У виконавчих інформаційних системах здійснюється оперативне управління матеріальними потоками. Для цих систем особливо важливо фіксувати і обробляти інформацію в темпі проходження матеріального потоку, тобто за умови застосування сучасної техніки і технології збору, обробки і передачі інформації в режимі реального масштабу часу.

Через кожну ланку логістичного ланцюга проходить велика кількість одиниць товарів. При цьому всередині кожної ланки товари неодноразово переміщаються по місцях зберігання і обробки. Для того щоб мати можливість ефективно управляти цією динамічною логістичної системою, необхідно в будь-який момент часу мати інформацію в детальному асортименті про що входять і виходять з неї матеріальних потоках, а також про матеріальні потоки, що циркулюють всередині її. Дана проблема вирішується шляхом використання мікропроцесорної техніки, здатної ідентифікувати (впізнати) окрему вантажну одиницю завдяки скануванню (зчитуванню) різноманітних штрихових кодів. Це обладнання дозволяє отримувати інформацію про логістичної операції в момент і в місці її скоєння - на складах промислових підприємств, оптових баз, магазинів, на

транспорті. Отримана інформація обробляється в режимі реального масштабу часу, що дозволяє керуючої системі реагувати на неї в оптимальні терміни.

Автоматизований збір інформації заснований на використанні штрихових кодів різних видів, кожен з яких має свої технологічні переваги.

У сфері обігу широке застосування отримав штриховий код EAN-13, який часто можна зустріти на товари масового споживання.

Оскільки в даній роботі маємо справу із зображеннями, можемо використовувати заздалегіть завантажені зображення (відскановані, сфотографовані чи отримані в інтернеті) або отримувати зображення в процесі роботи з зовнішнього пристрою, наприклад, веб-камери комп’ютера.

Для роботи із розрізнанням зображення в LabVIEW необхідно встановити модуль NI Vision Acquisition Software (NI-IMAQ).

В програмі NI MAX (NI Measurement and automation explorer) слід перевірити чи підключена веб-камера і розпізнається серед Devices and interfaces (рис. 6.1). Можливо, доведеться налаштувати фокусну відстань на камері для збільшення якості дрібного зображення.

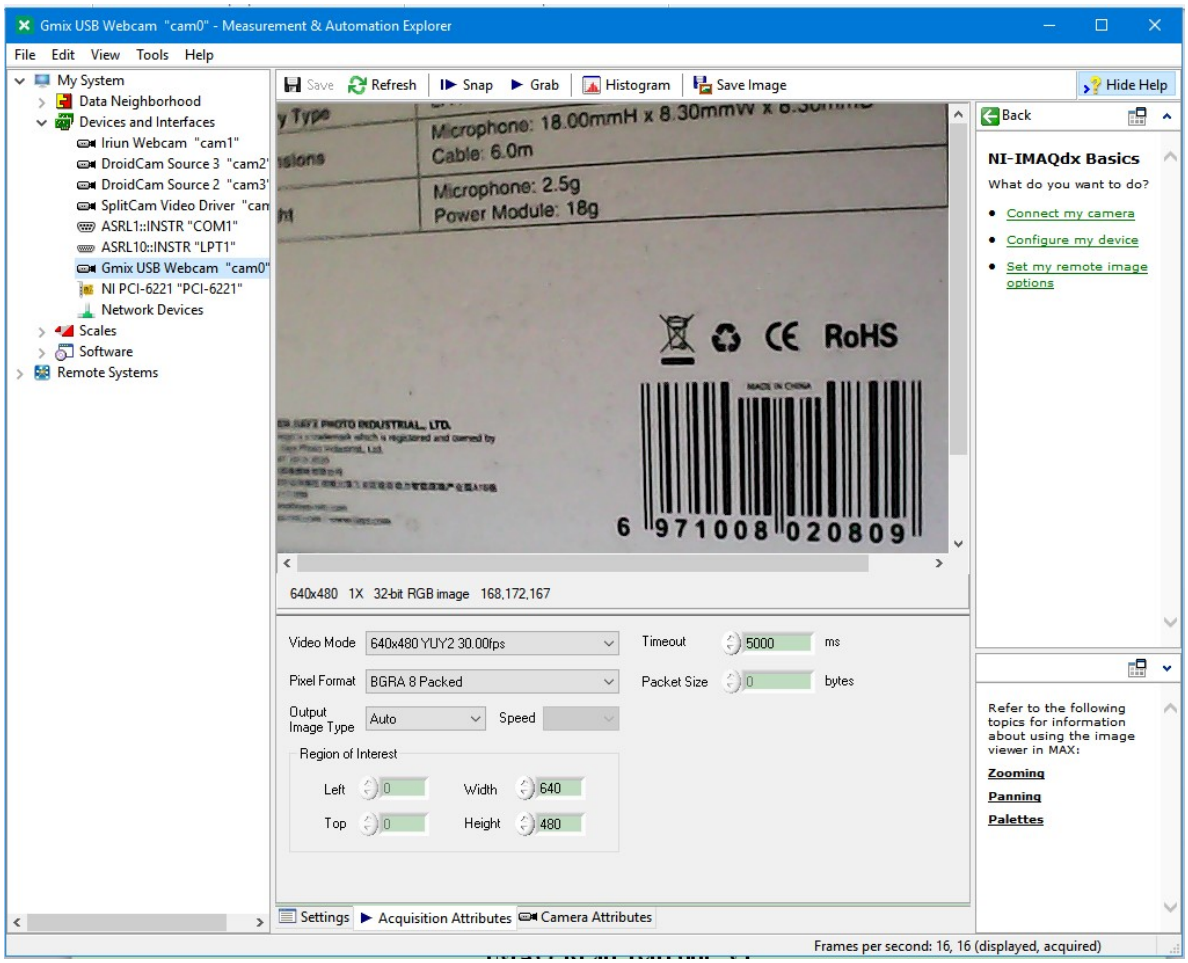


Рис. 6.1.

Запам’ятайте як називається камера, яку плануєте використовувати – це ім’я буде ідентифікатором у програмі під час звернення до неї. В даному випадку камера «Gmix USB Webcam» має ідентифікатор “cam0”.

Послідовність роботи із камерою аналогічна роботі із будь-яким зовнішнім пристроєм у LabVIEW: відкриття/ініціалізація, конфігурація, зчитування/запис, закриття.

Для відкриття камери для роботи використовуємо блок «IMAQdx Open Camera VI».

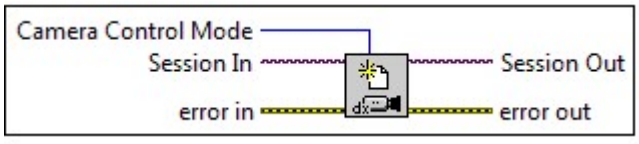


Рис. 6.2

До терміналу **Session In** – підключаємо посилання (reference) на камеру, яку плануємо використовувати. **Session Out** та **error out** використовуємо для підключення до наступного блоку конфігурування і захоплення зображення «IMAQdx Configure Grab VI» (рис. 6.3). Цей блок починає захоплювати зображення з камери у буфер.

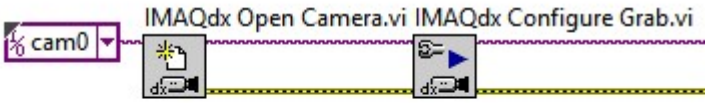


Рис. 6.3

З буферу зображення отримуємо за допомогою наступного блоку «IMAQdx Grab2 VI». Цей блок зчитує останнє отримане зображення і передає його на Image Out.

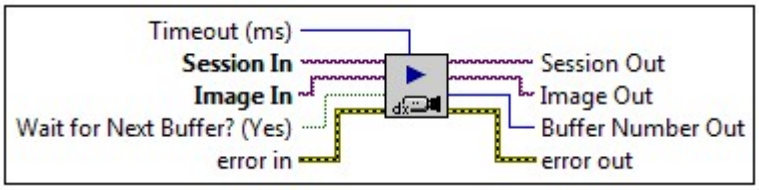


Рис. 6.4

Session In є таким самим посиланням на поточну камеру з попереднього блоку. Image In є посиланням на зображення, яке отримує зчитані з камери дані. Це посилання слід попередньо створити за допомогою «IMAQ Create VI» (рис. 6.5).

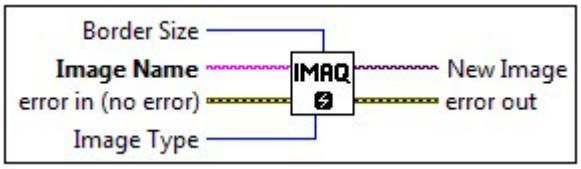


Рис. 6.5

В даному випадку з параметрів достатньо задати лише ім’я зображення **Image Name.** Тип зображення **Image Type** може бути заданий з наступного переліку (рис. 6.6).

Grayscale (U8) (0)	8 bits per pixel (unsigned, standard monochrome)
Grayscale (16) (1)	16 bits per pixel (signed)
Grayscale (SGL) (2)	32 bits per pixel (floating point)
Complex (CSG) (3)	2 × 32 bits per pixel (floating point)
RGB (U32) (4)	32 bits per pixel (red, green, blue, alpha)
HSL (U32) (5)	32 bits per pixel (hue, saturation, luminance, alpha)
RGB (U64) (6)	64 bits per pixel (red, green, blue, alpha)
Grayscale (U16) (7)	16 bits per pixel (unsigned, standard monochrome)

Рис. 6.6

Для завершення роботи із камерою і закриття сесії використовуємо блок «IMAQdx Close Camera VI» (рис. 6.7).

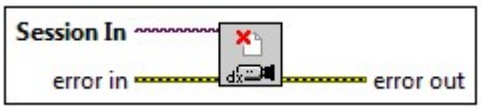


Рис. 6.7

Безпосередньо для розпізнання штрих-коду за зображенням використовуємо блок «IMAQ Read Barcode 2 VI» який розпізнає найбільш поширені одновимірні штрих-коди, такі як Codabar, Code 39, Code 93, Code 128, EAN 8, EAN 13, Interleaved 2 of 5, MSI, UPCA, Pharmacode, RSS Limited (рис. 6.8).

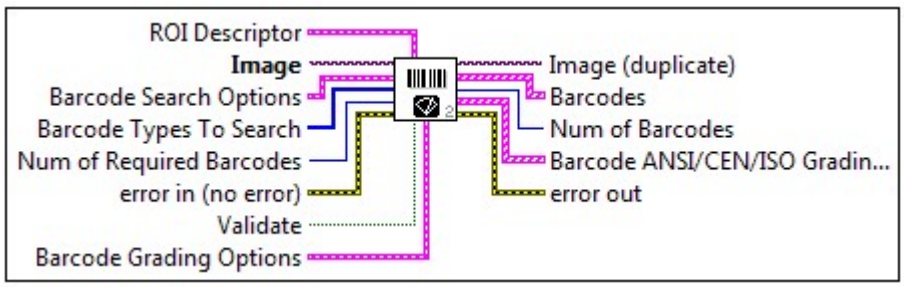


Рис. 6.8

Для роботи блоку підєднуємо зчитане зображення **Image** та задаємо

тип штрих-коду який слід шукати **Barcode Types To Search** з переліку (рис. 6.9).

Codabar (1)	Reads a Codabar barcode.
Code 39 (2)	Reads a Code 39 barcode.
Code 93 (3)	Reads a Code 93 barcode.
Code 128 (4)	Reads a Code 128 barcode.
EAN 8 (5)	Reads an EAN 8 barcode.
EAN 13 (6)	Reads an EAN 13 barcode.
Interleaved 2 of 5 (7)	Reads an Interleaved 2 or 5 barcode
MSI (8)	Reads an MSI barcode.
UPCA (9)	Reads a UPCA barcode.
Pharmacode (10)	Reads a Pharmacode barcode.
RSS Limited (11)	Reads an RSS Limited (GS1 DataBar Limited) barcode.

Рис. 6.9

В нашому випадку обираємо EAN 13. Результат розпізнання виводимо в кластер **Barcodes** який складається з:

- **Data** – декодовані дані штрих-коду.
- Special Char 1 містить інформацію про штрих-код, яка залежить від типу штрих-коду. Для Codabar це перший символ. Для Code 128 –FNC символ. Для EAN 8 та EAN 13 — перший символ країни. Для усіх інших цей параметр дорівнює 0.
- Special Char 2 містить інформацію про штрих-код, яка залежить від типу штрих-коду. Для Codabar це символ зупинки. Для EAN 8 та EAN 13 — другий символ країни. Для UPCA — номер системи. Для усіх інших цей параметр дорівнює 0.
- Checksum(s) – це інформація для виправлення помилок за допомогою якої можна перевірити правильність декодування.
- Barcode Type – тип зчитаного штрих-коду.

- Confidence Level – рівень певності зчитування, має діапазон від 0 до 1000, де 1000 є найкращим результатом. Оцінює відхилення ширини штрихів та пробілів у коді. Значення менше 800 означає непевність декодування.
- Bounding Box – масив пікселей, який містить штрих-код.

Для більшої зручності роботи з програмою, створимо безкінечний цикл, в якому буде відбуватися зчитування і розпізнання штрих-коду. Візуальний контроль зображення буде відбуватися за допомогою індикатора Image (Controls/Vision/Image Display). Зупинка роботи програми – кнопкою Stop (рис. 6.10).

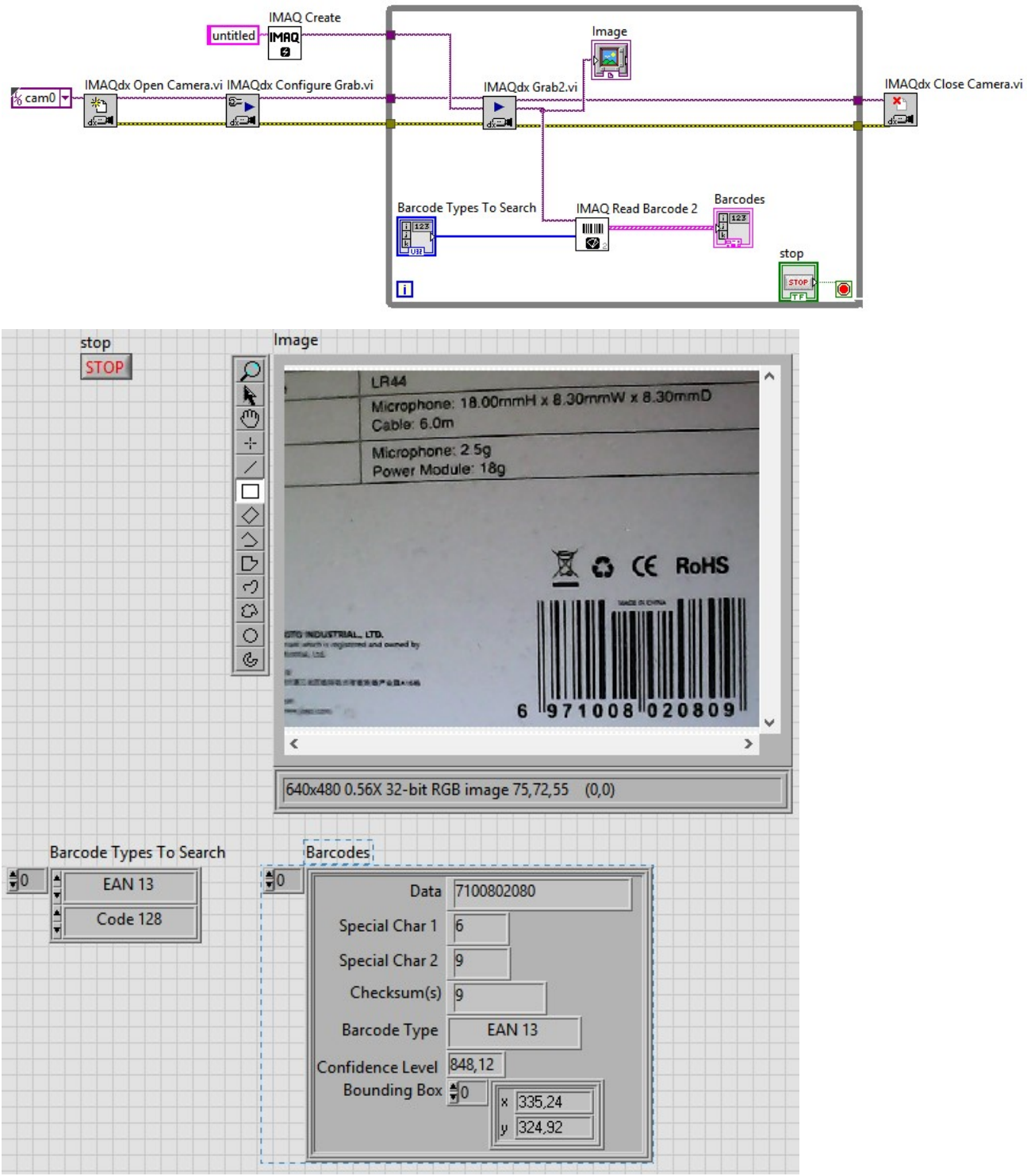

IMAQ Create
untitled
Image
cam0
IMAQdx Open Camera.vi
IMAQdx Configure Grab.vi
IMAQdx Grab2.vi
IMAQdx Close Camera.vi
Barcode Types To Search
IMAQ Read Barcode 2
Barcodes
stop
STOP
Image
LR44
Microphone: 18.00mmH x 8.30mmW x 8.30mmD
Cable: 6.0m
Microphone: 2.5g
Power Module: 18g
RoHS
6 971008 020809
640x480 0.56X 32-bit RGB image 75,72,55 (0,0)
Barcode Types To Search
EAN 13
Code 128
Barcodes
Data
7100802080
Special Char 1
6
Special Char 2
9
Checksum(s)
9
Barcode Type
EAN 13
Confidence Level
848,12
Bounding Box
x
335,24
y
324,92

Рис. 6.10

Для роботи із попередньо записаними в файл зображеннями слід використовувати блок «IMAQ Load Image Dialog VI» в якості зручного інтерфейсу для створення шляху до файлу зображення та блок «IMAQ ReadFile 2 VI» для зчитування зображення з файлу із використанням створеного шляху. Попередньо слід за допомогою «IMAQ Create VI» створити посилання на зображення куди буде записані зчитані з файлу дані. Далі використовуємо блок «IMAQ Read Barcode 2 VI» як описано вище (рис. 6.11).

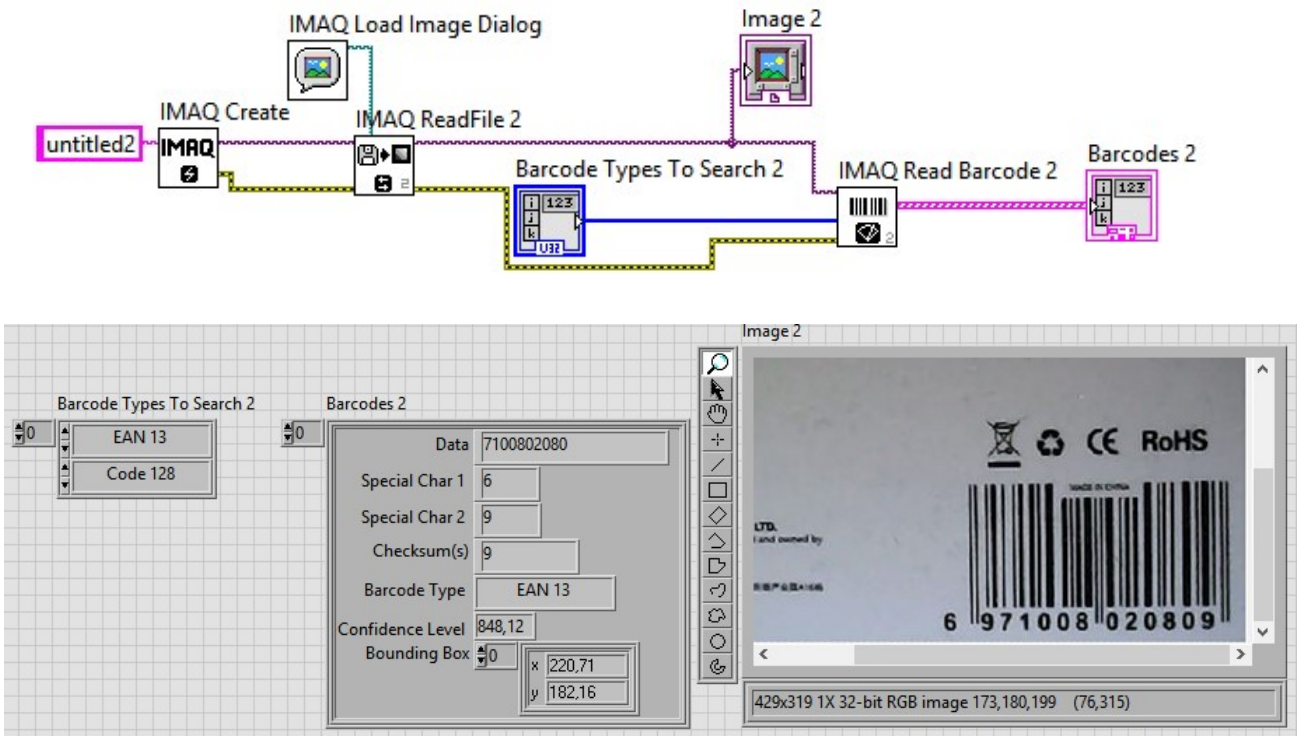


Рис. 6.11

Завдання:

1. Зібрати описану програму і перевірити її роботу;

2. Дослідіть можливості розпізнання частково пошкоджених кодів та нечітких зображень кодів. Приклади штрих-кодів наведено у Додатку 3.

Контрольні питання:

1. Що таке штриховий код?
2. Що таке NI MAX?
3. Що таке NI-IMAQ?

7. СТИСНЕННЯ ДАНИХ У ІНФОРМАЦІЙНИХ СИСТЕМАХ

Мета роботи: *Дослідження методів стиснення даних.*

Стиснення повідомлень застосовується для прискорення передачі та обробки повідомлень, зменшення витрат на обробку, зберігання та пошук інформації, а також зменшення об'єму пам'яті ЕОМ.

Під стисненням повідомлень будемо розуміти операцію, у результаті якої даному повідомленню ставиться у відповідність повідомлення меншої довжини.

Для стиснення повідомлень при передачі даних використовують способи, які дозволяють повністю відновити початковий стан повідомлень або з частковою втратою інформації. Останні використовуються, головним чином, при цифровій обробці сигналів та зображень, тобто графічної інформації (креслень, графіків, діаграм тощо), і тому розглядатися у цьому розділі не будуть.

Різниця між способами стиснення повідомлень, які застосовуються при передачі даних, та способами (архіваторами), які використовуються при архівації повідомлень в ЕОМ, полягає у тому, що при передачі даних оперують з інформаційними масивами значно меншого обсягу (від 32 біт до 2 кбіт), а при архівації - з великими масивами (до 10 і більше кбайт). Викликане це тим, що при передачі даних, як правило, вводяться жорсткі обмеження на час обробки (затримки) повідомлень у передавальному та приймальному пристроях системи передачі даних.

Ефективність стиснення визначається коефіцієнтом стиснення $К_{ст} = N_1/N_2$, де N_1 і N_2 - відповідно кількість бітів (байтів) у первинному і стисненому масивах.

Найбільше поширення зі способів стиснення при архівації набули алгоритми Лемпеля-Зіва та їх різновиди (Лемпеля-Зіва-Велча, Міллера-Ведмана M, AP тощо).

Розглянемо як приклад, алгоритм 77, запропонований А. Лемпелем та Я. Зівом у 1977 році. Основною операцією кодера є пошук рядка у вхідному буфері, який якомога краще співпадав з поточною позицією. Для прискорення процедури пошуку використовують хешування. При цьому по першим двом або трьом символам вхідного потоку визначається деяка величина, яка не дивлячись на те, що є обмеженою зверху, наприклад числом 2000,досить добре характеризує рядок, з якого вона була утворена. Значення цієї величини використовують далі в якості індекса до деякої таблиці, в якій розміщують адреси рядків з визначеними величинами хеш-індекса, в порядку їх виникнення у вхідному буфері.

Якщо тепер при оновленні змісту хеш-таблиці з'ясується, що в ній вже зберігалась адреса деякого рядка, то при цьому залишиться тільки перевірити наскільки цей рядок співпадає з поточною послідовністю.

Для збільшення імовірності знаходження потрібного рядка використовують також перелік дозволення колізій – ланцюга адрес рядків, що мають однакові хеш-індекси. Тобто, якщо найближчий рядок, занесений до хеш-таблиці, в дійсності не співпадає з поточною, то процес пошуку рядка, що найбільш підходить, продовжується вже в переліку колізій. Після закінчення цього аналізу в хеш-таблицю заноситься адреса поточного рядка, а перелік колізій доповнюється рядком, який витискується з хеш-таблиці.

Якщо такий пошук є вдалим, тобто довжина співпадання одного з підрядків з поточною послідовністю символів у вхідному буфері перевищує деякий поріг (звичайно 2 або 3 символи), то у вихідний потік пересилається її код, який складається з величин відстані D та довжини L.

У тому разі, коли ні один з рядків, які розглядаються, не підходить, то у вихідний буфер пересилається поточний символ вхідного потоку і увесь цей процес повторюється для подальших символів.

Для розділення кодів рядків, що співпадають, та окремих символів у вихідний потік подаються також біти ознак рядок/символ.

Ефективність стиснення при використанні окремих способів залежить від інформації, яку стискають, і може змінюватися в широких межах. Так, наприклад, ступень стиснення для різних способів може змінюватися від 1,2 … 1,5 разів для текстової інформації до 6 … 8 разів для чорно-білих зображень. Що стосується швидкості стиснення, то вона для різних способів стиснення може змінюватися від 0,07 … 0,2 Мбайт/с для текстової інформації до 1,5 … 1,9 Мбайт/с для чорно-білих зображень.

Алгоритми стиснення Лемпеля-Зіва реалізовані у програмних продуктах – програмах-архіваторах, які створюють архіви із розширеннями zip та rar.

Для створення zip архіву в системі LabVIEW можна скористатися існуючими в бібліотеках LabVIEW функціями зі створення та зміни zip архівів. Розглянемо задачу із створення архіву який має містити файли із певної теки із збереженням структури теки.

Для початку видаляємо zip файл, який, можливо, вже існує в кінцевій теці. Це робиться для даного навчального прикладу для того щоб уникнути помилки, яка може виникати при повторному створенні архіву. В реальній ситуації проблему створення файлу, ім’я якого збігається із вже існуючим файлом, слід розв’язувати інакше.

Файл буде створено в теці, яка визначена, як тека для зберігання даних системи LabVIEW. Шлях до цієї теки отримуємо за допомогою підпрограми Dflt Data Dir.vi (Default Data Directory VI). Ім’я файлу задаємо як файлову константу “Example Zip File.zip” і додаємо до отриманої теки за

допомогою функції Build Path. Отриманий шлях до визначеного файлу передається до блоку Delete який має видалити вже існуючий файл або проігнорувати команду в разі відсутності такого файлу. Шлях до файлу та ім’я передаємо до індикатору Generated Zip File (Рис. 7.1).

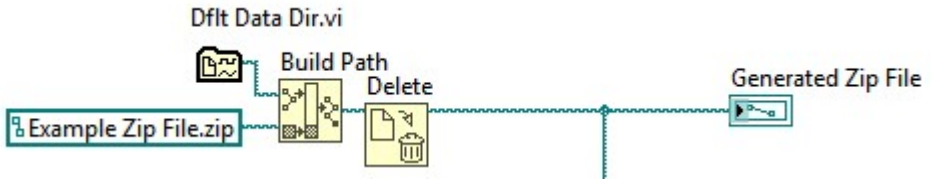


Рис. 7.1

Далі, створюємо шлях до джерела файлів, які будуть стискатися. В даному прикладі це буде тека [LabVIEW]\examples\Synchronization, де [LabVIEW] – тека розташування системи LabVIEW. Для того, щоб її отримати, використаємо блок Property Node на якому лівою кнопкою миші виберемо меню Application \ Directory Path (рис. 7.2).

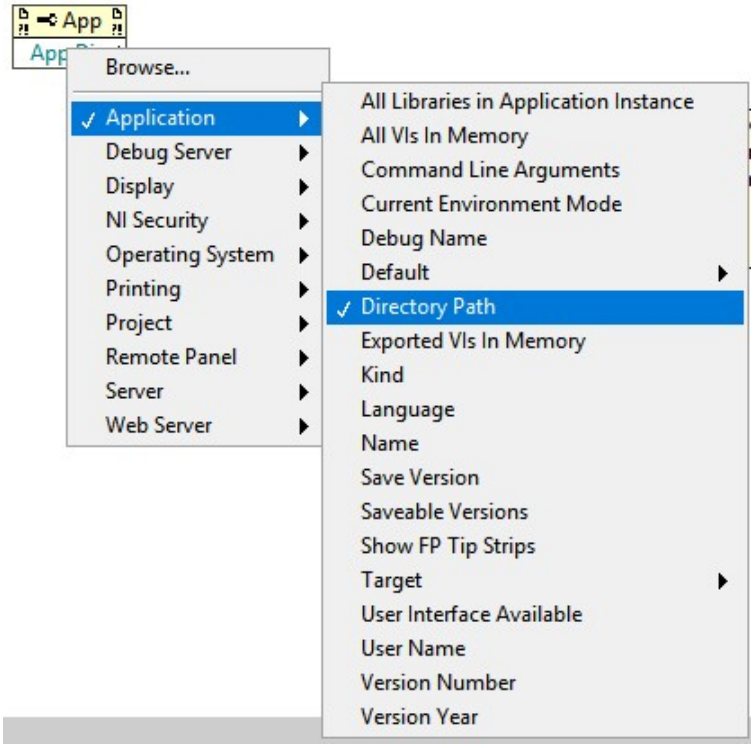


Рис. 7.2

Будуємо шлях до теки із файлами для стиснення за допомогою функції Build Path.

Оскільки файли до архіву будемо додавати по одному, слід отримати список усіх файлів у теці за допомогою блоку Recursive File List.vi. На вхід його терміналу «Folder Path» подаємо побудований шлях до теки, на терміналі-виході «All Files in Dir» отримуємо список (масив) шляхів до усіх файлів в теці.

Далі, створюємо новий Zip архів «Example Zip File.zip» за допомогою New Zip File.vi (рис. 7.3).

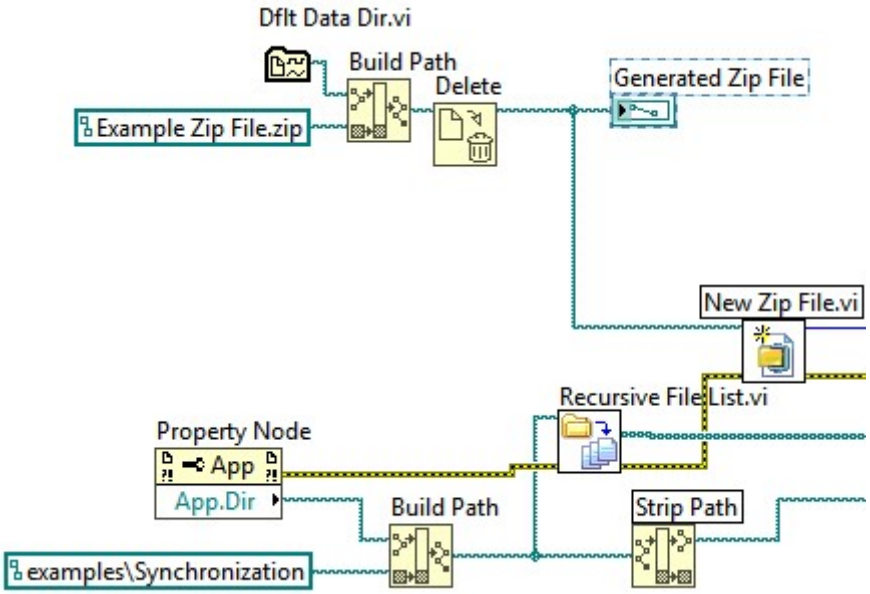


Рис. 7.3

Для побудови відносних шляхів, які будуть збережені у архіві, відокремимо останній елемент шляху до теки із файлами за допомогою Strip Path та порівняємо повні шляхи до файлів із «обрізаним» шляхом у блоці Compare Two Paths.vi, який повертає відносний шлях між шляхами Path 1 та Path 2 «Relative Path from 1 to 2» – такий шлях, який треба задати на диску щоб перейти з Path 1 до Path 2.

Далі збираємо решту програми (рис. 7.4). Окремі файли зі списку будуть додаватися в циклі For, в якому передбачимо наявність умовної

зупинки (Conditional terminal), який обираємо в контекстному меню циклу For (рис. 7.5)

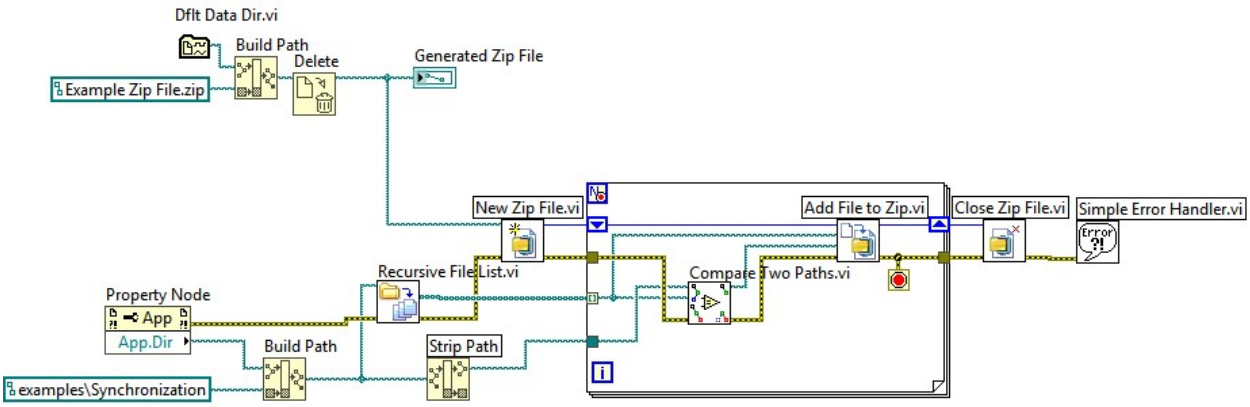


Рис. 7.4

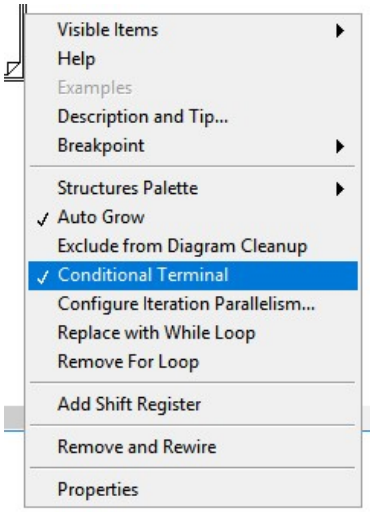


Рис. 7.5

В середині циклу використовуємо блок додавання файлу до архіву «Add File to Zip.vi», до якого передаємо на термінал «zip file in» створений новий Zip-файл, який буде передаватися в циклі через зсувовий регістр. До терміналу «source file path» подається шлях до файлу із масиву. Зверніть увагу, що масив на вході до циклу має індексацію, що означає що на кожній ітерації із масиву буде вибиратися послідовно по одному елементу. Індексацію масиву включаємо у контекстному меню вхідного тунелю циклу For. До терміналу «destination path in zip» подаємо відносний шлях, який буде записаний в архів.

Блок Close Zip File.vi закриває створений архів. Блок «Simple Error Handler» виконує просту обробку помилок, які можуть виникнути при файлових операціях. Інформація про помилки має передаватися між відповідними блоками. Також в циклі було передбачено наявність умовної зупинки (Conditional terminal) – його слід підключити до лінії зв'язку із даними про помилки.

Отже, дана програма створює Zip-архів заданої теки із збереженням файлової структури цієї теки, використовуючи бібліотеки LabVIEW із алгоритмами архівації.

Існує можливість викликати до виконання зовнішні програми з середовища LabVIEW. Наприклад, якщо в операційній системі встановлено програму архівації WinRAR чи WinZip, в комплекті з нею іде консольна версія, яку можна використовувати з командного рядка операційної системи.

В LabVIEW є блок System Exec.vi (рис. 7.6) за допомогою якого можна скористатися командним рядком операційної системи і запустити консольну версію архіватора Rar.exe.

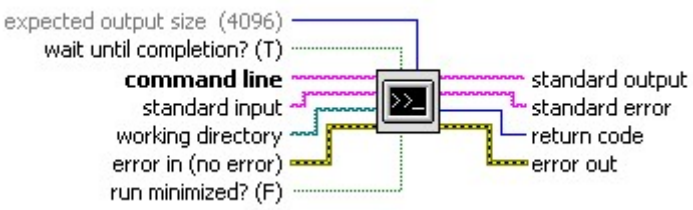


Рис. 7.6

В командному рядку "command line" передаються команди, параметри та шляхи до архіву та файлів, які підлягають архівації. Так, запис в командному рядку

"C:\Program Files (x86)\WinRAR\Rar.exe" a D:\Arch\archiv.rar D:\Book\New.doc*

архівує всі *.doc файли у теці D:\Book\New\ і створить архів з ім’ям archiv.rar у теці D:\Arch\. Зверніть увагу на лапки, в які узятий шлях до *Rar.exe* – вони необхідні для коректної інтерпретації шляху командним рядком.

Програма буде виглядати наступним чином (рис. 7.7).

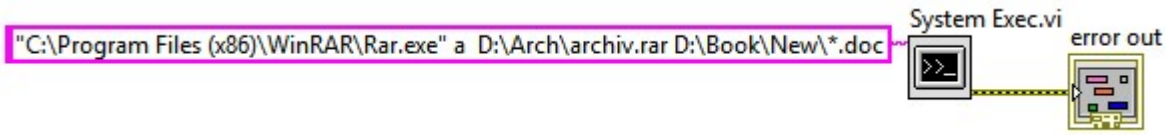


Рис. 7.7

Архіватор Rar.exe має багато параметрів, які дозволяють налаштувати процес стиснення. Так, дана програма, підтримує так звані «ключі» (пишуться в командному рядку із знаком «мінус» перед ними), які визначають методи та алгоритми роботи.

-*m*<n> Метод стиснення.

-m0 Без стиснення – додати файли в архів без стиснення

-m1 Швидкісний – самий швидкий метод (мінімальне стиснення)

-m2 Швидкий – швидкий метод

-m3 Звичайний – Звичайний метод стиснення (за умовчанням)

-m4 Хороший – хороший метод стиснення (більш високий ступінь стиснення)

-m5 Максимальний – найкращий метод стиснення (найкраще, але й повільне стиснення)

Додаткові параметри стиснення.

-*mc*[параметр1][:параметр2][модуль][+ або -]

де <модуль> – односимвольне поле, яке вказує частину алгоритму стиснення, що конфігурується. Приймає значення:

A – стиснення аудіоданих

C – стиснення графічних даних True Color (RGB)

D – дельта- стиснення

E – стиснення файлів архітектури x86, що виконуються

I – стиснення файлів Intel Itanium, що виконуються

T – стиснення текстів

Знак "+" в кінці ключа вказує, що обраний алгоритмічний модуль має бути використаний для всіх даних, а знак "-" відключає цей модуль зовсім. Якщо не вказано ані плюс, ані мінус, RAR обирає модулі автоматично в залежності від даних та поточного метода стиснення.

Ключ *"-mc-"* забороняє використання усіх додаткових модулів та дозволяє застосовувати тільки загальний алгоритм стиснення.

<Параметр1> и <Параметр2> залежать від конкретного модуля.

Стиснення тексту:

Алгоритм стиснення тексту забезпечує помітно більш високий ступінь упаковки простих текстових даних. Однак оскільки цей алгоритм не може ефективно використовувати багатоядерні процесори, він повільніший за основний алгоритм у системах с багатоядерними процесорами, тому алгоритм стиснення тексту за умовчанням вимкнений. Для використання цього алгоритму при упаковці відповідних даних, слід вказати ключ *"-mct"*. Ключ *"-mct+"* примусово вмикає текстове стиснення для всіх даних.

Ключ *"-mct"* може також містити <Параметр1> и <Параметр2>, тому його повний синтаксис такий:

-mc[Параметр1][:Параметр2]*t*[+ або -].

<Параметр1> – порядок алгоритму PPM (от 2 до 63).

Більш високе значення дещо покращує ступінь стиснення надлишкових даних, але тільки якщо є достатньо пам'яті для PPM. Чим вище значення порядку, тим повільніше стиснення та розпакування.

<Параметр2> – пам'ять для алгоритму PPM в мегабайтах (от 1 до 128).

Більш високе значення може покращити ступінь стиснення.

Приклади:

1) Ключ *"-mc10:40t+"* примусово включає для всіх даних алгоритм стиснення тексту, встановлює порядок стиснення 10 та виділяє 40 МБ пам'яті.

2) Ключ *"-mc12t"* встановлює порядок стиснення тексту в значення 12, якщо вступає в дію алгоритм стиснення тексту, але лишає RAR можливість вирішувати, коли саме слід його використовувати.

3) Ключі *"-mct -mcd-"* дозволяють RAR застосувати алгоритм стиснення текста для даних, що підходять, та відключати дельта-стиснення.

-md<n>[*k,m,g*]

Розмір словника в КБ, МБ або ГБ.

Динамічний словник – це спеціальна область пам'яті, що використовується алгоритмом стиснення. Якщо розмір файлу що стискається більший розміру словника, то збільшення розміру словника

зазвичай призводить до поліпшення ступеню стиснення, зменшенню швидкості обробки та збільшенню вимог до обсягу доступної пам’яті.

Для архивів формату RAR5 розмір словника може приймати значення: 128/256/512 КБ, 1/2/4/8/16/32/64/128/256/512 МБ, 1 ГБ.

Для зазначення розміру в кіло-, мега- та гігабайтах можна використовувати модифікатори "k", "m" и "g". Наприклад, ключ *"-md64m"* задає словник розміром 64 МБ. Якщо модифікатор не вказаний, застосовують мегабайти, тобто ключі *"-md64m"* и *"-md64"* еквівалентні.

При архівації пам’яті необхідно в шість разів більше, ніж вказаний розмір словника. При розпаковці витрачається пам’яті дещо більше, ніж один розмір словника.

Якщо розмір всіх вихідних файлів для неперервного архіву або розмір найбільшого вихідного файлу для не неперервного архіву хоча б вдвічі менше заданого розмірг словника, то RAR може зменшити розмір словника. Це знижує вимоги до пам’яті, не погіршуючи стиснення.

За умовчанням розмір динамічного словника для формату RAR4 складає 4 МБ, а для RAR5 – 32 МБ.

Приклад:

*RAR a -s -ma -md128 lib *.dll*

Ця команда створить неперервний архів RAR5 зі словником 128 МБ.

Докладніше про команди та параметри архіватора Rar.exe слід дивитися у довіднику користувача.

Завдання:

1. Зібрати описані програми і перевірити їх роботу;

2. Порівняти ефективність стиснення зазначених програмних реалізацій алгоритмів для створення zip та rar архівів для різних типів

даних (текст, таблиця, зображення, відео). Визначити коефіцієнти стиснення алгоритмів.

Контрольні питання:

1. Що таке стиснення даних?

2. Як визначити коефіцієнт стиснення?

3. Який блок дозволяє викликати командний рядок операційної системи?

Що таке обробка помилок?

Список рекомендованої літератури

1. Jeffrey Travis, Jim Kring. LabVIEW for Everyone: Graphical Programming Made Easy and Fun - Prentice Hall, 2006. - ISBN: 9780131856721; 0131856723

2. Larsen, Ronald W. LabVIEW for engineers.- Prentice Hall/Pearson, 2010. ISBN: 9780136094296; 0136094295

3. Riccardo De Asmundis. Modeling, Programming and Simulations Using LabVIEW Software -InTech, 2011.

4. LabVIEW Basics II - Development Course Manual. – National Instruments Corporation, 2008. – 197 с. (Електроний ресурс https://learn.ni.com/learn/article/labview-tutorial).

5. Mütterlein, Bernward. Handbuch für die Pogrammierung mit LabVIEW [mit Studentenversion LabVIEW 8.6]. – Heidelberg : Spektrum Akademischer Verlag, 2009. – 516 S. – ISBN 978-3-8274-2337-5

6. Thomas Klinger. Image Processing with LabVIEW and IMAQ Vision. – Prentice Hall PTR, 2003. – 319 S. – ISBN 0-13-047415-0

7. Relf, Christopher G. Image acquisition and processing with LabVIEW. – CRC Press, 2003. – 268 S. – ISBN 0-8493-1480-1

8. Bitter, Rick. LabVIEW: advanced programming techniques / Richard Bitter, Taqi Mohiuddin, Matthew R. Nawrocki. - 2nd ed. – CRC Press, 2006. – 520 S. – ISBN 0-8493-3325-3

9. Жураковський Ю. П. Теорія інформації та кодування: Підручник / Ю. П. Жураковський, В. П. Полторак. –К.: Вища шк., 2001. – 255 с.

10. LabVIEW: A Developer's Guide to Real World Integration/ Edited by I. Fairweather, A. Brumfield– CRC Press, 2012. – 271 S. – ISBN 978-1-4398-3982-9

Додаток 1. Formula Node and Expression Node Functions

abs(x)	Absolute Value	Returns the absolute value of x.
acos(x)	Inverse Cosine	Computes the inverse cosine of x in radians.
acosh(x)	Inverse Hyperbolic Cosine	Computes the inverse hyperbolic cosine of x in degrees.
asin(x)	Inverse Sine	Computes the inverse sine of x in radians.
asinh(x)	Inverse Hyperbolic Sine	Computes the inverse hyperbolic sine of x in degrees.
atan(x)	Inverse Tangent	Computes the inverse tangent of x in radians.
atanh(x)	Inverse Hyperbolic Tangent	Computes the inverse hyperbolic tangent of x in degrees.
ceil(x)	Round to +Infinity	Rounds x to the next higher integer (smallest integer is greater than or equal to x).
cos(x)	Cosine	Computes the cosine of x in radians.
cosh(x)	Hyperbolic Cosine	Computes the hyperbolic cosine of x in degrees.
cot(x)	Cotangent	Computes the cotangent of x in radians (1/tan(x)).
csc(x)	Cosecant	Computes the cosecant of x in radians (1/sin(x)).
exp(x)	Exponential	Computes the value of e raised to the x power.
expm1(x)	Exponential (Arg - 1)	Computes the value of e raised to the x power minus one (e^(x-1)).
floor(x)	Round to -Infinity	Truncates x to the next lower integer (largest integer is smaller than or equal to x).
getexp(x)	Mantissa & Exponent	Returns the exponent of x.
getman(x)	Mantissa & Exponent	Returns the mantissa of x.
int(x)	Round To Nearest	Rounds its argument to the nearest integer.
intrz(x)	Round Toward 0	Rounds x to the nearest integer between x and zero.
ln(x)	Natural Logarithm	Computes the natural logarithm of x (to the base e).
lnp1(x)	Natural Logarithm (Arg +1)	Computes the natural logarithm of (x + 1).
log(x)	Logarithm Base 10	Computes the logarithm of x (to the base of 10).
log2(x)	Logarithm Base 2	Computes the logarithm of x (to the base 2).
max(x,y)	Maximum	Compares x and y and returns the larger value.
min(x,y)	Minimum	Compares x and y and returns the smaller value.
mod(x,y)	Quotient & Remainder	Computes the remainder of x/y, when the quotient is rounded toward -Infinity.
power(x,y)	x^y	Computes the value of x raised to the y power.
rand( )	Random Number (0- 1)	Produces a floating-point number between 0 and 1 exclusively.
rem(x,y)	Remainder	Computes the remainder of x/y, when the quotient is rounded to the nearest integer.
sec(x)	Secant	Computes the secant of x, where x is in radians (1/cos(x)).
sign(x)	Sign	Returns 1 if x is greater than 0, returns 0 if x is equal to 0, and returns -1 if x is less than 0.
sin(x)	Sine	Computes the sine of x, where x is in radians.
sinc(x)	Sinc	Computes the sine of x divided by x radians (sin(x)/x).
sinh(x)	Hyperbolic Sine	Computes the hyperbolic sine of x in degrees.
sqrt(x)	Square Root	Computes the square root of x.
tan(x)	Tangent	Computes the tangent of x in radians.
tanh(x)	Hyperbolic Tangent	Computes the hyperbolic tangent of x in degrees.

Додаток 2. Специфікатори формату

Синтаксис елемента	Опис
%	Починає специфікатор формату.
- (optional)	Задає вирівнювання параметру по лівому краю, а не вирівнювати по правому в межах своєї ширини
+ (optional)	Використовується з числовими параметрами, включає в себе знак, навіть якщо число додатне.
^ (optional)	Використовується з кодами перетворення e або g, використовує інженерне позначення. `Precision` повинна бути кратна 3.
0 (optional)	Заповнити надлишок місця зліва від числового параметру нулями замість пробілів.
`Width` (optional)	При використанні функції сканування, визначає точну ширину поля для використання. LabVIEW сканує тільки вказану кількість символів при обробці параметра. При використанні функції форматування, вказує мінімальну ширину поля символу виводу. Ця ширину не максимальна ширина. LabVIEW використовує стільки символів, скільки необхідно для форматування параметр без його урізання. LabVIEW заповнює поля ліворуч або праворуч від параметра пробілами, в залежності від вирівнювання. Якщо `Width` відсутній або дорівнює 0, на виході буде тільки стільки символів, скільки необхідно, щоб містити перетворений вхідний параметр.
.	Відділяє `Width` від `Precision`.
`Precision` (optional)	Використовується з параметрами з плаваючою точкою, визначає кількість цифр праворуч від десяткової точки. Якщо `Width` не стоїть після крапки, LabVIEW вставляє дробову частину з шести цифр. Якщо `Width` стоїть після крапки і `Precision` відсутній або дорівнює 0, LabVIEW не вставляє дробову частину. Використовується з параметрами рядка, визначає максимальну ширину поля. LabVIEW усікає рядки більші за цю довжину.
`{unit}` (optional)	Замінює оригінальну одиницю вимірювання BI при використанні функції перетворення фізичної величини (значення з відповідною одиницею вимірювання). Мають бути сумісні одиницю вимірювання.

Conversion Codes	Один символ, який визначає, як сканувати або форматувати параметр, а саме:
d x o b f e g s	decimal integer hex integer octal integer binary integer floating-point number with fractional format floating-point number with scientific notation floating-point number використовує формат e якщо експонента менше -4 або більше Precision, або формат f в іншому випадку string
Символи перетворення можуть бути у верхньому або нижньому регістрі.	
Localization Codes	Наступні коди визначають роздільник цілої і дробової частини для чисел:
%,; %.; %;	кома крапка за умовчанням у системі Ці коди не призводять до якихось змін на вхід або виході. Вони змінюють цілої і дробової частини десяткових чисел для всіх наступних входів / виходів доки не надійдуть нові символи «%;».

LabVIEW використовує коди перетворення для визначення текстового формату параметра. Наприклад, специфікатор формату %x перетворює шестнатцяткове ціле число в рядок або навпаки.

Використовуйте коди перетворення d, x, o, b, f, e, і g для обробки будь-якого числового типу даних LabVIEW, у тому числі комплексних чисел, булевих типів даних і типів enumerated.

Для комплексних чисел використовують специфікатор формату для обробки реальної та уявної частини, як одного параметра.

Використовують код перетворення s для обробки рядків або шляхів, логічних типів даних або типів enumerated.

Зверніть увагу, що можна використовувати або числовий або строковий код перетворення з типу enumerated або логічним типом, залежно від того, чи хочете отримати числове значення або символічне значення (рядок) типу enumerated.

Додаток 3. Приклади штрих-кодів

Code 128 (FORMAT_CODE_128)

Code 39 (FORMAT_CODE_39)

Code 93 (FORMAT_CODE_93)

Codabar (FORMAT_CODABAR)

EAN-13 (FORMAT_EAN_13)

EAN-8 (FORMAT_EAN_8)

ITF (FORMAT_ITF)

UPC-A (FORMAT_UPC_A)

UPC-E (FORMAT_UPC_E)

Data Matrix (FORMAT_DATA_MATRIX)

QR Code (FORMAT_QR_CODE)

PDF417 (FORMAT_PDF417)

Aztec (FORMAT_AZTEC)